
Python3 solutions for codewars problems

Release 0.2

Egor Kostan

Oct 16, 2020

CONTENTS:

1	codewars	1
1.1	img package	1
1.2	kyu_2 package	1
1.3	kyu_3 package	2
1.4	kyu_4 package	7
1.5	kyu_5 package	17
1.6	kyu_6 package	36
1.7	kyu_7 package	61
1.8	kyu_8 package	83
1.9	utils package	105
2	Indices and tables	107
	Python Module Index	109
	Index	117

CODEWARS

1.1 img package

1.1.1 Module contents

1.2 kyu_2 package

1.2.1 Subpackages

kyu_2.evaluate_mathematical_expression package

Submodules

kyu_2.evaluate_mathematical_expression.evaluate module

Evaluate mathematical expression.

Given a mathematical expression as a string you must return the result as a number.

kyu_2.evaluate_mathematical_expression.evaluate.**calc**(string: str) → float

kyu_2.evaluate_mathematical_expression.evaluate.**calculate**(i: int, char: str, strings: list)

kyu_2.evaluate_mathematical_expression.evaluate.**normalize_string**(string: str) → str

kyu_2.evaluate_mathematical_expression.evaluate.**process_brakets**(string)

kyu_2.evaluate_mathematical_expression.evaluate.**process_duplicate_minus**(string: str) → str

kyu_2.evaluate_mathematical_expression.evaluate.**process_math_expression**(string: str, operators: list) → str

kyu_2.evaluate_mathematical_expression.test_evaluate module

Testing calc method

```
class kyu_2.evaluate_mathematical_expression.test_evaluate.CalcTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing calc method

```
test_calc()
```

Testing calc class

Given a mathematical expression as a string you must return the result as a number.

Module contents

1.2.2 Module contents

1.3 kyu_3 package

1.3.1 Subpackages

kyu_3.calculator package

Submodules

kyu_3.calculator.calculator module

Create a simple calculator that given a string of operators `()`, `+`, `-`, `*`, `/` and numbers separated by spaces returns the value of that expression

```
class kyu_3.calculator.calculator.Calculator  
    Bases: object
```

Given string of operators `()`, `+`, `-`, `*`, `/` and numbers separated by spaces. Returns the value of that expression.

```
__calculate (char: str, strings: list)
```

1. Perform math operation
2. Reorganize math expression

Parameters

- **i** – char (math operation) index
- **char** – math operation
- **strings** – math expression

Returns result

```
__process_math_expression (string: str, operators: list) → str
```

Perform all operation with: multiplications, divisions, additions and subtractions

Parameters **string** – input string

Returns output string with no `*`, `/`, `+`, `-`

evaluate (*string: str*) → float

Returns value of the given expression

Parameters **string** – a string of operators (), +, -, *, / and numbers separated by spaces

Returns calculated value of the given expression

kyu_3.calculator.test_calculator module

Testing Calculator class

class kyu_3.calculator.test_calculator.**CalculatorTestCase** (*methodName='runTest'*)

Bases: unittest.case.TestCase

Testing Calculator class

test_calculator ()

Testing Calculator class

A simple calculator that given a string of operators (), +, -, *, / and numbers separated by spaces will return the value of that expression

Module contents

kyu_3.rail_fence_cipher_encoding_and_decoding package

Submodules

kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding module

kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding.**decode_rail_fence_cipher**

Function/method that takes 2 arguments, an encoded string and the number of rails, and returns the DECODED string.

Parameters

- **string** – an encoded string
- **n** – the number of rails

Returns the DECODED string

kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding.**encode_rail_fence_cipher**

This cipher is used to encode a string by placing each character successively in a diagonal along a set of “rails”. First start off moving diagonally and down. When you reach the bottom, reverse direction and move diagonally and up until you reach the top rail. Continue until you reach the end of the string. Each “rail” is then read left to right to derive the encoded string.

Parameters

- **string** – a string
- **n** – the number of rails

Returns the ENCODED string

```
kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding.get_rails (string:  
                                                                              str,  
                                                                              n:  
                                                                              int)  
→  
list
```

Create rails matrix.

Parameters

- **string** – a string
- **n** – the number of rails

Returns rails matrix

kyu_3.rail_fence_cipher_encoding_and_decoding.test_decoding module

Testing Decoding functionality

```
class kyu_3.rail_fence_cipher_encoding_and_decoding.test_decoding.DecodingTestCase (methodNam  
    Bases: unittest.case.TestCase  
    Testing Decoding functionality  
    test_decoding ()  
        Testing Decoding functionality
```

kyu_3.rail_fence_cipher_encoding_and_decoding.test_encoding module

Testing Encoding functionality

```
class kyu_3.rail_fence_cipher_encoding_and_decoding.test_encoding.EncodingTestCase (methodNam  
    Bases: unittest.case.TestCase  
    Testing Encoding functionality  
    test_encoding ()  
        Testing Encoding functionality
```

Module contents

kyu_3.make_spiral package

Submodules

kyu_3.make_spiral.solution module

```
kyu_3.make_spiral.solution.down (spiral: list, coordinates: dict) → bool  
    Move spiral down
```


Parameters

- **coordinates** – starting point
- **spiral** – NxN spiral 2D array

Returns boolean 'done'

`kyu_3.make_spiral.solution.left(spiral: list, coordinates: dict) → bool`
 Move spiral left

Parameters

- **coordinates** – starting point
- **spiral** – NxN spiral 2D array

Returns None

`kyu_3.make_spiral.solution.right(spiral: list, coordinates: dict) → bool`
 Move spiral right

Parameters

- **coordinates** – starting point
- **spiral** – NxN spiral 2D array

Returns boolean 'done'

`kyu_3.make_spiral.solution.set_initial_params(size: int) → tuple`
 Set initial parameters: line, spiral, direction, coordinate, done

Parameters **size** –**Returns**

`kyu_3.make_spiral.solution.spiralize(size: int) → list`
 Creates a NxN spiral 2D list with a given size

Parameters **size** – size of the 2D array**Returns** NxN spiral 2D array

`kyu_3.make_spiral.solution.up(spiral: list, coordinates: dict) → bool`
 Move spiral up

Parameters

- **coordinates** – starting point
- **spiral** – NxN spiral 2D array

Returns None**kyu_3.make_spiral.test_spiralize module**

Testing spiralize function

class `kyu_3.make_spiral.test_spiralize.SpiralizeTestCase` (*methodName='runTest'*)
 Bases: `unittest.case.TestCase`

Testing spiralize function

test_spiralize()
 Testing spiralize function

Module contents

kyu_3.battleship_field_validator package

Submodules

kyu_3.battleship_field_validator.test_battleship_validator module

Testing Battleship field validator

```
class kyu_3.battleship_field_validator.test_battleship_validator.BattleshipFieldValidatorTest(unittest.case.TestCase)
```

Testing Battleship field validator

```
test_validate_battlefield()  
Testing Battleship field validator
```

Testing a method that takes a field for well-known board game “Battleship” as an argument and returns true if it has a valid disposition of ships, false otherwise. Argument is guaranteed to be 10*10 two-dimension array. Elements in the array are numbers, 0 if the cell is free and 1 if occupied by ship.

kyu_3.battleship_field_validator.validator module

```
kyu_3.battleship_field_validator.validator.is_valid_cell(ships: dict, field: list,  
                                                         cell: list, direction: str)  
                                                         → bool
```

Validates if single cell result is valid (valid submarine or single ship cell)

Parameters

- **ships** – collection of valid ships (dict)
- **field** – board game “Battleship” (list)
- **cell** – candidate for single ship/submarine
- **direction** – str -> horizontal, vertical, submarine

Returns

```
kyu_3.battleship_field_validator.validator.ship_counter_by_col(field: list, ships:  
                                                             dict)
```

```
kyu_3.battleship_field_validator.validator.ship_counter_by_row(field: list, ships:  
                                                             dict)
```

```
kyu_3.battleship_field_validator.validator.validate_battlefield(field: list) →  
                                                                    bool
```

A method that takes a field for well-known board game “Battleship” as an argument and returns true if it has a valid disposition of ships, false otherwise. Argument is guaranteed to be 10*10 two-dimension array. Elements in the array are numbers, 0 if the cell is free and 1 if occupied by ship.

Parameters **field** – board game “Battleship” (2D list)

Returns returns true if it has a valid disposition of ships, false otherwise

Module contents

1.3.2 Module contents

1.4 kyu_4 package

1.4.1 Subpackages

kyu_4.sum_of_intervals package

Submodules

kyu_4.sum_of_intervals.sum_of_intervals module

`kyu_4.sum_of_intervals.sum_of_intervals.clean_interval (intervals, i, b) → bool`

`kyu_4.sum_of_intervals.sum_of_intervals.remove_overlaps (intervals: list) → list`
Remove overlaps and duplicates :param intervals: :return:

`kyu_4.sum_of_intervals.sum_of_intervals.sum_of_intervals (intervals: list) → int`
Accepts an array of intervals, and returns the sum of all the interval lengths.

Overlapping intervals should only be counted once. :param intervals: :return:

kyu_4.sum_of_intervals.test_sum_of_intervals module

class `kyu_4.sum_of_intervals.test_sum_of_intervals.SumOfIntervalsTestCase (methodName='runTest')`
Bases: `unittest.case.TestCase`

Testing sum_of_intervals function

test_sum_of_intervals ()
Testing sum_of_intervals function

The function should accept an array of intervals, and return the sum of all the interval lengths.

Overlapping intervals should only be counted once.

Intervals Intervals are represented by a pair of integers in the form of an array. The first value of the interval will always be less than the second value. Interval example: [1, 5] is an interval from 1 to 5. The length of this interval is 4. :return:

Module contents

kyu_4.human_readable_duration_format package

Submodules

kyu_4.human_readable_duration_format.format_duration module

A function which formats a duration, given as a number of seconds, in a human-friendly way.

`kyu_4.human_readable_duration_format.format_duration.calc_days (seconds: int) → int`

Calculate days

Parameters `seconds` –

Returns

`kyu_4.human_readable_duration_format.format_duration.calc_hours (seconds: int) → int`

Calculate hours

Parameters `seconds` –

Returns

`kyu_4.human_readable_duration_format.format_duration.calc_minutes (seconds: int) → int`

calculate minutes

Parameters `seconds` –

Returns

`kyu_4.human_readable_duration_format.format_duration.calc_seconds (seconds: int) → int`

Calculate seconds

Parameters `seconds` –

Returns

`kyu_4.human_readable_duration_format.format_duration.calc_years (seconds: int) → int`

Calculate years

Parameters `seconds` –

Returns

`kyu_4.human_readable_duration_format.format_duration.format_duration (seconds: int) → str`

A function which formats a duration, given as a number of seconds, in a human-friendly way.

The resulting expression is made of components like 4 seconds, 1 year, etc. In general, a positive integer and one of the valid units of time, separated by a space. The unit of time is used in plural if the integer is greater than 1.

The components are separated by a comma and a space (“,”). Except the last component, which is separated by ” and “, just like it would be written in English.

A more significant units of time will occur before than a least significant one. Therefore, 1 second and 1 year is not correct, but 1 year and 1 second is.

Different components have different unit of times. So there is not repeated units like in 5 seconds and 1 second.

A component will not appear at all if its value happens to be zero. Hence, 1 minute and 0 seconds is not valid, but it should be just 1 minute.

A unit of time must be used “as much as possible”. It means that the function should not return 61 seconds, but 1 minute and 1 second instead. Formally, the duration specified by of a component must not be greater than any valid more significant unit of time.

Parameters `seconds` –

Returns

```
kyu_4.human_readable_duration_format.format_duration.get_string(number: int,
                                                                string: str) →
                                                                str
```

Concatenate string result

Parameters

- **number** –
- **string** –

Returns

kyu_4.human_readable_duration_format.test_format_duration module

```
class kyu_4.human_readable_duration_format.test_format_duration.FormatDurationTestCase(meth
Bases: unittest.case.TestCase
```

Testing format_duration

```
test_format_duration()
```

Test a function which formats a duration, given as a number of seconds, in a human-friendly way.

The function must accept a non-negative integer. If it is zero, it just returns “now”. Otherwise, the duration is expressed as a combination of years, days, hours, minutes and seconds. :return:

Module contents

kyu_4.sudoku_solution_validator package

Submodules

kyu_4.sudoku_solution_validator.test_valid_solution module

```
class kyu_4.sudoku_solution_validator.test_valid_solution.ValidSolutionTestCase(methodName=''
Bases: unittest.case.TestCase
```

Testing validSolution function

```
test_valid_solution()
```

Test a function validSolution/ValidateSolution/valid_solution() that accepts a 2D array representing a Sudoku board, and returns true if it is a valid solution, or false otherwise. The cells of the sudoku board may also contain 0's, which will represent empty cells. Boards containing one or more zeroes are considered to be invalid solutions.

The board is always 9 cells by 9 cells, and every cell only contains integers from 0 to 9. :return:

kyu_4.sudoku_solution_validator.valid_solution module

kyu_4.sudoku_solution_validator.valid_solution.**test_horizontally**(*board: list*) → bool
test horizontally

kyu_4.sudoku_solution_validator.valid_solution.**test_sub_grids**(*board: list*) → bool
test each of the nine 3x3 sub-grids (also known as blocks)

kyu_4.sudoku_solution_validator.valid_solution.**test_vertically**(*board: list*) → bool
test vertically

kyu_4.sudoku_solution_validator.valid_solution.**validSolution**(*board: list*) → bool
A function validSolution/ValidateSolution/valid_solution() that accepts a 2D array representing a Sudoku board, and returns true if it is a valid solution, or false otherwise :param board: :return:

Module contents

kyu_4.range_extraction package

Submodules

kyu_4.range_extraction.solution module

kyu_4.range_extraction.solution.**solution**(*args: list*) → str

kyu_4.range_extraction.test_solution module

```
class kyu_4.range_extraction.test_solution.SolutionTestCase(methodName='runTest')
    Bases: unittest.case.TestCase
    test_solution()
        Testing solution function
```

Module contents

kyu_4.validate_sudoku_with_size package

Submodules

kyu_4.validate_sudoku_with_size.sudoku module

```
class kyu_4.validate_sudoku_with_size.sudoku.Sudoku(data: list)
    Bases: object
    is_valid() → bool
        A method to validate if given a Sudoku has been filled out correctly. Sudoku: data structure with size NxN,
        N > 0 and N == integer. :return:
```

kyu_4.validate_sudoku_with_size.test_sudoku module

```
class kyu_4.validate_sudoku_with_size.test_sudoku.SudokuTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
    Testing Sudoku class  
  
    test_sudoku_class()  
        Testing Sudoku class  
  
        Given a Sudoku data structure with size NxN, N > 0 and N == integer, assert a method that validates if it  
        has been filled out correctly. :return:
```

Module contents

kyu_4.strip_comments package

Submodules

kyu_4.strip_comments.solution module

```
kyu_4.strip_comments.solution.solution (string: str, markers: list) → str  
    The solution strips all text that follows any of a set of comment markers passed in. Any whitespace at the end  
    of the line will be stripped out as well.
```

Parameters

- **string** –
- **markers** –

Returns

kyu_4.strip_comments.test_solution module

```
class kyu_4.strip_comments.test_solution.SolutionTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
  
    test_solution()  
        Testing 'solution' function  
  
        The solution should strips all text that follows any of a set of comment markers passed in. Any whitespace  
        at the end of the line should also be stripped out.
```

Module contents

kyu_4.snail package

Submodules

kyu_4.snail.snail_sort module

Returns the array elements arranged from outermost elements to the middle element, traveling clockwise.

`kyu_4.snail.snail_sort.snail (snail_map: list) → list`

Returns the array elements arranged from outermost elements to the middle element, traveling clockwise.

Parameters `snail_map` – n x n array

Returns array elements arranged from outermost elements to the middle element, traveling clockwise

`kyu_4.snail.test_snail` module

class `kyu_4.snail.test_snail.SnailTestCase (methodName='runTest')`

Bases: `unittest.case.TestCase`

test_snail ()

Testing 'snail' function

Given an n x n array, 'snail' function should return the array elements arranged from outermost elements to the middle element, traveling clockwise.

Module contents

`kyu_4.sum_by_factors` package

Submodules

`kyu_4.sum_by_factors.sum_for_list` module

`kyu_4.sum_by_factors.sum_for_list.sum_for_list (lst: list) → list`

Given an array of positive or negative integers $I = [i_1, \dots, i_n]$ the function have to produce a sorted array P of the form:

$[[p, \text{sum of all } i_j \text{ of } I \text{ for which } p \text{ is a prime factor (} p \text{ positive) of } i_j] \dots]$

P will be sorted by increasing order of the prime numbers.

Parameters `lst` – an array of positive or negative integers

Returns sorted array P

`kyu_4.sum_by_factors.test_sum_for_list` module

Testing `sum_for_list` function

class `kyu_4.sum_by_factors.test_sum_for_list.SumForListTestCase (methodName='runTest')`

Bases: `unittest.case.TestCase`

Testing `sum_for_list` function

test_sum_for_list ()

Testing `sum_for_list` function :return:

Module contents

kyu_4.most_frequently_used_words package

Submodules

kyu_4.most_frequently_used_words.solution module

Most frequently used words in a text

`kyu_4.most_frequently_used_words.solution.top_3_words(text: str) → list`

Given a string of text (possibly with punctuation and line-breaks), returns an array of the top-3 most occurring words, in descending order of the number of occurrences.

Parameters `text` – a string of text

Returns an array of the top-3 most occurring words

kyu_4.most_frequently_used_words.test_top_3_words module

class `kyu_4.most_frequently_used_words.test_top_3_words.Top3WordsTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing `top_3_words`

test_top_3_words()

Test `top_3_words` function

Module contents

kyu_4.the_greatest_warrior package

Submodules

kyu_4.the_greatest_warrior.test_battle module

class `kyu_4.the_greatest_warrior.test_battle.BattleTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing Battle method

test_battle()

Testing Battle method

kyu_4.the_greatest_warrior.test_warrior module

```
class kyu_4.the_greatest_warrior.test_warrior.WarriorTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing Warrior class

    test_warrior_bruce_lee()
        Testing Warrior class >>> bruce_lee

    test_warrior_tom()
        Testing Warrior class >>> tom
```

kyu_4.the_greatest_warrior.warrior module

The Greatest Warrior

```
class kyu_4.the_greatest_warrior.warrior.Warrior
    Bases: object

    A class called Warrior which calculates and keeps track of level and skills, and ranks.

    __set_level() → int
        A warrior starts at level 1 and can progress all the way to 100.
        A warrior cannot progress beyond level 100.
        Each time the warrior's experience increases by another 100, the warrior's level rises to the next level.

        Returns

    __set_rank() → str
        Returns warrior's experience

    __update_experience(experience: int)
        A warrior's experience is cumulative, and does not reset with each rise of level. The only exception is
        when the warrior reaches level 100, with which the experience stops at 10000. :return:

    property achievements

    battle(enemy_level: int)

    property experience

    property level
        A warrior's level

        Returns A warrior's level

    property rank
        A warrior starts at rank "Pushover" and can progress all the way to "Greatest"

        Returns warrior's rank

    training(params: list) → str

        Training will accept an array of three elements: the description, the experience points your warrior
        earns, and the minimum level requirement.

        Parameters params –

        Returns
```

Module contents

kyu_4.strings_mix package

Submodules

kyu_4.strings_mix.solution module

Strings Mix

kyu_4.strings_mix.solution.get_counters(*s: str*) → dict

kyu_4.strings_mix.solution.mix(*s1: str, s2: str*) → str

Given two strings *s1* and *s2*, we want to visualize how different the two strings are. We will only take into account the lowercase letters (a to z). First let us count the frequency of each lowercase letters in *s1* and *s2*.
:param *s1*: string *a* :param *s2*: string *b* :return: the difference between two strings

kyu_4.strings_mix.solution.sort_results(*results: list*) → list

The results will be in decreasing order of their length and when they have the same length sorted in ascending lexicographic order (letters and digits - more precisely sorted by code-point) :param *results*: :return:

kyu_4.strings_mix.test_mix module

Testing 'mix' function

```
class kyu_4.strings_mix.test_mix.MixTestCase (methodName='runTest')
```

Bases: unittest.case.TestCase

```
test_smix()
```

Testing 'mix' function

Given two strings *s1* and *s2*, the 'mix' function should visualize how different the two strings are.

Module contents

kyu_4.next_smaller_number_with_the_same_digits package

Submodules

kyu_4.next_smaller_number_with_the_same_digits.next_smaller module

kyu_4.next_smaller_number_with_the_same_digits.next_smaller.find_x(*n: int*) →
int

kyu_4.next_smaller_number_with_the_same_digits.next_smaller.find_y(*n: int, x_i: int*) → int

kyu_4.next_smaller_number_with_the_same_digits.next_smaller.next_smaller(*n: int*) →
int

A function that takes a positive integer and returns the next smaller positive integer containing the same digits.

If no smaller number can be composed using those digits, return -1

kyu_4.next_smaller_number_with_the_same_digits.test_next_smaller module

```
class kyu_4.next_smaller_number_with_the_same_digits.test_next_smaller.NextSmallerTestCase
    Bases: unittest.case.TestCase

    test_next_smaller()
        Testing next_smaller function

        You have to test a function that takes a positive integer number and returns the next smaller number formed
        by the same digits:

        21 ==> 12 531 ==> 513 2071 ==> 2017

        If no smaller number can be composed using those digits, return -1
```

Module contents

kyu_4.next_bigger_number_with_the_same_digits package

Submodules

kyu_4.next_bigger_number_with_the_same_digits.next_bigger module

```
kyu_4.next_bigger_number_with_the_same_digits.next_bigger.digit_that_breaks_ordering_index
```

Starting from last digit of given number, find the first digit which breaks the sorted ordering. Let the index of this found digit be 'i' and the digit be number[i].

Parameters **digits** – list of digits

Returns the index of the first digit which breaks the sorted ordering

```
kyu_4.next_bigger_number_with_the_same_digits.next_bigger.next_bigger(n:
                                                                    int)
                                                                    → int
```

A function that takes a positive integer number and returns the next bigger number formed by the same digits.

If no bigger number can be composed using those digits, return -1

```
kyu_4.next_bigger_number_with_the_same_digits.next_bigger.next_greater_digit_index(digits:
                                                                                    list,
                                                                                    i:
                                                                                    int)
                                                                                    →
                                                                                    int
```

Find the next greater digit in the right portion of number[i] - that is from digit at index i+1 to last digit. Let that digit be number[j] at index 'j'.

Parameters

- **digits** – list of digits
- **i** – index of number[i]

Returns next greater digit in the right portion of number[i]

kyu_4.next_bigger_number_with_the_same_digits.test_next_bigger module

```
class kyu_4.next_bigger_number_with_the_same_digits.test_next_bigger.NextBiggerTestCase (met  
    Bases: unittest.case.TestCase  
  
    test_next_bigger()  
        Testing next_bigger function  
  
        You have to test a function that takes a positive integer number and returns the next bigger number formed  
        by the same digits:  
  
        12 ==> 21 513 ==> 531 2017 ==> 2071  
  
        If no bigger number can be composed using those digits, return -1
```

Module contents

1.4.2 Module contents

1.5 kyu_5 package

1.5.1 Subpackages

kyu_5.fibonacci_streaming package

Submodules

kyu_5.fibonacci_streaming.all_fibonacci_numbers module

```
kyu_5.fibonacci_streaming.all_fibonacci_numbers.all_fibonacci_numbers()  
    A utility method that generates an infinite sized, sequential IntStream (in Python generator) which contains all  
    the numbers in a fibonacci sequence. :return:
```

kyu_5.fibonacci_streaming.test_all_fibonacci_numbers module

```
class kyu_5.fibonacci_streaming.test_all_fibonacci_numbers.AllFibonacciNumbersTestCase (meth  
    Bases: unittest.case.TestCase  
  
    Testing all_fibonacci_numbers function  
  
    test_all_fibonacci_numbers()  
        Testing all_fibonacci_numbers function  
  
        You're going to provide a needy programmer a utility method that generates an infinite sized, sequential  
        IntStream (in Python generator) which contains all the numbers in a fibonacci sequence.  
  
        A fibonacci sequence starts with two 1s. Every element afterwards is the sum of the two previous elements.  
        :return:
```

Module contents

kyu_5.count_ip_addresses package

Submodules

kyu_5.count_ip_addresses.ips_between module

kyu_5.count_ip_addresses.ips_between.**calc_ip_range**(*ip, ip_id, ips_range*) → None

kyu_5.count_ip_addresses.ips_between.**calc_result**(*ips_range*)

kyu_5.count_ip_addresses.ips_between.**ips_between**(*start: str, end: str*) → int

A function that receives two IPv4 addresses, and returns the number of addresses between them (including the first one, excluding the last one).

All inputs will be valid IPv4 addresses in the form of strings. The last address will always be greater than the first one. :param start: :param end: :return:

kyu_5.count_ip_addresses.test_ips_between module

class kyu_5.count_ip_addresses.test_ips_between.**IpsBetweenTestCase**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Testing ips_between function

pytestmark = [Mark(name='skip', args=(), kwargs={'reason': 'The solution is not ready

test_ips_between()

Testing ips_between function

Testing a function that receives two IPv4 addresses, and returns the number of addresses between them (including the first one, excluding the last one).

All inputs will be valid IPv4 addresses in the form of strings. The last address will always be greater than the first one. :return:

Module contents

kyu_5.not_very_secure package

Submodules

kyu_5.not_very_secure.alphanumeric module

kyu_5.not_very_secure.alphanumeric.**alphanumeric**(*password: str*) → bool

The string has the following conditions to be alphanumeric:

1. At least one character ("" is not valid)
2. Allowed characters are uppercase / lowercase latin letters and digits from 0 to 9
3. No whitespaces / underscore :param password: :return:

kyu_5.not_very_secure.test_alphanumeric module

```
class kyu_5.not_very_secure.test_alphanumeric.AlphanumericTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing alphanumeric function

```
test_alphanumeric ()
```

Testing alphanumeric function with various test inputs

The string has the following conditions to be alphanumeric only:

1. At least one character (“” is not valid)
2. Allowed characters are uppercase / lowercase latin letters and digits from 0 to 9
3. No whitespaces / underscore / special chars :return:

Module contents

kyu_5.simple_pig_latin package

Submodules

kyu_5.simple_pig_latin.pig_it module

```
kyu_5.simple_pig_latin.pig_it.pig_it (text: str) → str
```

Move the first letter of each word to the end of it, then add “ay” to the end of the word. Leave punctuation marks untouched. :param text: :return:

```
kyu_5.simple_pig_latin.pig_it.word_processor (word: str, result: list) → None
```

Processing a single word for the requested pattern :param word: :param result: :return:

kyu_5.simple_pig_latin.test_pig_it module

```
class kyu_5.simple_pig_latin.test_pig_it.PigItTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing pig_it function

```
test_pig_it ()
```

Testing pig_it function

The function should move the first letter of each word to the end of it, then add “ay” to the end of the word. Leave punctuation marks untouched. :return:

Module contents

kyu_5.human_readable_time package

Submodules

kyu_5.human_readable_time.make_readable module

kyu_5.human_readable_time.make_readable.**make_readable**(seconds: int) → str

Write a function, which takes a non-negative integer (seconds) as input and returns the time in a human-readable format (HH:MM:SS)

HH = hours, padded to 2 digits, range: 00 - 99 MM = minutes, padded to 2 digits, range: 00 - 59 SS = seconds, padded to 2 digits, range: 00 - 59

The maximum time never exceeds 359999 (99:59:59)

Parameters seconds –

Returns

kyu_5.human_readable_time.test_make_readable module

class kyu_5.human_readable_time.test_make_readable.**MakeReadableTestCase**(methodName='runTest')
Bases: unittest.case.TestCase

Testing make_readable function

test_make_readable()

Testing make_readable function

Write a function, which takes a non-negative integer (seconds) as input and returns the time in a human-readable format (HH:MM:SS)

HH = hours, padded to 2 digits, range: 00 - 99 MM = minutes, padded to 2 digits, range: 00 - 59 SS = seconds, padded to 2 digits, range: 00 - 59

The maximum time never exceeds 359999 (99:59:59) :return:

Module contents

kyu_5.alphabet_wars_nuclear_strike package

Submodules

kyu_5.alphabet_wars_nuclear_strike.alphabet_war module

kyu_5.alphabet_wars_nuclear_strike.alphabet_war.**alphabet_war**(battlefield: str) → str

A function that accepts battlefield string and returns letters that survived the nuclear strike. :param battlefield: :return:

kyu_5.alphabet_wars_nuclear_strike.alphabet_war.**clean_battlefield**(battlefield: str) → str

Clean the battlefield and return only survived letters :param battlefield: :return:

`kyu_5.alphabet_wars_nuclear_strike.alphabet_war.clean_unsheltered` (*battlefield: str*) → str
 Clean letters outside the shelter :param battlefield: :return:

kyu_5.alphabet_wars_nuclear_strike.test_alphabet_war module

class `kyu_5.alphabet_wars_nuclear_strike.test_alphabet_war.AlphabetWarTestCase` (*methodName='run'*)
 Bases: `unittest.case.TestCase`

Testing alphabet_war function

test_alphabet_war ()
 Testing alphabet_war function

Introduction There is a war and nobody knows - the alphabet war! The letters hide in their nuclear shelters. The nuclear strikes hit the battlefield and killed a lot of them.

Task Write a function that accepts battlefield string and returns letters that survived the nuclear strike.

1. The battlefield string consists of only small letters, #, [and].
2. The nuclear shelter is represented by square brackets []. The letters inside the square brackets represent letters inside the shelter.
3. The # means a place where nuclear strike hit the battlefield. If there is at least one # on the battlefield, all letters outside of shelter die. When there is no any # on the battlefield, all letters survive (but do not expect such scenario too often ;-P).
4. The shelters have some durability. When 2 or more # hit close to the shelter, the shelter is destroyed and all letters inside evaporate. The 'close to the shelter' means on the ground between the shelter and the next shelter (or beginning/end of battlefield). The below samples make it clear for you. :return:

Module contents

kyu_5.valid_parentheses package

Submodules

kyu_5.valid_parentheses.test_valid_parentheses module

class `kyu_5.valid_parentheses.test_valid_parentheses.ValidParenthesesTestCase` (*methodName='run'*)
 Bases: `unittest.case.TestCase`

Testing valid_parentheses function

test_valid_parentheses ()
 Test the function called that takes a string of parentheses, and determines if the order of the parentheses is valid. The function should return true if the string is valid, and false if it's invalid.

Examples

`"()" => true` `"()(())" => false` `"(" => false` `"()((()())())" => true` :return:

kyu_5.valid_parentheses.valid_parentheses module

`kyu_5.valid_parentheses.valid_parentheses.clean_up_string(string: str) → str`
Cleaning up string from invalid chars :param string: :return:

`kyu_5.valid_parentheses.valid_parentheses.valid_parentheses(string: str) → bool`
A function called that takes a string of parentheses, and determines if the order of the parentheses is valid. The function should return true if the string is valid, and false if it's invalid. :param string: :return:

Module contents

kyu_5.moving_zeros_to_the_end package

Submodules

kyu_5.moving_zeros_to_the_end.move_zeros module

`kyu_5.moving_zeros_to_the_end.move_zeros.move_zeros(array: list)`
An algorithm that takes an array and moves all of the zeros to the end, preserving the order of the other elements.
:param array: :return:

kyu_5.moving_zeros_to_the_end.test_move_zeros module

class `kyu_5.moving_zeros_to_the_end.test_move_zeros.MoveZerosTestCase` (*methodName='runTest'*)
Bases: `unittest.case.TestCase`
Testing move_zeros function
test_move_zeros()
Test an algorithm that takes an array and moves all of the zeros to the end, preserving the order of the other elements. :return:

Module contents

kyu_5.directions_reduction package

Submodules

kyu_5.directions_reduction.directions_reduction module

`kyu_5.directions_reduction.directions_reduction.dirReduc(arr: list) → list`
A function dirReduc which will take an array of strings and returns an array of strings with the needless directions removed (W<->E or S<->N side by side).
The Haskell version takes a list of directions with data Direction = North | East | West | South.
The Clojure version returns nil when the path is reduced to nothing.
The Rust version takes a slice of enum Direction {NORTH, SOUTH, EAST, WEST}. :param arr: :return:

kyu_5.directions_reduction.test_directions_reduction module

class kyu_5.directions_reduction.test_directions_reduction.DirectionsReductionTestCase (method)
 Bases: unittest.case.TestCase

Testing dirReduc function

test_directions_reduction()

Test a function dirReduc which will take an array of strings and returns an array of strings with the needless directions removed (W<->E or S<->N side by side).

The Haskell version takes a list of directions with data Direction = North | East | West | South.

The Clojure version returns nil when the path is reduced to nothing.

The Rust version takes a slice of enum Direction {NORTH, SOUTH, EAST, WEST}. :return:

Module contents**kyu_5.did_i_finish_my_sudoku package****Submodules****kyu_5.did_i_finish_my_sudoku.is_sudoku_done module**

kyu_5.did_i_finish_my_sudoku.is_sudoku_done.**done_or_not** (board: list) → str
 return 'Finished!' or return 'Try again!' :param board: :return:

kyu_5.did_i_finish_my_sudoku.sudoku_by_column module

kyu_5.did_i_finish_my_sudoku.sudoku_by_column.**assert_sudoku_by_column** (board:
 list)
 →
 bool

kyu_5.did_i_finish_my_sudoku.sudoku_by_regions module

kyu_5.did_i_finish_my_sudoku.sudoku_by_regions.**assert_sudoku_by_region** (board:
 list)
 →
 bool

Assert Sudoku by region

Parameters **board** – Sudoku list

Returns boolean value (is Sudoku done or not)

kyu_5.did_i_finish_my_sudoku.sudoku_by_row module

kyu_5.did_i_finish_my_sudoku.sudoku_by_row.**assert_sudoku_by_row**(board: list) →
bool

kyu_5.did_i_finish_my_sudoku.testDid_i_finish_sudoku module

```
class kyu_5.did_i_finish_my_sudoku.testDid_i_finish_sudoku.DidIFinishedSudokuTestCase (meth
    Bases: unittest.case.TestCase
    Testing done_or_not function
    test_done_or_not ()
        Testing done_or_not function
        Testing a function done_or_not/DoneOrNot passing a board (list[list_lines]) as parameter. If the board is
        valid return 'Finished!', otherwise return 'Try again!' :return:
```

Module contents

kyu_5.where_my_anagrams_at package

Submodules

kyu_5.where_my_anagrams_at.anagrams module

kyu_5.where_my_anagrams_at.anagrams.**anagrams**(word, words)
A function that will find all the anagrams of a word from a list. You will be given two inputs a word and an array with words. You should return an array of all the anagrams or an empty array if there are none.

kyu_5.where_my_anagrams_at.test_anagrams module

```
class kyu_5.where_my_anagrams_at.test_anagrams.AnagramsTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing anagrams function
    test_anagrams ()
        Test a function that will find all the anagrams of a word from a list. You will be given two inputs a word
        and an array with words. You should return an array of all the anagrams or an empty array if there are
        none.
        For example:
        anagrams('abba', ['aabb', 'abcd', 'bbaa', 'dada']) => ['aabb', 'bbaa'] anagrams('racer', ['crazer', 'carer',
        'racar', 'caers', 'racer']) => ['carer', 'racer'] anagrams('laser', ['lazing', 'lazy', 'lacer']) => [] :return:
```

Module contents

kyu_5.master_your_primes_sieve_with_memoization package

Submodules

kyu_5.master_your_primes_sieve_with_memoization.primes module

`kyu_5.master_your_primes_sieve_with_memoization.primes.is_prime(n)`

A function that checks if a given number *n* is a prime looping through it and, possibly, expanding the array/list of known primes only if/when necessary (ie: as soon as you check for a potential prime which is greater than a given threshold for each *n*, stop). :param *n*: :return:

kyu_5.master_your_primes_sieve_with_memoization.test_primes module

class `kyu_5.master_your_primes_sieve_with_memoization.test_primes.PrimesTestCase` (*methodName=*

Bases: `unittest.case.TestCase`

Testing `is_prime` function

test_primes()

Testing a function that checks if a given number *n* is a prime looping through it and, possibly, expanding the array/list of known primes only if/when necessary (ie: as soon as you check for a potential prime which is greater than a given threshold for each *n*, stop).

Returns

Module contents

kyu_5.number_of_trailing_zeros_of_n package

Submodules

kyu_5.number_of_trailing_zeros_of_n.test_zeros module

class `kyu_5.number_of_trailing_zeros_of_n.test_zeros.ZerosTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing zeros function

test_zeros()

Testing 'zeros' program that should calculate the number of trailing zeros in a factorial of a given number.
:return:

kyu_5.number_of_trailing_zeros_of_n.zeros module

`kyu_5.number_of_trailing_zeros_of_n.zeros.zeros(n)`

A program that will calculate the number of trailing zeros in a factorial of a given number.

$N! = 1 * 2 * 3 * \dots * N$

For more info, see: <http://mathworld.wolfram.com/Factorial.html>

A simple way is to calculate $\text{floor}(n/5)$. For example, $7!$ has one 5, $10!$ has two 5s. It is done yet, there is one more thing to consider. Numbers like 25, 125, etc have more than one 5.

For example if we consider $28!$, we get one extra 5 and number of 0s become 6. Handling this is simple, first divide n by 5 and remove all single 5s, then divide by 25 to remove extra 5s and so on.

Following is the summarized formula for counting trailing 0s.

Trailing 0s in $n!$ = Count of 5s in prime factors of $n!$ = $\text{floor}(n/5) + \text{floor}(n/25) + \text{floor}(n/125) + \dots$

Parameters n –

Returns

Module contents

kyu_5.flatten package

Submodules

kyu_5.flatten.flatten module

`kyu_5.flatten.flatten.flatten(*args)`

The method takes in any number of arguments and flattens them into a single array. If any of the arguments passed in are an array then the individual objects within the array will be flattened so that they exist at the same level as the other arguments. Any nested arrays, no matter how deep, should be flattened into the single array result. :return:

`kyu_5.flatten.flatten.unpack(data, collection: list)`

Helper method. Unpack data until its not list or a tuple. :param data: :param collection: :return:

kyu_5.flatten.test_flatten module

`class kyu_5.flatten.test_flatten.FlattenTestCase (methodName='runTest')`

Bases: `unittest.case.TestCase`

Testing flatten function

`test_flatten()`

For this exercise you will create a global flatten method. The method takes in any number of arguments and flattens them into a single array. If any of the arguments passed in are an array then the individual objects within the array will be flattened so that they exist at the same level as the other arguments. Any nested arrays, no matter how deep, should be flattened into the single array result.

The following are examples of how this function would be used and what the expected results would be:

`flatten(1, [2, 3], 4, 5, [6, [7]])` # returns `[1, 2, 3, 4, 5, 6, 7]` `flatten('a', ['b', 2], 3, None, [[4], ['c']])` # returns `['a', 'b', 2, 3, None, 4, 'c']` :return:

Module contents

kyu_5.first_non_repeating_character package

Submodules

kyu_5.first_non_repeating_character.first_non_repeating_letter module

kyu_5.first_non_repeating_character.first_non_repeating_letter.**first_non_repeating_letter** (s)

A function named first_non_repeating_letter that takes a string input, and returns the first character that is not repeated anywhere in the string. :param string: :return:

kyu_5.first_non_repeating_character.test_first_non_repeating_letter module

class kyu_5.first_non_repeating_character.test_first_non_repeating_letter.**FirstNonRepeatingLetter**

Bases: unittest.case.TestCase

Testing first_non_repeating_letter function

test_first_non_repeating_letter ()

Testing a function named first_non_repeating_letter that takes a string input, and returns the first character that is not repeated anywhere in the string.

For example, if given the input 'stress', the function should return 't', since the letter t only occurs once in the string, and occurs first in the string.

As an added challenge, upper- and lowercase letters are considered the same character, but the function should return the correct case for the initial letter. For example, the input 'sTreSS' should return 'T'.

If a string contains all repeating characters, it should return an empty string ("") or None – see sample tests. :return:

Module contents

kyu_5.sports_league_table_ranking package

Submodules

kyu_5.sports_league_table_ranking.compute_ranks module

kyu_5.sports_league_table_ranking.compute_ranks.**calc_for_against** (teams, team, team_1, team_2) → None

Collect "For:Against" data

Parameters

- **teams** –
- **team** –

- **team_1** –
- **team_2** –

Returns

`kyu_5.sports_league_table_ranking.compute_ranks.calc_gd(teams)` → None

Calculates “GD”

Parameters teams –

Returns

`kyu_5.sports_league_table_ranking.compute_ranks.calc_rank(teams: dict)` → None

Calculates Rank

First you sort the teams by their points. If two or more teams reached the same number of points, the second criteria comes into play and so on. Finally, if all criteria are the same, the teams share a place.

Parameters teams –

Returns

`kyu_5.sports_league_table_ranking.compute_ranks.calc_team_points(team, teams, score_a, score_b)` → None

Calculates team points

Parameters

- **team** –
- **teams** –
- **score_a** –
- **score_b** –

Returns

`kyu_5.sports_league_table_ranking.compute_ranks.calc_teams_score(team_a, team_b, teams, team, number)` → None

Calculate following: For : Against Points

Set default values for team as well

Parameters

- **team_a** –
- **team_b** –
- **teams** –
- **team** –
- **number** –

Returns


```
kyu_5.sports_league_table_ranking.compute_ranks.compute_ranks (number:      int,
                                                                games:  list) →
                                                                list
```

organize a sports league in a round-robin-system. Each team meets all other teams. In your league a win gives a team 2 points, a draw gives both teams 1 point. After some games you have to compute the order of the teams in your league. You use the following criteria to arrange the teams:

Points Scoring differential (the difference between goals scored and those conceded)

Goals scored First you sort the teams by their points. If two or more teams reached the same number of points, the second criteria comes into play and so on. Finally, if all criteria are the same, the teams share a place.

Parameters

- **number** –
- **games** –

Returns

```
kyu_5.sports_league_table_ranking.compute_ranks.process_not_played_games (teams:
                                                                           dict,
                                                                           num-
                                                                           ber:
                                                                           int)
                                                                           →
                                                                           None
```

Set default values for teams who did not play a single game :param teams: :param number: :return:

```
kyu_5.sports_league_table_ranking.compute_ranks.test_if_team_registered (team,
                                                                           teams,
                                                                           num-
                                                                           ber)
                                                                           →
                                                                           None
```

Check if team data was processed. Set default values otherwise.

Parameters

- **team** –
- **teams** –
- **number** –

Returns

kyu_5.sports_league_table_ranking.test_compute_ranks module

```
class kyu_5.sports_league_table_ranking.test_compute_ranks.ComputeRanksTestCase (methodName='
Bases: unittest.case.TestCase
```

test_something()

Test the function that organizes a sports league in a round-robin-system. Each team meets all other teams. In your league a win gives a team 2 points, a draw gives both teams 1 point. After some games you have to compute the order of the teams in your league. You use the following criteria to arrange the teams:

- Points
- Scoring differential (the difference between goals scored and those conceded)
- Goals scored

Returns

Module contents

kyu_5.find_the_safest_places_in_town package

Submodules

kyu_5.find_the_safest_places_in_town.advice module

`kyu_5.find_the_safest_places_in_town.advice.advice` (*agents: set, n: int*) → list

The function should return a list of coordinates that are the furthest away (by Manhattan distance) from all agents.

Edge cases:

- If there is an agent on every grid cell, there is no safe space, so return an empty list.
- If there are no agents, then every cell is a safe spaces, so return all coordinates.
- if n is 0, return an empty list.
- If agent coordinates are outside of the map, they are simply not considered.
- There are no duplicate agents on the same square.

Parameters

- **agents** – is an array of agent coordinates
- **n** – defines the size of the city that Bassi needs to hide in,

in other words the side length of the square grid :return:

`kyu_5.find_the_safest_places_in_town.advice.agents_cleanup` (*agents, n*) → set

Remove all agents that are outside of the city boundaries. If agent coordinates are outside of the map, they are simply not considered.

Parameters

- **agents** – is an array of agent coordinates
- **n** – defines the size of the city that Bassi needs to hide in, in other words the side length of the square grid

Returns

`kyu_5.find_the_safest_places_in_town.advice.city_map_processing` (*city: set,*
agents: set)
→ None

Parameters

- **city** – the full city map (set)
- **agents** – is an set of agent coordinates.

Returns

`kyu_5.find_the_safest_places_in_town.advice.create_city_map` (*n: int*) → set

Generate city map with coordinates :param n: defines the size of the city that Bassi needs to hide in,

in other words the side length of the square grid

Returns

`kyu_5.find_the_safest_places_in_town.cell` module

`kyu_5.find_the_safest_places_in_town.manhattan_distance` module

`kyu_5.find_the_safest_places_in_town.print_agents` module

`kyu_5.find_the_safest_places_in_town.print_agents.print_map`(*agents: list, n: int,*
expected: list)

Use for debug purposes only. Prints city map with agents (*) and expected results (longest distance as +) on it.

Parameters

- **agents** – is an array of agent coordinates
- **n** – defines the size of the city that Bassi needs to hide in, in other words the side length of the square grid
- **expected** – expected results

Returns

`kyu_5.find_the_safest_places_in_town.test_advice` module

Testing advice and all related help functions

class `kyu_5.find_the_safest_places_in_town.test_advice.FirstAdviceTestCase` (*methodName='runTest'*
Bases: `unittest.case.TestCase`)

Testing advice and all related help functions

test_agents_cleanup()

Testing a function named agents_cleanup where:

- *agents*: is an array of agent coordinates
- *n*: defines the size of the city that Bassi needs to hide in, in other words the side length of the square grid.

The function should remove all agents that are outside of the city boundaries. :return:

test_create_city_map()

Testing a function named create_city_map where:

- *n* defines the size of the city that Bassi needs to hide in, in other words the side length of the square grid.

The function should generate city map with coordinates. :return:

test_first_non_repeating_letter()

Testing a function named advice(agents, n) where:

- *agents* is an array of agent coordinates.
- *n* defines the size of the city that Bassi needs to hide in, in other words the side length of the square grid.

The function should return a list of coordinates that are the furthest away (by Manhattan distance) from all agents. :return:

Module contents

kyu_5.extract_the_domain_name_from_url package

Submodules

kyu_5.extract_the_domain_name_from_url.extract_domain_from_url module

Extract the domain name from a URL

```
kyu_5.extract_the_domain_name_from_url.extract_domain_from_url.domain_name(url:  
                                                                    str)  
                                                                    →  
                                                                    str
```

Parses out just the domain name and returns it as a string.

Parameters `url` – URL as a string

Returns domain name as a string

kyu_5.extract_the_domain_name_from_url.test_domain_name module

Assert that 'domain_name' function returns domain name from given URL string.

```
class kyu_5.extract_the_domain_name_from_url.test_domain_name.DomainNameTestCase(methodName=  
    Bases: unittest.case.TestCase
```

Testing domain_name function

test_domain_name()

Assert that 'domain_name' function returns domain name from given URL string.

Returns

Module contents

kyu_5.the_hashtag_generator package

Submodules

kyu_5.the_hashtag_generator.hashtag_generator module

```
kyu_5.the_hashtag_generator.hashtag_generator.generate_hashtag(s: str)  
    The Hashtag Generator.
```

1. It must start with a hashtag (#).
2. All words must have their first letter capitalized.
3. If the final result is longer than 140 chars it must return false.
4. If the input or the result is an empty string it must return false.

Parameters **s** –

Returns

kyu_5.the_hashtag_generator.test_generate_hashtag module

Testing 'generate_hashtag' function

```
class kyu_5.the_hashtag_generator.test_generate_hashtag.GenerateHashtagTestCase (methodName='')
    Bases: unittest.case.TestCase

    test_generate_hashtag ()
        Testing 'generate_hashtag' function
```

Module contents

kyu_5.sum_of_pairs package

Submodules

kyu_5.sum_of_pairs.sum_pairs module

kyu_5.sum_of_pairs.sum_pairs.**simplify** (*ints: list*) → list

In order to speed up the process we should simplify the input list by reducing duplicate values, see sample below:

```
[1,4,5,1,1,1,1,1,4,7,8] >>> [1,4,5,1,4,7,8]
```

Parameters **ints** – a list of integers

Returns simplified list of integers

kyu_5.sum_of_pairs.sum_pairs.**sum_pairs** (*ints: list, s: int*)

Given a list of integers and a single sum value, returns the first two values (parse from the left please) in order of appearance that add up to form the sum.

Parameters

- **ints** – a list of integers
- **s** – a single sum value

Returns the first two values = s

kyu_5.sum_of_pairs.test_sum_pairs module

Testing 'sum_pairs' function

```
class kyu_5.sum_of_pairs.test_sum_pairs.SumPairsTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing 'sum_pairs' function

    test_sum_pairs ()
        Testing 'sum_pairs' function
```

Given a list of integers and a single sum value, the function should return the first two values (parse from the left please) in order of appearance that add up to form the sum.

Module contents

kyu_5.tic_tac_toe_checker package

Submodules

kyu_5.tic_tac_toe_checker.checker module

`kyu_5.tic_tac_toe_checker.checker.check_cols(board)`

Check board by column

Parameters `board` – list

Returns 1, 2, or None

`kyu_5.tic_tac_toe_checker.checker.check_diagonals(board)`

Check board by diagonal

Parameters `board` – list

Returns 1, 2, or None

`kyu_5.tic_tac_toe_checker.checker.check_rows(board: list)`

Check board by row

Parameters `board` – list

Returns 1, 2, or None

`kyu_5.tic_tac_toe_checker.checker.is_solved(board)`

Checks whether the board's current state is solved: -1 if the board is not yet finished (there are empty spots), 1 if "X" won, 2 if "O" won, 0 if it's a cat's game (i.e. a draw).

Parameters `board` – list

Returns -1, 0, 1, or 2

kyu_5.tic_tac_toe_checker.test_checker module

Testing `is_solved` function

class `kyu_5.tic_tac_toe_checker.test_checker.IsSolvedTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing `is_solved` function

test_is_solved()

Testing `is_solved` function

The function should return whether the board's current state is solved.

We want our function to return:

-1 if the board is not yet finished (there are empty spots), 1 if "X" won, 2 if "O" won, 0 if it's a cat's game (i.e. a draw).

Module contents

kyu_5.string_incrementer package

Submodules

kyu_5.string_incrementer.string_incrementer module

`kyu_5.string_incrementer.string_incrementer.get_first_digit_index(string: str)`

Find index of first non digit char from right to left

Parameters `string` – input string

Returns index of first non digit char or None

`kyu_5.string_incrementer.string_incrementer.increment_string(string: str) → str`

A function which increments a string, to create a new string: 1. If the string already ends with a number, the number should be incremented by 1. 2. If the string does not end with a number. the number 1 should be appended to the new string.

Parameters `string` – input string

Returns output string with incremented number

kyu_5.string_incrementer.test_increment_string module

class `kyu_5.string_incrementer.test_increment_string.StringIncrementerTestCase` (*methodName='run'*)

Bases: `unittest.case.TestCase`

Testing increment_string function

test_increment_string()

Testing a function named increment_string

Returns

Module contents

kyu_5.integers_recreation_one package

Submodules

kyu_5.integers_recreation_one.solution module

`kyu_5.integers_recreation_one.solution.digital_root(num: str) → int`

The digital root or digital sum of a non-negative integer is the single-digit value obtained by an iterative process of summing digits, on each iteration using the result from the previous iteration to compute the digit sum. The process continues until a single-digit number is reached.

Parameters `num` – a digit/number/integer

Returns digital root

`kyu_5.integers_recreation_one.solution.divisor_generator(n: int)`

The best way to get all the divisors of a number.

Parameters *n* – integers

Returns all dividers of *n*

`kyu_5.integers_recreation_one.solution.is_perfect_square(n: str) → bool`

Check if a number is a perfect square. (number made by squaring a whole number: $4 * \$ = 16$).

Parameters *n* – integer

Returns bool

`kyu_5.integers_recreation_one.solution.list_squared(m: int, n: int) → list`

Given two integers *m*, *n* ($1 \leq m \leq n$) we want to find all integers between *m* and *n* whose sum of squared divisors is itself a square.

Parameters

- *m* – start
- *n* – end

Returns list of integers between *m* and *n* whose sum of squared divisors is itself a square

kyu_5.integers_recreation_one.test_list_squared module

`class kyu_5.integers_recreation_one.test_list_squared.ListSquaredTestCase (methodName='runTest')`

Bases: `unittest.case.TestCase`

Integers: Recreation One

Divisors of 42 are : 1, 2, 3, 6, 7, 14, 21, 42. These divisors squared are: 1, 4, 9, 36, 49, 196, 441, 1764. The sum of the squared divisors is 2500 which is $50 * 50$, a square!

Given two integers *m*, *n* ($1 \leq m \leq n$) we want to find all integers between *m* and *n* whose sum of squared divisors is itself a square. 42 is such a number.

The result should be an array of arrays or of tuples (in C an array of Pair) or a string, each sub-array having two elements, first the number whose squared divisors is a square and then the sum of the squared divisors.

`test_flatten()`

Testing `list_squared` function

Returns

Module contents

1.5.2 Module contents

1.6 kyu_6 package

1.6.1 Subpackages

kyu_6.find_the_odd_int package

Submodules

kyu_6.find_the_odd_int.find_the_odd_int module

Find the odd int

`kyu_6.find_the_odd_int.find_the_odd_int.find_it(seq: List[int]) → int`
Given an array, find the int that appears an odd number of times. :param seq: :return:

kyu_6.find_the_odd_int.test_find_the_odd_int module

```
class kyu_6.find_the_odd_int.test_find_the_odd_int.FindTheOddIntTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing find_it function
    test_something()
        Sample testing. Expected result is 5 :return:
```

Module contents

kyu_6.first_character_that_repeats package

Submodules

kyu_6.first_character_that_repeats.first_character_that_repeats module

`kyu_6.first_character_that_repeats.first_character_that_repeats.first_dup(s)`
Find the first character that repeats in a String and return that character. :param s: :return:

kyu_6.first_character_that_repeats.test_first_character_that_repeats module

```
class kyu_6.first_character_that_repeats.test_first_character_that_repeats.FirstDupTestCase
    Bases: unittest.case.TestCase
    Testing first_dup function
    Find the first character that repeats in a String and return that character.
    test_first_alpha_only()
        Test string with alphabet chars only :return:
    test_first_dup_mixed()
        Test string with mixed type of chars :return:
    test_first_dup_none()
        Test string with no duplicate chars :return:
    test_first_no_alpha()
        Test string with no alphabet chars :return:
    test_first_space()
        Repeating char is a space :return:
```

Module contents

kyu_6.pyramid_array package

Submodules

kyu_6.pyramid_array.pyramid_array module

`kyu_6.pyramid_array.pyramid_array.pyramid(n)`

Write a function that when given a number ≥ 0 , returns an Array of ascending length subarrays.

Note: the subarrays should be filled with 1s :param n: :return:

kyu_6.pyramid_array.test_pyramid_array module

class `kyu_6.pyramid_array.test_pyramid_array.PyramidTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing 'pyramid' function

test_pyramid()

The 'pyramid' function should return an Array of ascending length subarrays.

Note: the subarrays should be filled with 1s. :return:

Module contents

kyu_6.longest_repetition package

Submodules

kyu_6.longest_repetition.longest_repetition module

`kyu_6.longest_repetition.longest_repetition.longest_repetition(chars: str) → Tuple`

For a given string s find the character c (or C) with longest consecutive repetition and return: (c, l)

where l (or L) is the length of the repetition. If there are two or more characters with the same l return the first.

For empty string return: ('', 0) :param chars: :return:

kyu_6.longest_repetition.test_longest_repetition module

class `kyu_6.longest_repetition.test_longest_repetition.LongestRepetitionTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing longest_repetition function

test_longest_repetition()

For a given string s find the character c (or C) with longest consecutive repetition and return: (c, l) where l (or L) is the length of the repetition.

For empty string return: ('', 0) :return:

Module contents

kyu_6.numericals_of_string package

Submodules

kyu_6.numericals_of_string.numericals module

`kyu_6.numericals_of_string.numericals.numericals(s)`

For each symbol in the string if it's the first character occurrence, replace it with a '1', else replace it with the amount of times you've already seen it. :param s: :return:

kyu_6.numericals_of_string.test_numericals module

class `kyu_6.numericals_of_string.test_numericals.NumericalsTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing 'numericals' function

test_numericals()

Testing 'numericals' function :return:

Module contents

kyu_6.character_frequency package

Submodules

kyu_6.character_frequency.character_frequency module

Character frequency

`kyu_6.character_frequency.character_frequency.letter_frequency(text: str) → list`

return a list of tuples sorted by frequency with the most frequent letter first. Any letters with the same frequency are ordered alphabetically :param text: :return:

`kyu_6.character_frequency.character_frequency.sort_list(results) → list`

kyu_6.character_frequency.test_character_frequency module

class `kyu_6.character_frequency.test_character_frequency.LetterFrequencyTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing letter_frequency function

test_letter_frequency_all_caps()

Testing letter_frequency function where all chars are in upper case :return:

test_letter_frequency_all_lower()

Testing letter_frequency function where all chars are in lower case :return:

test_letter_frequency_mixed()

Testing letter_frequency function where all chars are in mixed case :return:

Module contents

kyu_6.string_subpattern_recognition_1 package

Submodules

kyu_6.string_subpattern_recognition_1.has_subpattern module

```
kyu_6.string_subpattern_recognition_1.has_subpattern.has_subpattern(string:  
                                                                    str) →  
                                                                    bool
```

String subpattern recognition I

In this kata you need to build a function to return either true/True or false/False if a string can be seen as the repetition of a simpler/shorter subpattern or not.

Strings will never be empty and can be composed of any character (just consider upper- and lowercase letters as different entities) and can be pretty long (keep an eye on performances!).

Parameters **string** –

Returns

kyu_6.string_subpattern_recognition_1.test_has_subpattern module

```
class kyu_6.string_subpattern_recognition_1.test_has_subpattern.HasSubpatternTestCase (method-wrapper: HasSubpatternTestCase)  
    Bases: unittest.case.TestCase
```

String subpattern recognition I Testing 'has_subpattern' function

```
test_has_subpattern()  
    String subpattern recognition I
```

Verify that 'has_subpattern' function to returns either true/True or false/False if a string can be seen as the repetition of a simpler/shorter subpattern or not. :return:

Module contents

kyu_6.string_subpattern_recognition_2 package

Submodules

kyu_6.string_subpattern_recognition_2.has_subpattern module

```
kyu_6.string_subpattern_recognition_2.has_subpattern.has_subpattern(string:  
                                                                    str) →  
                                                                    bool
```

String subpattern recognition II

if a subpattern has been used, it will be repeated at least twice, meaning the subpattern has to be shorter than the original string;

the strings you will be given might or might not be created repeating a given subpattern, then shuffling the result.

Parameters **string** –

kyu_6.string_subpattern_recognition_2.test_has_subpattern module

Bases: `unittest.case.TestCase`

2. the strings you will be given might or might not be created repeating a given subpattern, then shuffling the result. :return:

kyu_6.string_subpattern_recognition_3.has_subpattern module

Returns

Return a subpattern with sorted characters, otherwise return the base string with sorted characters (you might consider this case as an edge case, with the subpattern being repeated only once and thus equalling the original input string). :return:

Module contents

kyu_6.permute_a_palindrome package

Submodules

kyu_6.permute_a_palindrome.permute_a_palindrome module

```
kyu_6.permute_a_palindrome.permute_a_palindrome.permute_a_palindrome(string:  
                                                                    str) →  
                                                                    bool  
    A function that check whether the permutation of an input string is a palindrome. :param string: :return:
```

kyu_6.permute_a_palindrome.test_permute_a_palindrome module

```
class kyu_6.permute_a_palindrome.test_permute_a_palindrome.PermutePalindromeTestCase(methodl  
    Bases: unittest.case.TestCase  
    Testing permute_a_palindrome function  
    test_permute_a_palindrome_empty_string()  
    test_permute_a_palindrome_negative()  
    test_permute_a_palindrome_positive()  
    Testing permute_a_palindrome function :return:
```

Module contents

kyu_6.count_letters_in_string package

Submodules

kyu_6.count_letters_in_string.count_letters_in_string module

```
kyu_6.count_letters_in_string.count_letters_in_string.letter_count(s: str) →  
                                                                    dict  
    Count lowercase letters in a given string and return the letter count in a hash with 'letter' as key and count as  
    'value'. :param s: :return:
```

kyu_6.count_letters_in_string.test_count_letters_in_string module

```
class kyu_6.count_letters_in_string.test_count_letters_in_string.CountLettersInStringTestC  
    Bases: unittest.case.TestCase  
    Testing 'letter_count' function  
    test_count_letters_in_string()  
    Testing 'letter_count' function :return:
```

Module contents

kyu_6.unique_in_order package

Submodules

kyu_6.unique_in_order.test_unique_in_order module

```
class kyu_6.unique_in_order.test_unique_in_order.UniqueInOrderTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing the 'unique_in_order' function

    test_unique_in_order()
        Testing the 'unique_in_order' function with various test data :return:
```

kyu_6.unique_in_order.unique_in_order module

```
kyu_6.unique_in_order.unique_in_order.unique_in_order (iterable: Iterable) → list
    Takes as argument a sequence and returns a list of items without any elements with the same value next to each
    other and preserving the original order of elements. :param iterable: :return:
```

Module contents

kyu_6.duplicate_encoder package

Submodules

kyu_6.duplicate_encoder.duplicate_encode module

```
kyu_6.duplicate_encoder.duplicate_encode.duplicate_encode (word: str) → str
    Converts a string to a new string where each character in the new string is "(" if that character appears only once
    in the original string, or ")" if that character appears more than once in the original string.

    Ignore capitalization when determining if a character is a duplicate. :param word: :return:
```

kyu_6.duplicate_encoder.test_duplicate_encode module

```
class kyu_6.duplicate_encoder.test_duplicate_encode.DuplicateEncodeTestCase (methodName='runTe
    Bases: unittest.case.TestCase
    Testing duplicate_encode function

    test_duplicate_encode()
        Testing duplicate_encode function with various test inputs :return:
```

Module contents

kyu_6.vasya_clerk package

Submodules

kyu_6.vasya_clerk.test_tickets module

class kyu_6.vasya_clerk.test_tickets.**TicketsTestCase** (*methodName='runTest'*)

Bases: unittest.case.TestCase

Testing tickets function

test_tickets ()

Testing tickets function with various test inputs.

The new “Avengers” movie has just been released! There are a lot of people at the cinema box office standing in a huge line. Each of them has a single 100, 50 or 25 dollar bill. An “Avengers” ticket costs 25 dollars.

Vasya is currently working as a clerk. He wants to sell a ticket to every single person in this line.

Can Vasya sell a ticket to every person and give change if he initially has no money and sells the tickets strictly in the order people queue?

The function should return YES, if Vasya can sell a ticket to every person and give change with the bills he has at hand at that moment. Otherwise return NO. :return:

kyu_6.vasya_clerk.tickets module

kyu_6.vasya_clerk.tickets.**tickets** (*people: list*) → str

Return YES, if Vasya can sell a ticket to every person and give change with the bills he has at hand at that moment. Otherwise return NO. :param people: :return:

Module contents

kyu_6.string_transformer package

Submodules

kyu_6.string_transformer.string_transformer module

kyu_6.string_transformer.string_transformer.**string_transformer** (*s: str*) → str

Given a string, return a new string that has transformed based on the input:

1. Change case of every character, ie. lower case to upper case, upper case to lower case. 2. Reverse the order of words from the input.

Note: You will have to handle multiple spaces, and leading/trailing spaces.

You may assume the input only contain English alphabet and spaces. :param s: :return:

kyu_6.string_transformer.test_string_transformer module

```
class kyu_6.string_transformer.test_string_transformer.StringTransformerTestCase (methodName=  
    Bases: unittest.case.TestCase
```

Testing string_transformer function

```
test_string_transformer()
```

Testing string_transformer function with multiple test data.

Given a string, return a new string that has transformed based on the input:

1. Change case of every character, ie. lower case to upper case, upper case to lower case.
2. Reverse the order of words from the input.

Returns

Module contents

kyu_6.multiples_of_3_or_5 package

Submodules

kyu_6.multiples_of_3_or_5.solution module

```
kyu_6.multiples_of_3_or_5.solution.solution (number: int) → int
```

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Finish the solution so that it returns the sum of all the multiples of 3 or 5 below the number passed in. :param number: :return:

kyu_6.multiples_of_3_or_5.test_solution module

```
class kyu_6.multiples_of_3_or_5.test_solution.SolutionTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing solution function

```
test_solution()
```

Testing solution function

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Finish the solution so that it returns the sum of all the multiples of 3 or 5 below the number passed in.

Note: If the number is a multiple of both 3 and 5, only count it once. :return:

Module contents

kyu_6.sum_of_digits_digital_root package

Submodules

kyu_6.sum_of_digits_digital_root.digital_root module

`kyu_6.sum_of_digits_digital_root.digital_root.digital_root (n: int) → int`

In this kata, you must create a digital root function.

A digital root is the recursive sum of all the digits in a number. Given *n*, take the sum of the digits of *n*. If that value has more than one digit, continue reducing in this way until a single-digit number is produced. This is only applicable to the natural numbers. :param *n*: :return:

kyu_6.sum_of_digits_digital_root.test_digital_root module

```
class kyu_6.sum_of_digits_digital_root.test_digital_root.DigitalRootTestCase (methodName='runTest')
```

Bases: `unittest.case.TestCase`

Testing `digital_root` function

```
test_digital_root ()
```

In this kata, you must create a digital root function.

A digital root is the recursive sum of all the digits in a number. Given *n*, take the sum of the digits of *n*. If that value has more than one digit, continue reducing in this way until a single-digit number is produced. This is only applicable to the natural numbers. :return:

Module contents

kyu_6.binary_to_text_ascii_conversion package

Submodules

kyu_6.binary_to_text_ascii_conversion.binary_to_string module

`kyu_6.binary_to_text_ascii_conversion.binary_to_string.binary_to_string (binary: str) → str`

Write a function that takes in a binary string and returns the equivalent decoded text (the text is ASCII encoded).

Each 8 bits on the binary string represent 1 character on the ASCII table.

The input string will always be a valid binary string.

Characters can be in the range from “00000000” to “11111111” (inclusive)

Note: In the case of an empty binary string your function should return an empty string

How to convert binary string to and from ASCII text in Python: <https://kite.com/python/answers/how-to-convert-binary-string-to-and-from-ascii-text-in-python> <https://stackoverflow.com/questions/7396849/convert-binary-to-ascii-and-vice-versa>

Parameters **binary** –

Returns

kyu_6.binary_to_text_ascii_conversion.test_binary_to_string module

```
class kyu_6.binary_to_text_ascii_conversion.test_binary_to_string.SequenceTestCase (methodName)
    Bases: unittest.case.TestCase
    Testing binary_to_string function
    test_binary_to_string()
```

Module contents

kyu_6.casino_chips package

Submodules

kyu_6.casino_chips.solve module

kyu_6.casino_chips.solve.**solve** (*arr: list*) → int

You are given three piles of casino chips: white, green and black chips:

the first pile contains only white chips the second pile contains only green chips the third pile contains only black chips

Each day you take exactly two chips of different colors and head to the casino. You can chose any color, but you are not allowed to take two chips of the same color in a day.

You will be given an array representing the number of chips of each color and your task is to return the maximum number of days you can pick the chips. Each day you need to take exactly two chips.

Parameters **arr** –

Returns

kyu_6.casino_chips.test_solve module

```
class kyu_6.casino_chips.test_solve.SolveTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing solve function
    test_solve()
```

Module contents

kyu_6.pokemon_damage_calculator package

Submodules

kyu_6.pokemon_damage_calculator.calculate_damage module

```
kyu_6.pokemon_damage_calculator.calculate_damage.calculate_damage(your_type:  
                                                                    str, oppo-  
                                                                    nent_type:  
                                                                    str, attack,  
                                                                    defense) →  
                                                                    int
```

It's a Pokemon battle! Your task is to calculate the damage that a particular move would do using the following formula (not the actual one from the game):

$$\text{damage} = 50 * (\text{attack} / \text{defense}) * \text{effectiveness}$$
Parameters

- **your_type** –
- **opponent_type** –
- **attack** –
- **defense** –

Returns

```
kyu_6.pokemon_damage_calculator.calculate_damage.effectiveness(your_type: str,  
                                                                    opponent_type:  
                                                                    str) → float
```

Effectiveness:

Super effective: 2x damage Neutral: 1x damage Not very effective: 0.5x damage

To prevent this kata from being tedious, you'll only be dealing with four types: fire, water, grass, and electric. Here is the effectiveness of each match-up:

fire > grass fire < water fire = electric water < grass water < electric grass = electric

Parameters

- **your_type** –
- **opponent_type** –

Returns**kyu_6.pokemon_damage_calculator.test_calculate_damage module**

```
class kyu_6.pokemon_damage_calculator.test_calculate_damage.CalculateDamageTestCase(methodN  
    Bases: unittest.case.TestCase  
    Testing calculate_damage function: damage = 50 * (attack / defense) * effectiveness  
    test_calculate_damage()
```

Module contents

kyu_6.help_the_bookseller package

Submodules

kyu_6.help_the_bookseller.stock_list module

`kyu_6.help_the_bookseller.stock_list.stock_list` (*listOfArt: list, listOfCat: list*) → str

You will be given a stocklist (e.g. : L) and a list of categories in capital letters e.g :

`M = {"A", "B", "C", "W"}`

or `M = ["A", "B", "C", "W"]` or ...

and your task is to find all the books of L with codes belonging to each category of M and to sum their quantity according to each category.

kyu_6.help_the_bookseller.test_stock_list module

class `kyu_6.help_the_bookseller.test_stock_list.StockListTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing stock_list function

`test_stock_list()`

Module contents

kyu_6.row_of_the_odd_triangle package

Submodules

kyu_6.row_of_the_odd_triangle.odd_row module

`kyu_6.row_of_the_odd_triangle.odd_row.calc_first_number` (*n: int*) → int

Calculate first number in the row :param n: :return:

`kyu_6.row_of_the_odd_triangle.odd_row.calc_last_number` (*n: int*) → int

Calculate last number in the row :param n: :return:

`kyu_6.row_of_the_odd_triangle.odd_row.odd_row` (*n: int*) → list

Given a triangle of consecutive odd numbers finds the triangle's row knowing its index (the rows are 1-indexed).
:param n: :return:

kyu_6.row_of_the_odd_triangle.test_odd_row module

```
class kyu_6.row_of_the_odd_triangle.test_odd_row.OddRowTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing odd_row function
    test_odd_row()
```

Module contents

kyu_6.potion_class_101 package

Submodules

kyu_6.potion_class_101.potion module

```
class kyu_6.potion_class_101.potion.Potion (color: Tuple[int, int, int], volume: int)
    Bases: object
```

This is your first potion class in Hogwarts and professor gave you a homework to figure out what color potion will turn into if he'll mix it with some other potion. All potions have some color that written down as RGB color from [0, 0, 0] to [255, 255, 255]. To make task more complicated teacher will do few mixing and after will ask you for final color. Besides color you also need to figure out what volume will have potion after final mix.

property color

mix (other) → object

Based on your programming background you managed to figure that after mixing two potions colors will mix as if mix two RGB colors.

Note: Use ceiling when calculating the resulting potion's color. :param other: :return:

property volume

kyu_6.potion_class_101.test_potion module

```
class kyu_6.potion_class_101.test_potion.PotionTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    test_potion()
```

Module contents

kyu_6.disease_spread package

Submodules

kyu_6.disease_spread.epidemic module

```
kyu_6.disease_spread.epidemic.epidemic (tm: int, n: int, s0: int, i0: int, b: float, a: float) →
                                         int
```

We want to study the spread of the disease through the population of this school. The total population may be divided into three:

the infecteds (i), those who have recovered (r), and those who are still susceptible (s) to get the disease.

We will study the disease on a period of t_m days.

The interval $[0, t_m]$ will be divided in n small intervals of length $dt = t_m/n$. Initial conditions here could be : $S_0 = 999$, $I_0 = 1$, $R_0 = 0$ Whatever S_0 and I_0 , R_0 (number of recovered at time 0) is always 0.

The function epidemic will return the maximum number of infecteds as an integer (truncate to integer the result of $\max(I)$).

Parameters

- **t_m** – the disease on a period of days
- **n** – small intervals of length
- **s_0** – those who are still susceptible to get the disease (Initial conditions)
- **i_0** – the infected (Initial conditions)
- **b** – representing a number of contacts which can spread the disease
- **a** – fraction of the infected that will recover

Returns the maximum number of infected as an integer (truncate to integer the result of $\max(I)$).

kyu_6.disease_spread.epidemic_test_data module

Epidemic Test Data Class

```
class kyu_6.disease_spread.epidemic_test_data.EpidemicTestData (**kwargs)
    Bases: object
```

Epidemic Test Data Class

```
property a
    Returns a value
```

Returns

```
property b
    Returns b value
```

Returns

```
property expected
    Returns expected value
```

Returns

```
property i0
    Returns i0 value
```

Returns

```
property n
    Returns n value
```

Returns

```
property s0
    Returns s0 value
```

Returns

property tm
Returns tm value
Returns

kyu_6.disease_spread.test_epidemic module

```
class kyu_6.disease_spread.test_epidemic.EpidemicTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
    test_epidemic()
```

Module contents

kyu_6.a_rule_of_divisibility_by_13 package

Submodules

kyu_6.a_rule_of_divisibility_by_13.test_thirt module

```
class kyu_6.a_rule_of_divisibility_by_13.test_thirt.ThirtTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
    Testing 'thirt' function  
    test_thirt()
```

kyu_6.a_rule_of_divisibility_by_13.thirt module

kyu_6.a_rule_of_divisibility_by_13.thirt.**thirt** (*n: int*) → int
The function which processes this sequence of operations on an integer n (≥ 0). *thirt* will return the stationary number. :param n: :return:

Module contents

kyu_6.color_choice package

Submodules

kyu_6.color_choice.checkchoose module

kyu_6.color_choice.checkchoose.**checkchoose** (*m: int, n: int*) → int
Knowing m (number of posters to design), knowing n (total number of available colors), search x (number of colors for each poster so that each poster has a unique combination of colors and the number of combinations is exactly the same as the number of posters). :param m: :param n: :return:

kyu_6.color_choice.test_checkchoose module

```
class kyu_6.color_choice.test_checkchoose.CheckchooseTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing checkchoose function

```
test_checkchoose ()
```

In mathematics the number of x combinations you can take from a set of n elements is called the binomial coefficient of n and x , or more often n choose x . The formula to compute $m = n$ choose x is: $m = n! / (x! * (n - x)!)$ where $!$ is the factorial operator.

You are a renowned poster designer and painter. You are asked to provide 6 posters all having the same design each in 2 colors. Posters must all have a different color combination and you have the choice of 4 colors: red, blue, yellow, green. How many colors can you choose for each poster?

Module contents

kyu_6.default_list package

Submodules

kyu_6.default_list.default_list module

```
class kyu_6.default_list.default_list.DefaultList (lst: list, default_value: str)
    Bases: object
```

```
append (item) → None
```

This class must also support the regular list functions: `append`. :param item: :return:

```
extend (items: list) → None
```

This class must also support the regular list functions: `extend`. :param items: iterable :return:

```
insert (index: int, item) → None
```

This class must also support the regular list functions: `insert`. :param index: :param item: :return:

```
pop (item)
```

This class must also support the regular list functions: `pop`. :param item: :return:

```
remove (item) → None
```

This class must also support the regular list functions: `remove`. :param item: :return:

kyu_6.default_list.test_default_list module

```
class kyu_6.default_list.test_default_list.DefaultListTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing 'DefaultList' class

Your job is to create a class (or a function which returns an object) called `DefaultList`. The class will have two parameters to be given: a list, and a default value. The list will obviously be the list that corresponds to that object. The default value will be returned any time an index of the list is called in the code that would normally raise an error (i.e. $i > \text{len}(\text{list}) - 1$ or $i < -\text{len}(\text{list})$). This class must also support the regular list functions `extend`, `append`, `insert`, `remove`, and `pop`.

```
test_default_list_append()
    Testing 'DefaultList' class: append :return:
```

```
test_default_list_basic()
    Testing 'DefaultList' class: __getitem__
```

Called to implement evaluation of self[key]. For sequence types, the accepted keys should be integers and slice objects. Note that the special interpretation of negative indexes (if the class wishes to emulate a sequence type) is up to the __getitem__() method. :return:

```
test_default_list_extend()
    Testing 'DefaultList' class: extend :return:
```

```
test_default_list_insert()
    Testing 'DefaultList' class: insert :return:
```

```
test_default_list_pop()
    Testing 'DefaultList' class: pop :return:
```

```
test_default_list_remove()
    Testing 'DefaultList' class: remove :return:
```

Module contents

kyu_6.easy_diagonal package

Submodules

kyu_6.easy_diagonal.diagonal module

kyu_6.easy_diagonal.diagonal.**diagonal** (*n: int, p: int*) → int

We want to calculate the sum of the binomial coefficients on a given diagonal. The sum on diagonal 0 is 8 (we'll write it S(7, 0), 7 is the number of the line where we start, 0 is the number of the diagonal). In the same way S(7, 1) is 28, S(7, 2) is 56.

Parameters

- **n** – n is the line where we start and
- **p** – p is the number of the diagonal

Returns the sum of the binomial coefficients on a given diagonal

kyu_6.easy_diagonal.test_diagonal module

```
class kyu_6.easy_diagonal.test_diagonal.EasyDiagonalTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing easy_diagonal function

```
test_easy_diagonal()
    Testing easy_diagonal function :param self: :return:
```

Module contents

kyu_6.array_to_html_table package

Submodules

kyu_6.array_to_html_table.test_to_table module

kyu_6.array_to_html_table.to_table module

`kyu_6.array_to_html_table.to_table.to_table` (*data: list, header: bool = False, index: bool = False*) \rightarrow str

Takes three arguments: data, headers, index, and returns a string containing HTML tags representing the table.

Parameters

- **data** – a 2D array (list)
- **header** – an optional boolean value. If True, the first row of the array is considered a header, defaults to False
- **index** – an optional boolean value. If False, the first column in the table should contain 1-based indices of the corresponding row. If headers arguments is True, this column should have empty header. Defaults to False.

Returns a string containing HTML tags representing the table.

Module contents

kyu_6.rotate_the_letters_of_each_element package

Submodules

kyu_6.rotate_the_letters_of_each_element.group_cities module

`kyu_6.rotate_the_letters_of_each_element.group_cities.group_cities` (*seq: list*) \rightarrow list

A function that given a sequence of strings, groups the elements that can be obtained by rotating others, ignoring upper or lower cases.

In the event that an element appears more than once in the input sequence, only one of them will be taken into account for the result, discarding the rest.

Parameters **seq** – Sequence of strings. Valid characters for those strings are uppercase and lower-case characters from the alphabet and whitespaces.

Returns Sequence of elements. Each element is the group of inputs that can be obtained by rotating the strings.

`kyu_6.rotate_the_letters_of_each_element.group_cities.rotate` (*item: str, element: str*) \rightarrow bool

`kyu_6.rotate_the_letters_of_each_element.group_cities.sort_results` (*results: list*) \rightarrow None

Sort the groups deafeningly by size and in the case of a tie, by the first element of the group alphabetically.
:param results: :return:

kyu_6.rotate_the_letters_of_each_element.test_group_cities module

```
class kyu_6.rotate_the_letters_of_each_element.test_group_cities.GroupCitiesTestCase (methodName=)
    Bases: unittest.case.TestCase
    Testing 'group_cities' function

    test_group_cities()
        Test that a function that given a sequence of strings, groups the elements that can be obtained by rotating
        others, ignoring upper or lower cases.

        In the event that an element appears more than once in the input sequence, only one of them will be taken
        into account for the result, discarding the rest. :return:
```

Module contents

kyu_6.number_zoo_patrol package

Submodules

kyu_6.number_zoo_patrol.missing_number module

```
kyu_6.number_zoo_patrol.missing_number.find_missing_number(numbers: list) → int
    A function that takes a shuffled list of unique numbers from 1 to n with one element missing (which can be any
    number including n). Return this missing number.

    Parameters numbers – a shuffled list of unique numbers from 1 to n with one element missing

    Returns a missing number
```

kyu_6.number_zoo_patrol.test_find_missing_number module

```
class kyu_6.number_zoo_patrol.test_find_missing_number.FindMissingNumberTestCase (methodName=)
    Bases: unittest.case.TestCase
    Testing 'find_missing_number' function

    test_find_missing_number()
        Test a function that should take a shuffled list of unique numbers from 1 to n with one element missing
        (which can be any number including n). Should return this missing number.

        Returns
```

Module contents

kyu_6.your_order_please package

Submodules

kyu_6.your_order_please.order module

```
kyu_6.your_order_please.order.order(sentence: str) → str
    Sorts a given string by following rules:
```

1. Each word in the string will contain a single number. This number is the position the word should have in the result.
2. Note: Numbers can be from 1 to 9. So 1 will be the first word (not 0).
3. If the input string is empty, return an empty string. The words in the input String will only contain valid consecutive numbers.

param sentence Each word in the string will contain a single number

return sorted string

kyu_6.your_order_please.test_order module

```
class kyu_6.your_order_please.test_order.OrderTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing 'order' function

test_order ()

Your task is to verify that 'order' function sorts a given string by following rules:

1. Each word in the string will contain a single number. This number is the position the word should have in the result.
2. Note: Numbers can be from 1 to 9. So 1 will be the first word (not 0).
3. If the input string is empty, return an empty string. The words in the input String will only contain valid consecutive numbers.

Returns

Module contents

kyu_6.who_likes_it package

Submodules

kyu_6.who_likes_it.likes_function module

```
kyu_6.who_likes_it.likes_function.likes (names: list) → str
```

A function which must take in input array, containing the names of people who like an item. It must return the display text.

For 4 or more names, the number in and 2 others simply increases.

Parameters **names** – input array, containing the names of people who like an item

Returns the display text

kyu_6.who_likes_it.test_likes_function module

Testing likes function

```
class kyu_6.who_likes_it.test_likes_function.LikesTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing likes function

The function should take in input array, containing the names of people who like an item. It must return the display text. For 4 or more names, the number in and 2 others simply increases.

```
test_likes_function()
```

Module contents

kyu_6.decipher_this package

Submodules

kyu_6.decipher_this.solution module

```
kyu_6.decipher_this.solution.decipher_this (string: str) → str
```

Given a secret message that you need to decipher.

For each word:

- the second and the last letter is switched (e.g. Hello becomes Holle)
- the first letter is replaced by its character code (e.g. H becomes 72)

Note: there are no special characters used, only letters and spaces

Parameters **string** –

Returns

```
kyu_6.decipher_this.solution.last_digit_index (word: str) → int
```

kyu_6.decipher_this.test_decipher_this module

```
class kyu_6.decipher_this.test_decipher_this.DecipherThisTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing decipher_this function

```
test_decipher_this()
```

Testing decipher_this function :param self: :return:

Module contents

kyu_6.encrypt_this package

Submodules

kyu_6.encrypt_this.solution module

`kyu_6.encrypt_this.solution.encrypt_this(text: str) → str`

Encrypts each word in the message using the following rules:

- The first letter needs to be converted to its ASCII code.
- The second letter needs to be switched with the last letter

Keepin' it simple: There are no special characters in input.

Parameters `text` – a string containing space separated words

Returns secret messages which can be deciphered by the “Decipher this!” kata

kyu_6.encrypt_this.test_encrypt_this module

```
class kyu_6.encrypt_this.test_encrypt_this.EncryptThisTestCase (methodName='runTest')
```

```
    Bases: unittest.case.TestCase
```

```
    Testing encrypt_this function
```

```
    test_encrypt_this ()
```

```
        Testing encrypt_this function :param self: :return:
```

Module contents

kyu_6.format_string_of_names package

Submodules

kyu_6.format_string_of_names.solution module

`kyu_6.format_string_of_names.solution.namelist(names: list) → str`

Format a string of names like ‘Bart, Lisa & Maggie’

Parameters `names` – an array containing hashes of names

Returns a string formatted as a list of names separated by commas except for the last two names, which should be separated by an ampersand.

kyu_6.format_string_of_names.test_namelist module

```
class kyu_6.format_string_of_names.test_namelist.NamelistTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing namelist function

```
test_namelist()
    Test namelist
```

Given: an array containing hashes of names

Return: a string formatted as a list of names separated by commas except for the last two names, which should be separated by an ampersand.

Returns

Module contents

kyu_6.sort_the_odd package

Submodules

kyu_6.sort_the_odd.solution module

```
kyu_6.sort_the_odd.solution.sort_array (source_array: list) → list
    Sorting ascending odd numbers but even numbers must be on their places.
```

Zero isn't an odd number and you don't need to move it. If you have an empty array, you need to return it.

Parameters `source_array` –

Returns

kyu_6.sort_the_odd.test_sort_array module

```
class kyu_6.sort_the_odd.test_sort_array.SortArrayTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing 'sort_array' function

```
test_sort_array()
    The 'sort_array' function.
```

The task is to sort ascending odd numbers but even numbers must be on their places.

Zero isn't an odd number and you don't need to move it. If you have an empty array, you need to return it.

Returns

Module contents

kyu_6.array_diff package

Submodules

kyu_6.array_diff.solution module

`kyu_6.array_diff.solution.array_diff(a: list, b: list) → list`
Difference function, which subtracts one list from another and returns the result.

Parameters

- **a** – list a
- **b** – list b

Returns diff between a and b

kyu_6.array_diff.test_array_diff module

`class kyu_6.array_diff.test_array_diff.ArrayDiffTestCase (methodName='runTest')`
Bases: `unittest.case.TestCase`

Testing array_diff function

Your goal in this kata is to implement a difference function, which subtracts one list from another and returns the result.

It should remove all values from list a, which are present in list b: `array_diff([1,2],[1]) == [2]`

If a value is present in b, all of its occurrences must be removed from the other: `array_diff([1,2,2,2,3],[2]) == [1,3]`

`test_array_diff_function()`

Module contents

1.6.2 Module contents

1.7 kyu_7 package

1.7.1 Subpackages

kyu_7.beginner_series_sum_of_numbers package

Submodules

kyu_7.beginner_series_sum_of_numbers.sum_of_numbers module

Beginner Series #3 Sum of Numbers

`kyu_7.beginner_series_sum_of_numbers.sum_of_numbers.get_sum(a, b)`

Given two integers a and b, which can be positive or negative, find the sum of all the numbers between including them too and return it. If the two numbers are equal return a or b. :param a: :param b: :return:

`kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers` module

class `kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers.SumOfNumbersTestCase` (*methodName*)

Bases: `unittest.case.TestCase`

Testing get_sum function

test_get_sum_equal_numbers ()

a and b are equal :return:

test_get_sum_negative_numbers ()

a or b is negative :return:

test_get_sum_positive_numbers ()

a and b are positive numbers :return:

Module contents

`kyu_7.disemvowel_trolls` package

Submodules

`kyu_7.disemvowel_trolls.disemvowel_trolls` module

`kyu_7.disemvowel_trolls.disemvowel_trolls.disemvowel(string)`

A function that takes a string and return a new string with all vowels removed.

For example, the string “This website is for losers LOL!” would become “Ths wbst s fr lsrs LL!”.

Note: for this kata y isn’t considered a vowel. :param string: :return:

`kyu_7.disemvowel_trolls.test_disemvowel_trolls` module

class `kyu_7.disemvowel_trolls.test_disemvowel_trolls.DisemvowelTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing disemvowel function

test_disemvowel ()

The string “This website is for losers LOL!” should become “Ths wbst s fr lsrs LL!” :return:

Module contents

kyu_7.jaden_casing_strings package

Submodules

kyu_7.jaden_casing_strings.jaden_casing_strings module

`kyu_7.jaden_casing_strings.jaden_casing_strings.toJadenCase(string)`

Convert strings to how they would be written by Jaden Smith. The strings are actual quotes from Jaden Smith, but they are not capitalized in the same way he originally typed them.

Example:

Not Jaden-Cased: "How can mirrors be real if our eyes aren't real"

Jaden-Cased: "How Can Mirrors Be Real If Our Eyes Aren't Real"

Parameters `string` –

Returns

kyu_7.jaden_casing_strings.test_jaden_casing_strings module

class `kyu_7.jaden_casing_strings.test_jaden_casing_strings.JadenCasingStringsTestCase` (*method*

Bases: `unittest.case.TestCase`

Testing toJadenCase function

test_to_jaden_case_negative()

Simple negative test :return:

test_to_jaden_case_positive()

Simple positive test :return:

Module contents

kyu_7.remove_the_minimum package

Submodules

kyu_7.remove_the_minimum.remove_the_minimum module

`kyu_7.remove_the_minimum.remove_the_minimum.remove_smallest(numbers)`

Given an array of integers, remove the smallest value. Do not mutate the original array/list. If there are multiple elements with the same value, remove the one with a lower index. If you get an empty array/list, return an empty array/list.

Don't change the order of the elements that are left.

param `numbers`

return

kyu_7.remove_the_minimum.test_remove_the_minimum module

```
class kyu_7.remove_the_minimum.test_remove_the_minimum.RemoveSmallestTestCase (methodName='run'  
    Bases: unittest.case.TestCase
```

Testing remove_smallest function

```
static random_list ()
```

Helper function :return:

```
test_remove_smallest ()
```

Test lists with multiple digits :return:

```
test_remove_smallest_empty_list ()
```

Test with empty list :return:

```
test_remove_smallest_one_element_list ()
```

Returns [] if list has only one element :return:

```
test_remove_smallest_random_list ()
```

Returns a list that misses only one element :return:

```
kyu_7.remove_the_minimum.test_remove_the_minimum.randint (low, high=None,  
                                                         size=None, dtype=int)
```

Return random integers from *low* (inclusive) to *high* (exclusive).

Return random integers from the “discrete uniform” distribution of the specified dtype in the “half-open” interval [*low*, *high*). If *high* is None (the default), then results are from [0, *low*).

Note: New code should use the `integers` method of a `default_rng()` instance instead; please see the random-quick-start.

low [int or array-like of ints] Lowest (signed) integers to be drawn from the distribution (unless *high*=None, in which case this parameter is one above the *highest* such integer).

high [int or array-like of ints, optional] If provided, one above the largest (signed) integer to be drawn from the distribution (see above for behavior if *high*=None). If array-like, must contain integer values

size [int or tuple of ints, optional] Output shape. If the given shape is, e.g., (*m*, *n*, *k*), then *m* * *n* * *k* samples are drawn. Default is None, in which case a single value is returned.

dtype [dtype, optional] Desired dtype of the result. Byteorder must be native. The default value is int.

New in version 1.11.0.

out [int or ndarray of ints] *size*-shaped array of random integers from the appropriate distribution, or a single such random int if *size* not provided.

random_integers [similar to *randint*, only for the closed] interval [*low*, *high*], and 1 is the lowest value if *high* is omitted.

Generator.integers: which should be used for new code.

```
>>> np.random.randint(2, size=10)  
array([1, 0, 0, 0, 1, 1, 0, 0, 1, 0]) # random  
>>> np.random.randint(1, size=10)  
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Generate a 2 x 4 array of ints between 0 and 4, inclusive:

```
>>> np.random.randint(5, size=(2, 4))
array([[4, 0, 2, 1], # random
       [3, 2, 2, 0]])
```

Generate a 1 x 3 array with 3 different upper bounds

```
>>> np.random.randint(1, [3, 5, 10])
array([2, 2, 9]) # random
```

Generate a 1 by 3 array with 3 different lower bounds

```
>>> np.random.randint([1, 5, 7], 10)
array([9, 8, 7]) # random
```

Generate a 2 by 4 array using broadcasting with dtype of uint8

```
>>> np.random.randint([1, 3, 5, 7], [[10], [20]], dtype=np.uint8)
array([[ 8,  6,  9,  7], # random
       [ 1, 16,  9, 12]], dtype=uint8)
```

Module contents

kyu_7.sum_of_two_lowest_int package

Submodules

kyu_7.sum_of_two_lowest_int.sum_two_smallest_int module

kyu_7.sum_of_two_lowest_int.sum_two_smallest_int.**sum_two_smallest_numbers**(numbers)

→

int

Returns the sum of the two lowest positive numbers given an array of minimum 4 positive integers. :param numbers: :return:

kyu_7.sum_of_two_lowest_int.test_sum_two_smallest_numbers module

class kyu_7.sum_of_two_lowest_int.test_sum_two_smallest_numbers.**SumTwoSmallestNumbersTestC**
Bases: unittest.case.TestCase

test_sum_two_smallest_numbers()

Test sum_two_smallest_numbers function The function should return the sum of the two lowest positive numbers :return:

Module contents

kyu_7.you_are_square package

Submodules

kyu_7.you_are_square.test_you_are_square module

```
class kyu_7.you_are_square.test_you_are_square.YouAreSquareTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing is_square function
    The tests will always use some integral number, so don't worry about that in dynamic typed languages.

    test_is_square_25 ()
        25 is a square number :return:

    test_is_square_26 ()
        26 is not a square number :return:

    test_is_square_four ()
        4 is a square number :return:

    test_is_square_negative_numbers ()
        -1: Negative numbers cannot be square numbers :return:

    test_is_square_negative_test ()
        3 is not a square number :return:

    test_is_square_zero ()
        0 is a square number :return:
```

kyu_7.you_are_square.you_are_square module

```
kyu_7.you_are_square.you_are_square.is_square (n) → bool
    Given an integral number, determine if it's a square number: :param n: :return:
```

Module contents

kyu_7.sum_of_powers_of_2 package

Submodules

kyu_7.sum_of_powers_of_2.sum_of_powers_of_2 module

```
kyu_7.sum_of_powers_of_2.sum_of_powers_of_2.powers (n: int) → list
    Return an array of numbers (that are a power of 2) for which the input "n" is the sum :param n: :return:
```

kyu_7.sum_of_powers_of_2.test_sum_of_powers_of_2 module

Testing 'powers' function

```
class kyu_7.sum_of_powers_of_2.test_sum_of_powers_of_2.SumOfPowerOfTwoTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing 'powers' function
    test_powers ()
        The function powers takes a single parameter, the number n, and should return an array of unique numbers.
        :return:
```

Module contents

kyu_7.powers_of_3 package

Submodules

kyu_7.powers_of_3.largest_power module

kyu_7.powers_of_3.largest_power.**largestPower** (*N: int*) → int
 Given a positive integer N, return the largest integer k such that $3^k < N$. :param N: :return:

kyu_7.powers_of_3.test_largest_power module

```
class kyu_7.powers_of_3.test_largest_power.LargestPowerTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing largestPower function
    test_largest_power ()
        Testing largestPower function :return:
```

Module contents

kyu_7.sum_of_triangular_numbers package

Submodules

kyu_7.sum_of_triangular_numbers.sum_triangular_numbers module

kyu_7.sum_of_triangular_numbers.sum_triangular_numbers.**sum_triangular_numbers** (*n: int*) → int
 returns the sum of Triangular Numbers up-to-and-including the nth Triangular Number. :param n: :return:

kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers module

```
class kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers.SumTriangularNumbersTest(  
    Bases: unittest.case.TestCase  
    Testing 'sum_triangular_numbers' function  
  
    test_sum_triangular_numbers_big_number()  
        Testing 'sum_triangular_numbers' function with big number as an input :return:  
  
    test_sum_triangular_numbers_negative_numbers()  
        Testing 'sum_triangular_numbers' function with negative numbers :return:  
  
    test_sum_triangular_numbers_positive_numbers()  
        Testing 'sum_triangular_numbers' function with positive numbers :return:  
  
    test_sum_triangular_numbers_zero()  
        Testing 'sum_triangular_numbers' function with zero as an input :return:
```

Module contents

kyu_7.vaporcode package

Submodules

kyu_7.vaporcode.test_vaporcode module

```
class kyu_7.vaporcode.test_vaporcode.VaporcodeTestCase(methodName='runTest')  
    Bases: unittest.case.TestCase  
    Testing 'vaporcode' function  
  
    test_vaporcode()  
        Testing 'vaporcode' function :return:
```

kyu_7.vaporcode.vaporcode module

```
kyu_7.vaporcode.vaporcode.vaporcode(s: str) → str  
    function that converts any sentence into a V A P O R W A V E sentence :param s: :return:
```

Module contents

kyu_7.simple_fun_152 package

Submodules

kyu_7.simple_fun_152.invite_more_women module

```
kyu_7.simple_fun_152.invite_more_women.invite_more_women(arr: list) → bool  
    Arthur wants to make sure that there are at least as many women as men at this year's party. He gave you a list  
    of integers of all the party goers.  
  
    Arthur needs you to return true if he needs to invite more women or false if he is all set.
```


An array representing the genders of the attendees, where -1 represents women and 1 represents men. :param arr: :return:

kyu_7.simple_fun_152.test_invite_more_women module

```
class kyu_7.simple_fun_152.test_invite_more_women.InviteMoreWomenTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Simple Fun #152: Invite More Women? Testing invite_more_women function
    test_invite_more_women_negative ()
        Simple Fun #152: Invite More Women? Testing invite_more_women function (negative) :return:
    test_invite_more_women_positive ()
        Simple Fun #152: Invite More Women? Testing invite_more_women function (positive) :return:
```

Module contents

kyu_7.significant_figures package

Submodules

kyu_7.significant_figures.number_of_sigfigs module

```
kyu_7.significant_figures.number_of_sigfigs.normalize_string (number: str) → str
    Normalize string by converting it into a number and back to string once again :param number: :return:
kyu_7.significant_figures.number_of_sigfigs.number_of_sigfigs (number: str) →
    int
    return the number of sigfigs in the passed in string "number" :param number: :return:
kyu_7.significant_figures.number_of_sigfigs.remove_extra_leading_zeroes (number:
    str)
    →
    str
    Remove all extra leading zeroes from the head of the string :param number: :return:
kyu_7.significant_figures.number_of_sigfigs.remove_extra_zeroes (number: str)
    → str
    Remove all zeroes from the end of the string :param number: :return:
```

kyu_7.significant_figures.test_number_of_sigfigs module

```
class kyu_7.significant_figures.test_number_of_sigfigs.NumberOfSigFigsTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing number_of_sigfigs function
    test_number_of_sigfigs ()
        Testing number_of_sigfigs function with various test inputs :return:
```

Module contents

kyu_7.sort_out_the_men_from_boys package

Submodules

kyu_7.sort_out_the_men_from_boys.men_from_boys module

kyu_7.sort_out_the_men_from_boys.men_from_boys.**men_from_boys** (arr: List[int]) → list

Sort out the men from the boys.

Men are the Even numbers and Boys are the odd.

Return an array/list where Even numbers come first then odds.

Since, Men are stronger than Boys, then Even numbers in ascending order while odds in descending. :param arr: :return:

kyu_7.sort_out_the_men_from_boys.test_men_from_boys module

class kyu_7.sort_out_the_men_from_boys.test_men_from_boys.**MenFromBoysTestCase** (methodName='run')
Bases: unittest.case.TestCase

Testing men_from_boys function

test_men_from_boys ()

Testing men_from_boys function with various test inputs

Scenario Now that the competition gets tough it will Sort out the men from the boys .

Men are the Even numbers and Boys are the odd !alt !alt

Task Given an array/list [] of n integers , Separate The even numbers from the odds , or Separate the men from the boys !alt !alt

Notes Return an array/list where Even numbers come first then odds. Since , Men are stronger than Boys , Then Even numbers in ascending order While odds in descending. :return:

Module contents

kyu_7.fun_with_lists_length package

Submodules

kyu_7.fun_with_lists_length.length module

kyu_7.fun_with_lists_length.length.**length** (head) → int

The method length, which accepts a linked list (head), and returns the length of the list. :param head: :return:

kyu_7.fun_with_lists_length.node module

```
class kyu_7.fun_with_lists_length.node.Node (data, next=None)
```

Bases: object

The linked list

kyu_7.fun_with_lists_length.test_length module

```
class kyu_7.fun_with_lists_length.test_length.LengthTestCase (methodName='runTest')
```

Bases: unittest.case.TestCase

Testing length function

```
test_length ()
```

Testing length function

The method length, which accepts a linked list (head), and returns the length of the list. :return:

```
test_length_none ()
```

Testing length function where head = None

The method length, which accepts a linked list (head), and returns the length of the list. :return:

Module contents**kyu_7.fill_the_hard_disk_drive package****Submodules****kyu_7.fill_the_hard_disk_drive.save module**

```
kyu_7.fill_the_hard_disk_drive.save.save (sizes: list, hd: int) → int
```

Your task is to determine how many files of the copy queue you will be able to save into your Hard Disk Drive.

Input: Array of file sizes ($0 \leq s \leq 100$) Capacity of the HD ($0 \leq c \leq 500$)

Output: Number of files that can be fully saved in the HD

Parameters

- **sizes** –
- **hd** –

Returns

kyu_7.fill_the_hard_disk_drive.test_save module

```
class kyu_7.fill_the_hard_disk_drive.test_save.SaveTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing 'save' function

    test_save_negative()
        Testing 'save' function: negative

        The function should determine how many files of the copy queue you will be able to save into your Hard
        Disk Drive. :return:

    test_save_positive()
        Testing 'save' function: positive

        The function should determine how many files of the copy queue you will be able to save into your Hard
        Disk Drive. :return:
```

Module contents

kyu_7.the_first_non_repeated_character_in_string package

Submodules

kyu_7.the_first_non_repeated_character_in_string.first_non_repeated module

```
kyu_7.the_first_non_repeated_character_in_string.first_non_repeated.first_non_repeated(s:
                                                                    str)
```

You need to write a function, that returns the first non-repeated character in the given string.

For example for string “test” function should return ‘e’. For string “teeter” function should return ‘r’.

If a string contains all unique characters, then return just the first character of the string. Example: for input “trend” function should return ‘t’

You can assume, that the input string has always non-zero length. :param s: :return:

kyu_7.the_first_non_repeated_character_in_string.test_first_non_repeated module

```
class kyu_7.the_first_non_repeated_character_in_string.test_first_non_repeated.FirstNonRepeated
    Bases: unittest.case.TestCase

    Testing first_non_repeated function

    test_first_non_repeated()
        Testing first_non_repeated function :return:
```

Module contents

`kyu_7.maximum_multiple` package

Submodules

`kyu_7.maximum_multiple.maximum_multiple` module

`kyu_7.maximum_multiple.maximum_multiple.max_multiple` (*divisor: int, bound: int*) → int

Given a Divisor and a Bound , Find the largest integer N , Such That ,

Conditions:

1. N is divisible by divisor
2. N is less than or equal to bound
3. N is greater than 0.

Notes:

1. **The parameters (divisor, bound)** passed to the function are only positive values .
2. It's guaranteed that a divisor is Found .

Parameters

- **divisor** –
- **bound** –

Returns

`kyu_7.maximum_multiple.test_maximum_multiple` module

Module contents

`kyu_7.make_class` package

Submodules

`kyu_7.make_class.animal` module

class `kyu_7.make_class.animal.Animal` (*name, species, age, health, weight, color*)

Bases: `object`

kyu_7.make_class.make_class module

```
kyu_7.make_class.make_class.make_class(*args)
```

kyu_7.make_class.test_make_class module

```
class kyu_7.make_class.test_make_class.MakeClassTestCase(methodName='runTest')
    Bases: unittest.case.TestCase
    Testing make_class function

    test_make_class()
        Testing make_class function :return:
```

Module contents

kyu_7.password_validator package

Submodules

kyu_7.password_validator.password module

```
kyu_7.password_validator.password.password(string: str) → bool
    Your job is to create a simple password validation function, as seen on many websites.
```

You are permitted to use any methods to validate the password.

The rules for a valid password are as follows:

1. **There needs to be at least 1 uppercase letter.**
2. There needs to be at least 1 lowercase letter.
3. There needs to be at least 1 number.
4. The password needs to be at least 8 characters long.

Parameters **string** –

Returns

kyu_7.password_validator.test_password module

```
class kyu_7.password_validator.test_password.PasswordTestCase(methodName='runTest')
    Bases: unittest.case.TestCase
    Testing password function

    test_password()
        Testing password function with various test inputs :return:
```

Module contents

kyu_7.share_prices package

Submodules

kyu_7.share_prices.share_price module

`kyu_7.share_prices.share_price.share_price` (*invested: int, changes: list*) → str

Calculates, and returns the current price of your share, given the following two arguments:

1. `invested(number)`, the amount of money you initially invested in the given share
2. `changes(array of numbers)`, contains your shares daily movement percentages

The returned number, should be in string format, and it's precision should be fixed at 2 decimal numbers. :param invested: :param changes: :return:

kyu_7.share_prices.test_share_price module

class `kyu_7.share_prices.test_share_price.SharePriceTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing share_price function

test_share_price ()

Testing share_price function with multiple test inputs :return:

Module contents

kyu_7.always_perfect package

Submodules

kyu_7.always_perfect.check_root module

`kyu_7.always_perfect.check_root.check_root` (*string: str*) → str

A function which takes numbers separated by commas in string format and returns the number which is a perfect square and the square root of that number.

If string contains other characters than number or it has more or less than 4 numbers separated by comma function returns "incorrect input".

If string contains 4 numbers but not consecutive it returns "not consecutive". :param string: :return:

kyu_7.always_perfect.test_check_root module

```
class kyu_7.always_perfect.test_check_root.CheckRootTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing check_root function

```
test_check_root ()
```

Testing check_root function with various test inputs

A function which takes numbers separated by commas in string format and returns the number which is a perfect square and the square root of that number.

If string contains other characters than number or it has more or less than 4 numbers separated by comma function returns “incorrect input”.

If string contains 4 numbers but not consecutive it returns “not consecutive”. :return:

Module contents

kyu_7.formatting_decimal_places_1 package

Submodules

kyu_7.formatting_decimal_places_1.test_two_decimal_places module

```
class kyu_7.formatting_decimal_places_1.test_two_decimal_places.TwoDecimalPlacesTestCase (m  
    Bases: unittest.case.TestCase
```

Testing two_decimal_places function

```
test_two_decimal_places ()
```

Testing two_decimal_places function with various test inputs

Each floating-point number should be formatted that only the first two decimal places are returned.

You don’t need to check whether the input is a valid number because only valid numbers are used in the tests.

Don’t round the numbers! Just cut them after two decimal places! :return:

kyu_7.formatting_decimal_places_1.two_decimal_places module

```
kyu_7.formatting_decimal_places_1.two_decimal_places.two_decimal_places (number)
```

Each floating-point number should be formatted that only the first two decimal places are returned.

You don’t need to check whether the input is a valid number because only valid numbers are used in the tests.

Don’t round the numbers! Just cut them after two decimal places!

Parameters **number** –

Returns

Module contents

kyu_7.substituting_variables_into_strings_padded_numbers package

Submodules

kyu_7.substituting_variables_into_strings_padded_numbers.solution module

kyu_7.substituting_variables_into_strings_padded_numbers.solution.**solution** (*value*:
int)
→
str

Complete the solution so that it returns a formatted string.

The return value should equal “Value is VALUE” where value is a 5 digit padded number. :param value: :return:

kyu_7.substituting_variables_into_strings_padded_numbers.test_solution module

Testing ‘solution’ function

```
class kyu_7.substituting_variables_into_strings_padded_numbers.test_solution.SolutionTestC  
    Bases: unittest.case.TestCase
```

Testing ‘solution’ function

```
test_solution()  
    Testing ‘solution’ function.
```

The should return a formatted string. The return value should equal “Value is VALUE” where value is a 5 digit padded number. :return:

Module contents

kyu_7.pull_your_words_together_man package

Submodules

kyu_7.pull_your_words_together_man.sentencify module

kyu_7.pull_your_words_together_man.sentencify.**sentencify** (*words*)
The function should:

1. Capitalise the first letter of the first word.
2. Add a period (.) to the end of the sentence.
3. Join the words into a complete string, with spaces.
4. Do no other manipulation on the words.

Parameters **words** –

Returns

kyu_7.pull_your_words_together_man.test_sentencify module

```
class kyu_7.pull_your_words_together_man.test_sentencify.SentencifyTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing 'sentencify' function

```
test_sentencify()  
    Testing 'sentencify' function.
```

The function should:

1. Capitalise the first letter of the first word.
2. Add a period (.) to the end of the sentence.
3. Join the words into a complete string, with spaces.
4. Do no other manipulation on the words.

Returns

Module contents

kyu_7.factorial package

Submodules

kyu_7.factorial.factorial module

```
kyu_7.factorial.factorial.factorial (n: int) → int
```

A function to calculate factorial for a given input. If input is below 0 or above 12 throw an exception of type ValueError (Python). :param n: :return:

kyu_7.factorial.test_factorial module

```
class kyu_7.factorial.test_factorial.FactorialTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing 'factorial' function

```
test_factorial()  
    Testing 'factorial' function
```

In mathematics, the factorial of a non-negative integer n , denoted by $n!$, is the product of all positive integers less than or equal to n . For example: $5! = 5 * 4 * 3 * 2 * 1 = 120$. By convention the value of $0!$ is 1.

Write a function to calculate factorial for a given input. If input is below 0 or above 12 throw an exception of type ValueError (Python). :return:

Module contents

kyu_7.find_the_longest_gap package

Submodules

kyu_7.find_the_longest_gap.gap module

`kyu_7.find_the_longest_gap.gap.calc_g_cur(g_cur, char)`

Calculates g_cur :param g_cur: :param char: :return:

`kyu_7.find_the_longest_gap.gap.calc_g_max(g_cur, g_max)`

Calculates g_max

`kyu_7.find_the_longest_gap.gap.gap(num: int) → int`

Returns the length of its longest binary gap.

The function should return 0 if num doesn't contain a binary gap. :param num: :return:

kyu_7.find_the_longest_gap.test_gap module

class `kyu_7.find_the_longest_gap.test_gap.GapTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing gap function

test_gap()

Testing gap function with various test inputs

A binary gap within a positive number num is any sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of num.

The gap function should return the length of its longest binary gap.

The function should return 0 if num doesn't contain a binary gap. :return:

Module contents

kyu_7.growing_plant package

Submodules

kyu_7.growing_plant.growing_plant module

`kyu_7.growing_plant.growing_plant.growing_plant(upSpeed, downSpeed, desiredHeight)`

Each day a plant is growing by upSpeed meters. Each night that plant's height decreases by downSpeed meters due to the lack of sun heat. Initially, plant is 0 meters tall. We plant the seed at the beginning of a day. We want to know when the height of the plant will reach a certain level. :param upSpeed: :param downSpeed: :param desiredHeight: :return:

kyu_7.growing_plant.test_growing_plant module

```
class kyu_7.growing_plant.test_growing_plant.GrowingPlantTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing growing_plant function

```
test_growing_plant ()  
    Testing growing_plant function
```

Task

Each day a plant is growing by upSpeed meters. Each night that plant's height decreases by downSpeed meters due to the lack of sun heat. Initially, plant is 0 meters tall. We plant the seed at the beginning of a day. We want to know when the height of the plant will reach a certain level.

Example

For upSpeed = 100, downSpeed = 10 and desiredHeight = 910, the output should be 10.

For upSpeed = 10, downSpeed = 9 and desiredHeight = 4, the output should be 1. Because the plant reach to the desired height at day 1(10 meters).

Input/Output

[input] integer upSpeed A positive integer representing the daily growth. Constraints: 5 upSpeed 100.

[input] integer downSpeed A positive integer representing the nightly decline. Constraints: 2 downSpeed < upSpeed.

[input] integer desiredHeight A positive integer representing the threshold. Constraints: 4 desiredHeight 1000.

[output] an integer

The number of days that it will take for the plant to reach/pass desiredHeight (including the last day in the total count).

Module contents

kyu_7.basic_math_add_or_subtract package

Submodules

kyu_7.basic_math_add_or_subtract.calculate module

```
kyu_7.basic_math_add_or_subtract.calculate.calculate (s: str) → str
```

kyu_7.basic_math_add_or_subtract.test_calculate module

```
class kyu_7.basic_math_add_or_subtract.test_calculate.CalculateTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
    Testing calculate function  
    test_calculate()
```

Module contents

kyu_7.sum_of_odd_numbers package

Submodules

kyu_7.sum_of_odd_numbers.row_sum_odd_numbers module

```
kyu_7.sum_of_odd_numbers.row_sum_odd_numbers.calc_first_number (n: int) → int  
    Calculate first number in the row :param n: :return:  
  
kyu_7.sum_of_odd_numbers.row_sum_odd_numbers.calc_last_number (n: int) → int  
    Calculate last number in the row :param n: :return:  
  
kyu_7.sum_of_odd_numbers.row_sum_odd_numbers.odd_row (n: int) → list  
    Given a triangle of consecutive odd numbers finds the triangle's row knowing its index (the rows are 1-indexed).  
    :param n: :return:  
  
kyu_7.sum_of_odd_numbers.row_sum_odd_numbers.row_sum_odd_numbers (n: int) → int  
    Given the triangle of consecutive odd numbers calculate the row sums of this triangle from the row index  
    (starting at index 1) :param n: :return:
```

kyu_7.sum_of_odd_numbers.test_row_sum_odd_numbers module

```
class kyu_7.sum_of_odd_numbers.test_row_sum_odd_numbers.OddRowTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
    Testing row_sum_odd_numbers function  
    test_row_sum_odd_numbers()
```

Module contents

kyu_7.help_bob_count_letters_and_digits package

Submodules

kyu_7.help_bob_count_letters_and_digits.count_letters_and_digits module

```
kyu_7.help_bob_count_letters_and_digits.count_letters_and_digits.count_letters_and_digits (s)
```

A method that can determine how many letters and digits are in a given string. :param s: :return:

kyu_7.help_bob_count_letters_and_digits.test_count_letters_and_digits module

```
class kyu_7.help_bob_count_letters_and_digits.test_count_letters_and_digits.CalculateTestCase
    Bases: unittest.case.TestCase
    Testing count_letters_and_digits function
    test_calculate()
```

Module contents

kyu_7.easy_line package

Submodules

kyu_7.easy_line.easylines module

`kyu_7.easy_line.easylines.calc_combination_per_row_item(row: int, i: int) → int`
Generates a specific combination from Pascal's Triangle row by specified index :param row: row :param i: index :return:

`kyu_7.easy_line.easylines.easy_line(n: int) → int`
The function will take n (with: $n \geq 0$) as parameter and will return the sum of the squares of the binomial coefficients on line n.

Parameters `n` – the line number (with: $n \geq 0$)

Returns

kyu_7.easy_line.test_easylines module

```
class kyu_7.easy_line.test_easylines.EasyLineTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

We want to calculate the sum of the squares of the binomial coefficients on a given line with a function called easylines (or easyLine or easy-line).

Can you write a program which calculate easylines(n) where n is the line number?

The function will take n (with: $n \geq 0$) as parameter and will return the sum of the squares of the binomial coefficients on line n.

```
test_calc_combinations_per_row()
test_easy_line()
test_easy_line_exception()
```

Module contents

kyu_7.isograms package

Submodules

kyu_7.isograms.is_isogram module

kyu_7.isograms.is_isogram.**is_isogram**(string: str) → bool

Determines whether a string that contains only letters is an isogram

Parameters string – str

Returns bool

kyu_7.isograms.test_is_isogram module

Testing 'is_isogram' function

```
class kyu_7.isograms.test_is_isogram.IsIsogramTestCase (methodName='runTest')
```

Bases: unittest.case.TestCase

Testing 'is_isogram' function

```
test_is_isogram()
```

Testing 'is_isogram' function

Module contents

1.7.2 Module contents

1.8 kyu_8 package

1.8.1 Subpackages

kyu_8.is_your_period_late package

Submodules

kyu_8.is_your_period_late.is_your_period_late module

kyu_8.is_your_period_late.is_your_period_late.**period_is_late**(last: datetime.date, today: datetime.date, cycle_length: int) → bool

Test whether a period is late.

Parameters

- **last** – The Date object with the date of the last period
- **today** – The Date object with the date of the check

- **cycle_length** – Integer representing the length of the cycle in days

Returns

kyu_8.is_your_period_late.test_is_your_period_late module

```
class kyu_8.is_your_period_late.test_is_your_period_late.PeriodIsLateTestCase (methodName='run')
    Bases: unittest.case.TestCase
    Testing period_is_late function
    test_period_is_late_negative()
        Negative tests :return:
    test_period_is_late_positive()
        Positive tests :return:
```

Module contents

kyu_8.logical_calculator package

Submodules

kyu_8.logical_calculator.logical_calculator module

```
kyu_8.logical_calculator.logical_calculator.logical_calc (array: list, op: str) → bool
```

Calculates logical value of boolean array.

Logical operations: AND, OR and XOR.

Begins at the first value, and repeatedly apply the logical operation across the remaining elements in the array sequentially.

Parameters

- **array** –
- **op** –

Returns

kyu_8.logical_calculator.test_logical_calculator module

```
class kyu_8.logical_calculator.test_logical_calculator.LogicalCalculatorTestCase (methodName='run')
    Bases: unittest.case.TestCase
    Testing logical_calc function
    test_logical_calc_and()
        And () is the truth-functional operator of logical conjunction
        The and of a set of operands is true if and only if all of its operands are true.
        Source: https://en.wikipedia.org/wiki/Logical\_conjunction
```

Returns

test_logical_calc_or()

In logic and mathematics, or is the truth-functional operator of (inclusive) disjunction, also known as alternation.

The or of a set of operands is true if and only if one or more of its operands is true.

Source: https://en.wikipedia.org/wiki/Logical_disjunction

Returns**test_logical_calc_xor()**

Exclusive or or exclusive disjunction is a logical operation that outputs true only when inputs differ (one is true, the other is false).

XOR outputs true whenever the inputs differ:

Source: https://en.wikipedia.org/wiki/Exclusive_or :return:

Module contents**kyu_8.multiply package****Submodules****kyu_8.multiply.multiply module****Multiply Problem Description**

The code does not execute properly. Try to figure out why.

```
def multiply(a, b): a * b
```

Source: <https://www.codewars.com/kata/50654ddff44f8002000000004/train/python>

```
kyu_8.multiply.multiply.multiply(a, b)
```

Multiply two numbers and return the result :param a: :param b: :return:

kyu_8.multiply.test_multiply module

```
class kyu_8.multiply.test_multiply.MultiplyTestCase(methodName='runTest')
```

Bases: unittest.case.TestCase

Testing multiply function

```
test_multiply()
```

Verify that multiply function returns correct result :return:

Module contents

kyu_8.grasshopper_personalized_message package

Submodules

kyu_8.grasshopper_personalized_message.grasshopper_personalized_message module

kyu_8.grasshopper_personalized_message.grasshopper_personalized_message.**greet** (*name*,
owner)
→
str

Function that gives a personalized greeting. This function takes two parameters: name and owner.

Parameters

- **name** –
- **owner** –

Returns

kyu_8.grasshopper_personalized_message.test_grasshopper_personalized_message module

class kyu_8.grasshopper_personalized_message.test_grasshopper_personalized_message.**GreetTest**

Bases: unittest.case.TestCase

Testing greet function

test_greet ()

Use conditionals to verify that greet function returns the proper message. :return:

Module contents

kyu_8.grasshopper_messi_goals_function package

Submodules

kyu_8.grasshopper_messi_goals_function.messi_goals_function module

kyu_8.grasshopper_messi_goals_function.messi_goals_function.**goals** (*laLiga*: int,
copaDel-
Rey: int,
champi-
onsLeague:
int) → int

The function returns Messi's total number of goals in all three leagues: - LaLiga - Copa del Rey - Champions

Parameters

- **laLiga** –
- **copaDelRey** –
- **championsLeague** –

Returns

kyu_8.grasshopper_messi_goals_function.test_messi_goals_function module

```
class kyu_8.grasshopper_messi_goals_function.test_messi_goals_function.GoalsTestCase (method  
    Bases: unittest.case.TestCase  
    Testing goals function  
    test_goals ()  
        Verify that the function returns Messi's total number of goals in all three leagues. :return:
```

Module contents

kyu_8.remove_string_spaces package

Submodules

kyu_8.remove_string_spaces.remove_string_spaces module

```
kyu_8.remove_string_spaces.remove_string_spaces.no_space (x) → str  
    Remove the spaces from the string, then return the resultant string. :param x: :return:
```

kyu_8.remove_string_spaces.test_remove_string_spaces module

```
class kyu_8.remove_string_spaces.test_remove_string_spaces.NoSpaceTestCase (methodName='runTest'  
    Bases: unittest.case.TestCase  
    Testing no_space function  
    test_something ()  
        Test that no_space function removes the spaces from the string, then return the resultant string. :return:
```

Module contents

kyu_8.well_of_ideas_easy_version package

Submodules

kyu_8.well_of_ideas_easy_version.test_well_of_ideas_easy_version module

```
class kyu_8.well_of_ideas_easy_version.test_well_of_ideas_easy_version.WellTestCase (methodName='runTest'  
    Bases: unittest.case.TestCase  
    Testing well function  
    test_well_fail ()  
        If there are no good ideas, as is often the case, return 'Fail!'. :return:  
    test_well_publish ()  
        If there are one or two good ideas, return 'Publish!'. :return:
```

```
test_well_series()
    if there are more than 2 return 'I smell a series!':return:
```

kyu_8.well_of_ideas_easy_version.well_of_ideas_easy_version module

```
kyu_8.well_of_ideas_easy_version.well_of_ideas_easy_version.well(x: List[str])
    → str
```

Module contents

kyu_8.make_upper_case package

Submodules

kyu_8.make_upper_case.make_upper_case module

```
kyu_8.make_upper_case.make_upper_case.make_upper_case(s)
    Function that make UpperCase. :param s: :return:
```

kyu_8.make_upper_case.test_make_upper_case module

```
class kyu_8.make_upper_case.test_make_upper_case.MakeUpperCaseTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing make_upper_case function
    test_make_upper_case()
        Sample Tests for make_upper_case function :return:
```

Module contents

kyu_8.terminal_game_move_function package

Submodules

kyu_8.terminal_game_move_function.terminal_game_move_function module

```
kyu_8.terminal_game_move_function.terminal_game_move_function.move(position:
    int, roll:
    int) → int
    A function for the terminal game that takes the current position of the hero and the roll (1-6) and return the new
    position. :param position: :param roll: :return:
```

kyu_8.terminal_game_move_function.test_terminal_game_move_function module

```
class kyu_8.terminal_game_move_function.test_terminal_game_move_function.MoveTestCase (methodNo
    Bases: unittest.case.TestCase
    Testing move function
    test_move()
        The player rolls the dice and moves the number of spaces indicated by the dice two times.
        Pass position and roll and compare the output to the expected result :return:
```

Module contents**kyu_8.wolf_in_sheep_clothing package****Submodules****kyu_8.wolf_in_sheep_clothing.test_wolf_in_sheep_clothing module**

```
class kyu_8.wolf_in_sheep_clothing.test_wolf_in_sheep_clothing.WarnTheSheepTestCase (methodNo
    Bases: unittest.case.TestCase
    Testing warn_the_sheep function
    test_warn_the_sheep_wolf_at_end()
        If the wolf is not the closest animal to you, return “Oi! Sheep number N! You are about to be eaten by a
        wolf!” where N is the sheep’s position in the queue. :return:
    test_warn_the_sheep_wolf_at_start()
        If the wolf is the closest animal to you, return “Pls go away and stop eating my sheep”. :return:
    test_warn_the_sheep_wolf_in_middle()
        If the wolf is the closest animal to you, return “Pls go away and stop eating my sheep”. :return:
```

kyu_8.wolf_in_sheep_clothing.wolf_in_sheep_clothing module

```
kyu_8.wolf_in_sheep_clothing.wolf_in_sheep_clothing.warn_the_sheep (queue:
                                                                    list)    →
                                                                    str
```

Warn the sheep in front of the wolf that it is about to be eaten.

If the wolf is the closest animal to you, return “Pls go away and stop eating my sheep”.

Otherwise, return “Oi! Sheep number N! You are about to be eaten by a wolf!” where N is the sheep’s position in the queue.

Parameters queue –

Returns

Module contents

kyu_8.find_the_first_non_consecutive_number package

Submodules

kyu_8.find_the_first_non_consecutive_number.first_non_consecutive module

kyu_8.find_the_first_non_consecutive_number.first_non_consecutive.**first_non_consecutive**(arr:
list

Find the first element of an array that is not consecutive.

E.g. If we have an array [1,2,3,4,6,7,8] then 1 then 2 then 3 then 4 are all consecutive but 6 is not, so that's the first non-consecutive number.

If the whole array is consecutive then return null or Nothing. :param arr: :return:

kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive module

class kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive.**FirstNonConsecutive**
Bases: unittest.case.TestCase

Testing first_non_consecutive function

test_first_non_consecutive_large_list()

Large lists :return:

test_first_non_consecutive_negative()

non-consecutive is a negative number. :return:

test_first_non_consecutive_none()

If the whole array is consecutive then return null or Nothing or None. :return:

test_first_non_consecutive_positive()

If we have an array [1,2,3,4,6,7,8] then 1 then 2 then 3 then 4 are all consecutive but 6 is not, so that's the first non-consecutive number. :return:

Module contents

kyu_8.third_angle_of_triangle package

Submodules

kyu_8.third_angle_of_triangle.test_third_angle_of_triangle module

class kyu_8.third_angle_of_triangle.test_third_angle_of_triangle.**OtherAngleTestCase** (methodN
Bases: unittest.case.TestCase

Testing other_angle

test_other_angle()

You are given two angles (in degrees) of a triangle. Find the 3rd. :return:

kyu_8.third_angle_of_triangle.third_angle_of_triangle module

kyu_8.third_angle_of_triangle.third_angle_of_triangle.**other_angle**(a: int, b: int) → int

You are given two angles (in degrees) of a triangle.

Write a function to return the 3rd.

Note: only positive integers will be tested. :param a: :param b: :return:

Module contents

kyu_8.remove_first_and_last_character package

Submodules

kyu_8.remove_first_and_last_character.remove_char module

kyu_8.remove_first_and_last_character.remove_char.**remove_char**(s)

A function that removes the first and last characters of a string.

You're given one parameter, the original string.

You don't have to worry with strings with less than two characters. :param s: :return:

kyu_8.remove_first_and_last_character.test_remove_char module

class kyu_8.remove_first_and_last_character.test_remove_char.**RemoveCharTestCase**(methodName='')

Bases: unittest.case.TestCase

Testing remove_char function

test_remove_char()

Test that 'remove_char' function removes the first and last characters of a string. :return:

Module contents

kyu_8.reversed_strings package

Submodules

kyu_8.reversed_strings.reversed_strings module

kyu_8.reversed_strings.reversed_strings.**solution**(string) → str

reverses the string value passed into it :param string: :return:

kyu_8.reversed_strings.test_reversed_strings module

```
class kyu_8.reversed_strings.test_reversed_strings.ReversedStringsTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing the solution for 'Reversed Strings' problem

    test_reversed_strings ()
        Test with regular string :return:

    test_reversed_strings_empty ()
        Test with empty string :return:

    test_reversed_strings_one_char ()
        Test with one char only :return:
```

Module contents

kyu_8.surface_area_and_volume_of_box package

Submodules

kyu_8.surface_area_and_volume_of_box.get_size module

```
kyu_8.surface_area_and_volume_of_box.get_size.get_size (w, h, d) → list
    Write a function that returns the total surface area and volume of a box as an array: [area, volume] :param w:
    :param h: :param d: :return:
```

kyu_8.surface_area_and_volume_of_box.test_get_size module

```
class kyu_8.surface_area_and_volume_of_box.test_get_size.GetSizeTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing get_size function

    test_get_size ()
        Testing get_size function with various inputs :return:
```

Module contents

kyu_8.alternating_case package

Submodules

kyu_8.alternating_case.alternating_case module

```
kyu_8.alternating_case.alternating_case.to_alternating_case (string: str) → str
    each lowercase letter becomes uppercase and each uppercase letter becomes lowercase :param string: :return:
```


kyu_8.alternating_case.test_alternating_case module

```
class kyu_8.alternating_case.test_alternating_case.AlternatingCaseTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing to_alternating_case function

    test_alternating_case()
        Testing to_alternating_case function :return:
```

Module contents**kyu_8.grasshopper_summation package****Submodules****kyu_8.grasshopper_summation.summation module**

```
kyu_8.grasshopper_summation.summation.summation (num: int) → int
    A program that finds the summation of every number from 1 to num. The number will always be a positive
    integer greater than 0. :param num: :return:
```

kyu_8.grasshopper_summation.test_summation module

```
class kyu_8.grasshopper_summation.test_summation.SummationTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing summation function

    test_summation()
        Testing summation function with various test inputs :return:
```

Module contents**kyu_8.my_head_is_at_the_wrong_end package****Submodules****kyu_8.my_head_is_at_the_wrong_end.fix_the_meerkat module**

```
kyu_8.my_head_is_at_the_wrong_end.fix_the_meerkat.fix_the_meerkat (arr: list)
                                     → list
    You will be given an array which will have three values (tail, body, head). It is your job to re-arrange the array
    so that the animal is the right way round (head, body, tail). :param arr: :return:
```

kyu_8.my_head_is_at_the_wrong_end.test_fix_the_meerkat module

```
class kyu_8.my_head_is_at_the_wrong_end.test_fix_the_meerkat.FixTheMeerkatTestCase (methodName
    Bases: unittest.case.TestCase
    Testing fix_the_meerkat function
    test_fix_the_meerkat ()
```

Module contents

kyu_8.swap_values package

Submodules

kyu_8.swap_values.swap_values module

```
kyu_8.swap_values.swap_values.swap_values (args: list) → None
    Swap values :param args: :return:
```

kyu_8.swap_values.test_swap_values module

```
class kyu_8.swap_values.test_swap_values.SwapValuesTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing swap_values function
    test_swap_values ()
        Testing swap_values function
```

Module contents

kyu_8.keep_hydrated package

Submodules

kyu_8.keep_hydrated.keep_hydrated module

```
kyu_8.keep_hydrated.keep_hydrated.litres (time) → int
    Because Nathan knows it is important to stay hydrated, he drinks 0.5 litres of water per hour of cycling.
    You get given the time in hours and you need to return the number of litres Nathan will drink, rounded to the
    smallest value. :param time: :return:
```

kyu_8.keep_hydrated.test_keep_hydrated module

```
class kyu_8.keep_hydrated.test_keep_hydrated.KeepHydratedTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
        Testing litres function  
  
    test_keep_hydrated()  
        Testing litres function with various test inputs :return:
```

Module contents

kyu_8.set_alarm package

Submodules

kyu_8.set_alarm.set_alarm module

```
kyu_8.set_alarm.set_alarm.set_alarm (employed, vacation)
```

A function named setAlarm which receives two parameters. The first parameter, employed, is true whenever you are employed and the second parameter, vacation is true whenever you are on vacation.

The function should return true if you are employed and not on vacation (because these are the circumstances under which you need to set an alarm). It should return false otherwise.

Examples:

```
setAlarm(true, true) -> false setAlarm(false, true) -> false setAlarm(false, false) -> false setAlarm(true, false) ->  
true
```

Parameters

- **employed** –
- **vacation** –

Returns

kyu_8.set_alarm.test_set_alarm module

```
class kyu_8.set_alarm.test_set_alarm.SetAlarmTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase  
        Testing set_alarm function  
  
    test_set_alarm()  
        Testing set_alarm function with various test inputs.
```

The function should return true if you are employed and not on vacation (because these are the circumstances under which you need to set an alarm). It should return false otherwise.

Examples:

```
setAlarm(true, true) -> false setAlarm(false, true) -> false setAlarm(false, false) -> false setAlarm(true,  
false) -> true :return:
```

Module contents

kyu_8.will_there_be_enough_space package

Submodules

kyu_8.will_there_be_enough_space.enough module

`kyu_8.will_there_be_enough_space.enough.enough (cap: int, on: int, wait: int) → int`

The driver wants you to write a simple program telling him if he will be able to fit all the passengers.

If there is enough space, return 0, and if there isn't, return the number of passengers he can't take.

You have to write a function that accepts three parameters:

cap is the amount of people the bus can hold excluding the driver. on is the number of people on the bus. wait is the number of people waiting to get on to the bus.

Parameters

- **cap** –
- **on** –
- **wait** –

Returns

kyu_8.will_there_be_enough_space.test_enough module

class `kyu_8.will_there_be_enough_space.test_enough.EnoughTestCase (methodName='runTest')`
Bases: `unittest.case.TestCase`

Testing enough function

test_enough ()

Testing enough function with various test data

If there is enough space, return 0, and if there isn't, return the number of passengers he can't take. :return:

Module contents

kyu_8.counting_sheep package

Submodules

kyu_8.counting_sheep.counting_sheep module

`kyu_8.counting_sheep.counting_sheep.count_sheeps (arrayOfSheeps: list) → int`

Consider an array of sheep where some sheep may be missing from their place. We need a function that counts the number of sheep present in the array (true means present).

Hint: Don't forget to check for bad values like null/undefined :param arrayOfSheeps: :return:

kyu_8.counting_sheep.test_counting_sheep module

```
class kyu_8.counting_sheep.test_counting_sheep.CountingSheepTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing 'count_sheeps' function

    test_counting_sheep ()
        Testing 'count_sheeps' function Consider an array of sheep where some sheep may be missing from their
        place. We need a function that counts the number of sheep present in the array (true means present).
        :return:

    test_counting_sheep_bad_input ()
        Testing 'count_sheeps' function Hint: Don't forget to check for bad values like null/undefined :return:

    test_counting_sheep_empty_list ()
        Testing 'count_sheeps' function Hint: Don't forget to check for bad values like empty list :return:

    test_counting_sheep_mixed_list ()
        Testing 'count_sheeps' function Hint: Don't forget to check for bad values like mixed list :return:
```

Module contents**kyu_8.grasshopper_check_for_factor package****Submodules****kyu_8.grasshopper_check_for_factor.check_for_factor module**

```
kyu_8.grasshopper_check_for_factor.check_for_factor.check_for_factor (base,
                                                                    factor)

    This function should test if the factor is a factor of base.

    Factors are numbers you can multiply together to get another number.

    Return true if it is a factor or false if it is not. :param base: :param factor: :return:
```

kyu_8.grasshopper_check_for_factor.test_check_for_factor module

```
class kyu_8.grasshopper_check_for_factor.test_check_for_factor.CheckForFactorTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing check_for_factor function.

    test_check_for_factor_false ()
        Testing check_for_factor function.

        This function should test if the factor is a factor of base.

        Return false if it is not a factor. :return:

    test_check_for_factor_true ()
        Testing check_for_factor function.

        This function should test if the factor is a factor of base.

        Return true if it is a factor. :return:
```

Module contents

kyu_8.check_the_exam package

Submodules

kyu_8.check_the_exam.check_exam module

`kyu_8.check_the_exam.check_exam.char_processor(char: str, results: list) → None`

Processing chars based on specified rule :param char: :param results: :return:

`kyu_8.check_the_exam.check_exam.check_exam(arr1, arr2)`

The first input array contains the correct answers to an exam, like ["a", "a", "b", "d"]. The second one is "answers" array and contains student's answers.

The two arrays are not empty and are the same length. Return the score for this array of answers, giving +4 for each correct answer, -1 for each incorrect answer, and +0 for each blank answer(empty string).

If the score < 0, return 0. :param arr1: :param arr2: :return:

kyu_8.check_the_exam.test_check_exam module

`class kyu_8.check_the_exam.test_check_exam.CheckExamTestCase (methodName='runTest')`

Bases: `unittest.case.TestCase`

Testing check_exam function

`test_check_exam()`

Testing check_exam function

The function should return the score for this array of answers, giving +4 for each correct answer, -1 for each incorrect answer, and +0 for each blank answer(empty string). :return:

Module contents

kyu_8.is_it_a_palindrome package

Submodules

kyu_8.is_it_a_palindrome.is_palindrome module

`kyu_8.is_it_a_palindrome.is_palindrome.is_palindrome(s: str) → bool`

Write function isPalindrome that checks if a given string (case insensitive) is a palindrome. :param s: :return:

kyu_8.is_it_a_palindrome.test_is_palindrome module

```
class kyu_8.is_it_a_palindrome.test_is_palindrome.IsPalindromeTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing is_palindrome function

    test_is_palindrome()
        Testing is_palindrome function with various test inputs
        The function should check if a given string (case insensitive) is a palindrome.
```

Module contents

kyu_8.formatting_decimal_places_0 package

Submodules

kyu_8.formatting_decimal_places_0.test_two_decimal_places module

```
class kyu_8.formatting_decimal_places_0.test_two_decimal_places.TwoDecimalPlacesTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing two_decimal_places function

    test_two_decimal_places()
        Testing two_decimal_places function with various test inputs.
        Each number should be formatted that it is rounded to two decimal places. You don't need to check
        whether the input is a valid number because only valid numbers are used in the tests. :return:
```

kyu_8.formatting_decimal_places_0.two_decimal_places module

```
kyu_8.formatting_decimal_places_0.two_decimal_places.two_decimal_places(n)
    Each number should be formatted that it is rounded to two decimal places. You don't need to check whether the
    input is a valid number because only valid numbers are used in the tests. :param n: :return:
```

Module contents

kyu_8.convert_string_to_an_array package

Submodules

kyu_8.convert_string_to_an_array.string_to_array module

```
kyu_8.convert_string_to_an_array.string_to_array.string_to_array(s: str) → list
    A function to split a string and convert it into an array of words :param s: :return:
```

kyu_8.convert_string_to_an_array.test_string_to_array module

```
class kyu_8.convert_string_to_an_array.test_string_to_array.StringToArrayTestCase (methodName)
    Bases: unittest.case.TestCase

    Testing string_to_array function.

    test_string_to_array()
        Testing string_to_array function.

        A function to split a string and convert it into an array of words. :return:
```

Module contents

kyu_8.the_feast_of_many_beasts package

Submodules

kyu_8.the_feast_of_many_beasts.feast module

```
kyu_8.the_feast_of_many_beasts.feast.feast (beast: str, dish: str) → bool
```

A function feast that takes the animal's name and dish as arguments and returns true or false to indicate whether the beast is allowed to bring the dish to the feast.

Assume that beast and dish are always lowercase strings, and that each has at least two letters. beast and dish may contain hyphens and spaces, but these will not appear at the beginning or end of the string. They will not contain numerals. :param beast: :param dish: :return:

kyu_8.the_feast_of_many_beasts.test_feast module

```
class kyu_8.the_feast_of_many_beasts.test_feast.FeastTestCase (methodName='runTest')
    Bases: unittest.case.TestCase

    Testing 'feast' function

    test_feast()
        Testing 'feast' function with various test inputs

        Testing a function feast that takes the animal's name and dish as arguments and returns true or false to indicate whether the beast is allowed to bring the dish to the feast.

        Assume that beast and dish are always lowercase strings, and that each has at least two letters. beast and dish may contain hyphens and spaces, but these will not appear at the beginning or end of the string. They will not contain numerals.

        There is just one rule: the dish must start and end with the same letters as the animal's name. For example, the great blue heron is bringing garlic naan and the chickadee is bringing chocolate cake. :return:
```


Module contents

kyu_8.count_the_monkeys package

Submodules

kyu_8.count_the_monkeys.monkey_count module

`kyu_8.count_the_monkeys.monkey_count.monkey_count (n: int) → list`

You take your son to the forest to see the monkeys. You know that there are a certain number there (n), but your son is too young to just appreciate the full number, he has to start counting them from 1.

As a good parent, you will sit and count with him. Given the number (n), populate an array with all numbers up to and including that number, but excluding zero. :param n: :return:

kyu_8.count_the_monkeys.test_monkey_count module

class `kyu_8.count_the_monkeys.test_monkey_count.MonkeyCountTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing monkey_count function

test_monkey_count ()

Testing monkey_count function

You take your son to the forest to see the monkeys. You know that there are a certain number there (n), but your son is too young to just appreciate the full number, he has to start counting them from 1.

As a good parent, you will sit and count with him. Given the number (n), populate an array with all numbers up to and including that number, but excluding zero. :return:

Module contents

kyu_8.keep_up_the_hoop package

Submodules

kyu_8.keep_up_the_hoop.hoop_count module

`kyu_8.keep_up_the_hoop.hoop_count.hoop_count (n) → str`

A program where Alex can input (n) how many times the hoop goes round and it will return him an encouraging message :param n: :return:

kyu_8.keep_up_the_hoop.test_hoop_count module

```
class kyu_8.keep_up_the_hoop.test_hoop_count.HoopCountTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing hoop_count function

```
test_hoop_count_negative()
```

```
test_hoop_count_positive()
```

Testing hoop_count function

Alex just got a new hula hoop, he loves it but feels discouraged because his little brother is better than him

Write a program where Alex can input (n) how many times the hoop goes round and it will return him an encouraging message

- If Alex gets 10 or more hoops, return the string “Great, now move on to tricks”.
- If he doesn't get 10 hoops, return the string “Keep at it until you get it”.

Returns

Module contents

kyu_8.enumerable_magic_25 package

Submodules

kyu_8.enumerable_magic_25.take module

```
kyu_8.enumerable_magic_25.take.take (arr: list, n: int) → list
```

Accepts a list/array and a number n, and returns a list/array array of the first n elements from the list/array.

Parameters

- **arr** –
- **n** –

Returns

kyu_8.enumerable_magic_25.test_take module

```
class kyu_8.enumerable_magic_25.test_take.TakeTestCase (methodName='runTest')  
    Bases: unittest.case.TestCase
```

Testing take function

```
test_take()
```

Module contents

kyu_8.will_you_make_it package

Submodules

kyu_8.will_you_make_it.test_zero_fuel module

```
class kyu_8.will_you_make_it.test_zero_fuel.ZeroFuelTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
    Testing zero_fuel
    test_zero_fuel()
```

kyu_8.will_you_make_it.zero_fuel module

```
kyu_8.will_you_make_it.zero_fuel.zero_fuel (distance_to_pump: int, mpg: int, fuel_left:
                                             int) → bool
```

You were camping with your friends far away from home, but when it's time to go back, you realize that you fuel is running out and the nearest pump is 50 miles away! You know that on average, your car runs on about 25 miles per gallon. There are 2 gallons left. Considering these factors, write a function that tells you if it is possible to get to the pump or not. Function should return true (1 in Prolog) if it is possible and false (0 in Prolog) if not. The input values are always positive.

Parameters

- **distance_to_pump** –
- **mpg** –
- **fuel_left** –

Returns

Module contents

kyu_8.century_from_year package

Submodules

kyu_8.century_from_year.century module

```
kyu_8.century_from_year.century.century (year)
    Given a year, return the century it is in :param year: :return:
```

kyu_8.century_from_year.test_century module

```
class kyu_8.century_from_year.test_century.CenturyTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

The first century spans from the year 1 up to and including the year 100, The second - from the year 101 up to and including the year 200, etc.

```
test_century()
    Testing century function
```

Module contents

kyu_8.holiday_vi_shark_pontoon package

Submodules

kyu_8.holiday_vi_shark_pontoon.shark module

```
kyu_8.holiday_vi_shark_pontoon.shark.shark(pontoonDistance, sharkDistance, youSpeed,
                                             sharkSpeed, dolphin) → str
```

You are given 5 variables: sharkDistance = distance the shark needs to cover to eat you in metres, sharkSpeed = how fast it can move in metres/second, pontoonDistance = how far you need to swim to safety in metres, youSpeed = how fast you can swim in metres/second, dolphin = a boolean, if true, you can half the swimming speed of the shark as the dolphin will attack it.

If you make it, return “Alive!”, if not, return “Shark Bait!”.

Parameters

- **pontoonDistance** –
- **sharkDistance** –
- **youSpeed** –
- **sharkSpeed** –
- **dolphin** –

Returns

kyu_8.holiday_vi_shark_pontoon.test_shark module

```
class kyu_8.holiday_vi_shark_pontoon.test_shark.SharkTestCase (methodName='runTest')
    Bases: unittest.case.TestCase
```

Testing shark function

```
test_shark_alive_1()
    Testing shark function -> positive :return:
```

```
test_shark_alive_2()
    Testing shark function -> positive :return:
```

```
test_shark_bait()
    Testing shark function -> negative :return:
```

Module contents

kyu_8.greek_sort package

Submodules

kyu_8.greek_sort.greek_comparator module

`kyu_8.greek_sort.greek_comparator.greek_comparator` (*lhs: str, rhs: str*) → int

A custom comparison function of two arguments (iterable elements) which should return a negative, zero or positive number depending on whether the first argument is considered smaller than, equal to, or larger than the second argument :param lhs: :param rhs: :return:

kyu_8.greek_sort.test_greek_comparator module

class `kyu_8.greek_sort.test_greek_comparator.GreekComparatorTestCase` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Testing greek_comparator function

test_greek_comparator ()

Testing greek_comparator function with various test inputs :return:

Module contents

1.8.2 Module contents

1.9 utils package

1.9.1 Subpackages

utils.primes package

Submodules

utils.primes.is_prime module

`utils.primes.is_prime.is_prime` (*n: int*) → bool

Function to check for a prime number Return TRUE if 'n' is prime number. False otherwise :param n: :return:

utils.primes.primes_generator module

`utils.primes.primes_generator.gen_primes()`
Generate an infinite sequence of prime numbers.

utils.primes.test_is_prime module

class `utils.primes.test_is_prime.IsPrimeTestCase` (*methodName='runTest'*)
Bases: `unittest.case.TestCase`
Testing `is_prime` function
test_is_prime_negative()
Negative test cases for `is_prime` function testing :return:
test_is_prime_positive()
Positive test cases for `is_prime` function testing :return:

utils.primes.test_primes_generator module

class `utils.primes.test_primes_generator.GenPrimesTestCase` (*methodName='runTest'*)
Bases: `unittest.case.TestCase`
Testing `gen_primes` function
test_gen_primes_negative()
Negative test cases for `gen_primes` function testing :return:
test_gen_primes_positive()
Positive test cases for `gen_primes` function testing :return:

Module contents

1.9.2 Submodules

1.9.3 utils.log_func module

`utils.log_func.print_log(**kwargs)` → None
Print log :param args: :return:

1.9.4 Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

i

img, 1

k

kyu_2, 2

kyu_2.evaluate_mathematical_expression,
2

kyu_2.evaluate_mathematical_expression.evaluate,
1

kyu_2.evaluate_mathematical_expression.test_evaluate,
2

kyu_3, 7

kyu_3.battleship_field_validator, 7

kyu_3.battleship_field_validator.test_battleship_validator,
6

kyu_3.battleship_field_validator.validator,
6

kyu_3.calculator, 3

kyu_3.calculator.calculator, 2

kyu_3.calculator.test_calculator, 3

kyu_3.make_spiral, 6

kyu_3.make_spiral.solution, 4

kyu_3.make_spiral.test_spiralize, 5

kyu_3.rail_fence_cipher_encoding_and_decoding,
4

kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding,
3

kyu_3.rail_fence_cipher_encoding_and_decoding.test_decoding,
4

kyu_3.rail_fence_cipher_encoding_and_decoding.test_encoding,
4

kyu_4, 17

kyu_4.human_readable_duration_format, 9

kyu_4.human_readable_duration_format.format_duration,
7

kyu_4.human_readable_duration_format.test_format_duration,
9

kyu_4.most_frequently_used_words, 13

kyu_4.most_frequently_used_words.solution,
13

kyu_4.most_frequently_used_words.test_top_3_words,
13

kyu_4.next_bigger_number_with_the_same_digits,
17

kyu_4.next_bigger_number_with_the_same_digits.next,
16

kyu_4.next_bigger_number_with_the_same_digits.test,
17

kyu_4.next_smaller_number_with_the_same_digits,
16

kyu_4.next_smaller_number_with_the_same_digits.next,
15

kyu_4.next_smaller_number_with_the_same_digits.test,
16

kyu_4.range_extraction, 10

kyu_4.range_extraction.solution, 10

kyu_4.range_extraction.test_solution,
10

kyu_4.snail, 12

kyu_4.snail.snail_sort, 11

kyu_4.snail.test_snail, 12

kyu_4.strings_mix, 15

kyu_4.strings_mix.solution, 15

kyu_4.strings_mix.test_mix, 15

kyu_4.strip_comments, 11

kyu_4.strip_comments.solution, 11

kyu_4.strip_comments.test_solution, 11

kyu_4.sudoku_solution_validator, 10

kyu_4.sudoku_solution_validator.test_valid_solution,

9

kyu_4.sudoku_solution_validator.valid_solution,

10

kyu_4.sum_by_factors, 13

kyu_4.sum_by_factors.sum_for_list, 12

kyu_4.sum_by_factors.test_sum_for_list,

12

kyu_4.sum_of_intervals, 7

kyu_4.sum_of_intervals.sum_of_intervals,

7

kyu_4.sum_of_intervals.test_sum_of_intervals,

7

kyu_4.the_greatest_warrior, 15

kyu_4.the_greatest_warrior.test_battle,

13

kyu_4.the_greatest_warrior.test_warrior, 14
 kyu_4.the_greatest_warrior.warrior, 14
 kyu_4.validate_sudoku_with_size, 11
 kyu_4.validate_sudoku_with_size.sudoku, 10
 kyu_4.validate_sudoku_with_size.test_sudoku, 11
 kyu_5, 36
 kyu_5.alphabet_wars_nuclear_strike, 21
 kyu_5.alphabet_wars_nuclear_strike.alphabet_war, 20
 kyu_5.alphabet_wars_nuclear_strike.test_alphabet_war, 21
 kyu_5.count_ip_addresses, 18
 kyu_5.count_ip_addresses.ips_between, 18
 kyu_5.count_ip_addresses.test_ips_between, 18
 kyu_5.did_i_finish_my_sudoku, 24
 kyu_5.did_i_finish_my_sudoku.is_sudoku_done, 23
 kyu_5.did_i_finish_my_sudoku.sudoku_by_column, 23
 kyu_5.did_i_finish_my_sudoku.sudoku_by_row, 23
 kyu_5.did_i_finish_my_sudoku.sudoku_by_row, 24
 kyu_5.did_i_finish_my_sudoku.test_did_i_finish_my_sudoku, 24
 kyu_5.directions_reduction, 23
 kyu_5.directions_reduction.directions_reduction, 22
 kyu_5.directions_reduction.test_directions_reduction, 23
 kyu_5.extract_the_domain_name_from_url, 32
 kyu_5.extract_the_domain_name_from_url.extract_domain_name_from_url, 32
 kyu_5.extract_the_domain_name_from_url.test_extract_domain_name_from_url, 32
 kyu_5.fibonacci_streaming, 18
 kyu_5.fibonacci_streaming.all_fibonacci_numbers, 17
 kyu_5.fibonacci_streaming.test_all_fibonacci_numbers, 17
 kyu_5.find_the_safest_places_in_town, 32
 kyu_5.find_the_safest_places_in_town.advice, 30
 kyu_5.find_the_safest_places_in_town.print_agents, 31
 kyu_5.find_the_safest_places_in_town.test_advice, 31
 kyu_5.first_non_repeating_character, 27
 kyu_5.first_non_repeating_character.first_non_repeating_character, 27
 kyu_5.first_non_repeating_character.test_first_non_repeating_character, 27
 kyu_5.flatten, 27
 kyu_5.flatten.flatten, 26
 kyu_5.flatten.test_flatten, 26
 kyu_5.human_readable_time, 20
 kyu_5.human_readable_time.make_readable, 20
 kyu_5.human_readable_time.test_make_readable, 20
 kyu_5.integers_recreation_one, 36
 kyu_5.integers_recreation_one.solution, 35
 kyu_5.integers_recreation_one.test_list_squared, 36
 kyu_5.master_your_primes_sieve_with_memoization, 25
 kyu_5.master_your_primes_sieve_with_memoization.primes, 25
 kyu_5.master_your_primes_sieve_with_memoization.test_primes, 25
 kyu_5.moving_zeros_to_the_end, 22
 kyu_5.moving_zeros_to_the_end.move_zeros, 22
 kyu_5.moving_zeros_to_the_end.test_move_zeros, 22
 kyu_5.not_very_secure, 19
 kyu_5.not_very_secure.alphanumeric, 18
 kyu_5.not_very_secure.test_alphanumeric, 19
 kyu_5.number_of_trailing_zeros_of_n, 26
 kyu_5.number_of_trailing_zeros_of_n.test_zeros, 25
 kyu_5.number_of_trailing_zeros_of_n.zeros, 25
 kyu_5.simple_pig_latin, 20
 kyu_5.simple_pig_latin.pig_it, 19
 kyu_5.simple_pig_latin.test_pig_it, 19
 kyu_5.sports_league_table_ranking, 30
 kyu_5.sports_league_table_ranking.compute_ranks, 27
 kyu_5.sports_league_table_ranking.test_compute_ranks, 29
 kyu_5.string_incrementer, 35
 kyu_5.string_incrementer.string_incrementer, 35
 kyu_5.string_incrementer.test_increment_string, 35
 kyu_5.sum_of_pairs, 34
 kyu_5.sum_of_pairs.sum_pairs, 33
 kyu_5.sum_of_pairs.test_sum_pairs, 33

kyu_5.the_hashtag_generator,33 58
kyu_5.the_hashtag_generator.hashtag_generator,32 kyu_6.default_list,54
kyu_5.the_hashtag_generator.test_generator,33 kyu_6.default_list.default_list,53
kyu_5.tic_tac_toe_checker,35 kyu_6.disease_spread,52
kyu_5.tic_tac_toe_checker.checker,34 kyu_6.disease_spread.epidemic,50
kyu_5.tic_tac_toe_checker.test_checker,34 kyu_6.disease_spread.epidemic_test_data,51
kyu_5.valid_parentheses,22 kyu_6.disease_spread.test_epidemic,52
kyu_5.valid_parentheses.test_valid_parentheses,21 kyu_6.duplicate_encoder,44
kyu_5.valid_parentheses.valid_parentheses,22 kyu_6.duplicate_encoder.duplicate_encode,43
kyu_5.where_my_anagrams_at,25 kyu_6.duplicate_encoder.test_duplicate_encode,43
kyu_5.where_my_anagrams_at.anagrams,24 kyu_6.easy_diagonal,55
kyu_5.where_my_anagrams_at.test_anagrams,24 kyu_6.easy_diagonal.diagonal,54
kyu_6,61 kyu_6.easy_diagonal.test_diagonal,54
kyu_6.a_rule_of_divisibility_by_13,52 kyu_6.encrypt_this,59
kyu_6.a_rule_of_divisibility_by_13.test_khir,59 kyu_6.encrypt_this.solution,59
kyu_6.a_rule_of_divisibility_by_13.third,52 kyu_6.encrypt_this.test_encrypt_this,59
kyu_6.array_diff,61 kyu_6.find_the_odd_int,37
kyu_6.array_diff.solution,61 kyu_6.find_the_odd_int.find_the_odd_int,37
kyu_6.array_diff.test_array_diff,61 kyu_6.find_the_odd_int.test_find_the_odd_int,37
kyu_6.array_to_html_table,55 kyu_6.first_character_that_repeats,38
kyu_6.array_to_html_table.to_table,55 kyu_6.first_character_that_repeats.first_character,37
kyu_6.binary_to_text_ascii_conversion,47 kyu_6.first_character_that_repeats.test_first_character,37
kyu_6.binary_to_text_ascii_conversion.binary_to_string,46 kyu_6.format_string_of_names,60
kyu_6.binary_to_text_ascii_conversion.test_kyubinfo,59 kyu_6.format_string_of_names.solution,59
kyu_6.casino_chips,47 kyu_6.format_string_of_names.test_namelist,60
kyu_6.casino_chips.solve,47 kyu_6.help_the_bookseller,49
kyu_6.casino_chips.test_solve,47 kyu_6.help_the_bookseller.stock_list,49
kyu_6.character_frequency,40 kyu_6.help_the_bookseller.test_stock_list,49
kyu_6.character_frequency.character_frequency,39 kyu_6.longest_repetition,39
kyu_6.character_frequency.test_character_frequency,39 kyu_6.longest_repetition.longest_repetition,38
kyu_6.color_choice,53 kyu_6.longest_repetition.test_longest_repetition,38
kyu_6.color_choice.checkchoose,52 kyu_6.multiples_of_3_or_5,46
kyu_6.color_choice.test_checkchoose,53 kyu_6.multiples_of_3_or_5.solution,45
kyu_6.count_letters_in_string,43 kyu_6.multiples_of_3_or_5.test_solution,45
kyu_6.count_letters_in_string.count_letters_in_string,42
kyu_6.count_letters_in_string.test_count_letters_in_string,42
kyu_6.decipher_this,59 kyu_6.number_zoo_patrol,56
kyu_6.decipher_this.solution,58 kyu_6.number_zoo_patrol.missing_number,56
kyu_6.decipher_this.test_decipher_this,56

kyu_6.number_zoo_patrol.test_find_missing_number, 56
 kyu_6.numericals_of_string, 39
 kyu_6.numericals_of_string.numericals, 39
 kyu_6.numericals_of_string.test_numericals, 39
 kyu_6.permute_a_palindrome, 42
 kyu_6.permute_a_palindrome.permute_a_palindrome, 42
 kyu_6.permute_a_palindrome.test_permute_a_palindrome, 42
 kyu_6.pokemon_damage_calculator, 49
 kyu_6.pokemon_damage_calculator.calculate_damage, 48
 kyu_6.pokemon_damage_calculator.test_calculate_damage, 48
 kyu_6.potion_class_101, 50
 kyu_6.potion_class_101.potion, 50
 kyu_6.potion_class_101.test_potion, 50
 kyu_6.pyramid_array, 38
 kyu_6.pyramid_array.pyramid_array, 38
 kyu_6.pyramid_array.test_pyramid_array, 38
 kyu_6.rotate_the_letters_of_each_element, 56
 kyu_6.rotate_the_letters_of_each_element.rotate_letters, 55
 kyu_6.rotate_the_letters_of_each_element.test_rotate_letters, 56
 kyu_6.row_of_the_odd_triangle, 50
 kyu_6.row_of_the_odd_triangle.odd_row, 49
 kyu_6.row_of_the_odd_triangle.test_odd_row, 50
 kyu_6.sort_the_odd, 61
 kyu_6.sort_the_odd.solution, 60
 kyu_6.sort_the_odd.test_sort_array, 60
 kyu_6.string_subpattern_recognition_1, 40
 kyu_6.string_subpattern_recognition_1.has_subpattern, 40
 kyu_6.string_subpattern_recognition_1.test_has_subpattern, 40
 kyu_6.string_subpattern_recognition_2, 41
 kyu_6.string_subpattern_recognition_2.has_subpattern, 40
 kyu_6.string_subpattern_recognition_2.test_has_subpattern, 41
 kyu_6.string_subpattern_recognition_3, 42
 kyu_6.string_subpattern_recognition_3.has_subpattern, 41
 kyu_6.string_subpattern_recognition_3.test_has_subpattern, 41
 kyu_6.string_transformer, 45
 kyu_6.string_transformer.string_transformer, 44
 kyu_6.string_transformer.test_string_transformer, 45
 kyu_6.sum_of_digits_digital_root, 46
 kyu_6.sum_of_digits_digital_root.digital_root, 46
 kyu_6.sum_of_digits_digital_root.test_digital_root, 46
 kyu_6.unique_in_order, 43
 kyu_6.unique_in_order.test_unique_in_order, 43
 kyu_6.unique_in_order.unique_in_order, 43
 kyu_6.vasya_clerk, 44
 kyu_6.vasya_clerk.test_tickets, 44
 kyu_6.vasya_clerk.tickets, 44
 kyu_6.who_likes_it, 58
 kyu_6.who_likes_it.likes_function, 57
 kyu_6.who_likes_it.test_likes_function, 58
 kyu_6.your_order_please, 57
 kyu_6.your_order_please.order, 56
 kyu_6.your_order_please.test_order, 57
 kyu_7, 83
 kyu_7.always_perfect, 76
 kyu_7.always_perfect.check_root, 75
 kyu_7.always_perfect.test_check_root, 76
 kyu_7.basic_math_add_or_subtract, 81
 kyu_7.basic_math_add_or_subtract.calculate, 80
 kyu_7.basic_math_add_or_subtract.test_calculate, 81
 kyu_7.beginner_series_sum_of_numbers, 62
 kyu_7.beginner_series_sum_of_numbers.sum_of_numbers, 61
 kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers, 62
 kyu_7.disemvowel_trolls, 63
 kyu_7.disemvowel_trolls.disemvowel_trolls, 62
 kyu_7.disemvowel_trolls.test_disemvowel_trolls, 62
 kyu_7.easy_line, 83
 kyu_7.easy_line.easyline, 82
 kyu_7.easy_line.test_easyline, 82
 kyu_7.factorial, 79
 kyu_7.factorial.factorial, 78
 kyu_7.factorial.test_factorial, 78

kyu_7.fill_the_hard_disk_drive, 72
 kyu_7.fill_the_hard_disk_drive.save, 71
 kyu_7.fill_the_hard_disk_drive.test_save, 72
 kyu_7.find_the_longest_gap, 79
 kyu_7.find_the_longest_gap.gap, 79
 kyu_7.find_the_longest_gap.test_gap, 79
 kyu_7.formatting_decimal_places_1, 77
 kyu_7.formatting_decimal_places_1.test_two_decimal_places, 76
 kyu_7.formatting_decimal_places_1.two_decimal_places, 76
 kyu_7.fun_with_lists_length, 71
 kyu_7.fun_with_lists_length.length, 70
 kyu_7.fun_with_lists_length.node, 71
 kyu_7.fun_with_lists_length.test_length, 71
 kyu_7.growing_plant, 80
 kyu_7.growing_plant.growing_plant, 79
 kyu_7.growing_plant.test_growing_plant, 80
 kyu_7.help_bob_count_letters_and_digits, 82
 kyu_7.help_bob_count_letters_and_digits.kyu_7_letters_and_digits, 81
 kyu_7.help_bob_count_letters_and_digits.kyu_7_letters_and_digits.test_letters_and_digits, 82
 kyu_7.isograms, 83
 kyu_7.isograms.is_isogram, 83
 kyu_7.isograms.test_is_isogram, 83
 kyu_7.jaden_casing_strings, 63
 kyu_7.jaden_casing_strings.jaden_casing_strings, 63
 kyu_7.jaden_casing_strings.test_jaden_casing_strings, 63
 kyu_7.make_class, 74
 kyu_7.make_class.animal, 73
 kyu_7.make_class.make_class, 74
 kyu_7.make_class.test_make_class, 74
 kyu_7.maximum_multiple, 73
 kyu_7.maximum_multiple.maximum_multiple, 73
 kyu_7.password_validator, 75
 kyu_7.password_validator.password, 74
 kyu_7.password_validator.test_password, 74
 kyu_7.powers_of_3, 67
 kyu_7.powers_of_3.largest_power, 67
 kyu_7.powers_of_3.test_largest_power, 67
 kyu_7.pull_your_words_together_man, 78
 kyu_7.pull_your_words_together_man.sentencify, 77
 kyu_7.pull_your_words_together_man.test_sentencify, 78
 kyu_7.remove_the_minimum, 65
 kyu_7.remove_the_minimum.remove_the_minimum, 63
 kyu_7.remove_the_minimum.test_remove_the_minimum, 64
 kyu_7.share_prices, 75
 kyu_7.share_prices.share_price, 75
 kyu_7.share_prices.test_share_price, 75
 kyu_7.significant_figures, 70
 kyu_7.significant_figures.number_of_sigfigs, 69
 kyu_7.significant_figures.test_number_of_sigfigs, 69
 kyu_7.simple_fun_152, 69
 kyu_7.simple_fun_152.invite_more_women, 68
 kyu_7.simple_fun_152.test_invite_more_women, 69
 kyu_7.sort_out_the_men_from_boys, 70
 kyu_7.sort_out_the_men_from_boys.men_from_boys, 70
 kyu_7.sort_out_the_men_from_boys.test_men_from_boys, 70
 kyu_7.substituting_variables_into_strings_padded_n, 77
 kyu_7.substituting_variables_into_strings_padded_n, 77
 kyu_7.substituting_variables_into_strings_padded_n, 77
 kyu_7.sum_of_odd_numbers, 81
 kyu_7.sum_of_odd_numbers.row_sum_odd_numbers, 81
 kyu_7.sum_of_odd_numbers.test_row_sum_odd_numbers, 81
 kyu_7.sum_of_powers_of_2, 67
 kyu_7.sum_of_powers_of_2.sum_of_powers_of_2, 66
 kyu_7.sum_of_powers_of_2.test_sum_of_powers_of_2, 67
 kyu_7.sum_of_triangular_numbers, 68
 kyu_7.sum_of_triangular_numbers.sum_triangular_numbers, 67
 kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers, 68
 kyu_7.sum_of_two_lowest_int, 66
 kyu_7.sum_of_two_lowest_int.sum_two_smallest_int, 65
 kyu_7.sum_of_two_lowest_int.test_sum_two_smallest_int, 65
 kyu_7.the_first_non_repeated_character_in_string, 73
 kyu_7.the_first_non_repeated_character_in_string, 73

kyu_7.the_first_non_repeated_character_in_string, 72	kyu_8.grasshopper_check_for_factor.test_check_for_factor, 97
kyu_7.vaporcode, 68	kyu_8.grasshopper_messi_goals_function, 87
kyu_7.vaporcode.test_vaporcode, 68	kyu_8.grasshopper_messi_goals_function.messi_goals, 86
kyu_7.vaporcode.vaporcode, 68	kyu_8.grasshopper_messi_goals_function.test_messi_goals, 87
kyu_7.you_are_square, 66	kyu_8.grasshopper_personalized_message, 86
kyu_7.you_are_square.test_you_are_square, 66	kyu_8.grasshopper_personalized_message.grasshopper, 86
kyu_8.alternating_case, 93	kyu_8.grasshopper_personalized_message.test_grasshopper, 86
kyu_8.alternating_case.alternating_case, 92	kyu_8.grasshopper_summation, 93
kyu_8.alternating_case.test_alternating_case, 93	kyu_8.grasshopper_summation.summation, 93
kyu_8.century_from_year, 104	kyu_8.grasshopper_summation.test_summation, 93
kyu_8.century_from_year.century, 103	kyu_8.greek_sort, 105
kyu_8.century_from_year.test_century, 104	kyu_8.greek_sort.greek_comparator, 105
kyu_8.check_the_exam, 98	kyu_8.greek_sort.test_greek_comparator, 105
kyu_8.check_the_exam.check_exam, 98	kyu_8.holiday_vi_shark_pontoon, 105
kyu_8.check_the_exam.test_check_exam, 98	kyu_8.holiday_vi_shark_pontoon.shark, 104
kyu_8.convert_string_to_an_array, 100	kyu_8.holiday_vi_shark_pontoon.test_shark, 104
kyu_8.convert_string_to_an_array.string_to_array, 99	kyu_8.is_it_a_palindrome, 99
kyu_8.convert_string_to_an_array.test_string_to_array, 100	kyu_8.is_it_a_palindrome.is_palindrome, 98
kyu_8.count_the_monkeys, 101	kyu_8.is_it_a_palindrome.test_is_palindrome, 99
kyu_8.count_the_monkeys.monkey_count, 101	kyu_8.is_your_period_late, 84
kyu_8.count_the_monkeys.test_monkey_count, 101	kyu_8.is_your_period_late.is_your_period_late, 83
kyu_8.counting_sheep, 97	kyu_8.is_your_period_late.test_is_your_period_late, 84
kyu_8.counting_sheep.counting_sheep, 96	kyu_8.keep_hydrated, 95
kyu_8.counting_sheep.test_counting_sheep, 97	kyu_8.keep_hydrated.keep_hydrated, 94
kyu_8.enumerable_magic_25, 103	kyu_8.keep_hydrated.test_keep_hydrated, 95
kyu_8.enumerable_magic_25.take, 102	kyu_8.keep_up_the_hoop, 102
kyu_8.enumerable_magic_25.test_take, 102	kyu_8.keep_up_the_hoop.hoop_count, 101
kyu_8.find_the_first_non_consecutive_numbers, 90	kyu_8.keep_up_the_hoop.test_hoop_count, 102
kyu_8.find_the_first_non_consecutive_numbers, 90	kyu_8.logical_calculator, 85
kyu_8.find_the_first_non_consecutive_numbers, 90	kyu_8.logical_calculator.logical_calculator, 84
kyu_8.formatting_decimal_places_0, 99	kyu_8.logical_calculator.test_logical_calculator, 84
kyu_8.formatting_decimal_places_0.test_formatting_decimal_places, 99	kyu_8.make_upper_case, 88
kyu_8.formatting_decimal_places_0.two_decimal_places, 99	kyu_8.make_upper_case.make_upper_case, 88
kyu_8.grasshopper_check_for_factor, 98	
kyu_8.grasshopper_check_for_factor.check_for_factor, 98	

[kyu_8.well_of_ideas_easy_version, 88](#)
[kyu_8.well_of_ideas_easy_version.test_well_of_ideas, 87](#)
[kyu_8.well_of_ideas_easy_version.well_of_ideas_easy, 88](#)
[kyu_8.will_there_be_enough_space, 96](#)
[kyu_8.will_there_be_enough_space.enough, 96](#)
[kyu_8.will_there_be_enough_space.test_enough, 96](#)
[kyu_8.will_you_make_it, 103](#)
[kyu_8.will_you_make_it.test_zero_fuel, 103](#)
[kyu_8.will_you_make_it.zero_fuel, 103](#)
[kyu_8.wolf_in_sheep_clothing, 90](#)
[kyu_8.wolf_in_sheep_clothing.test_wolf_in_sheep_clothing, 89](#)
[kyu_8.wolf_in_sheep_clothing.wolf_in_sheep_clothing, 89](#)
[U, 106](#)
[utils, 106](#)
[utils.log_func, 106](#)
[utils.primes, 106](#)
[utils.primes.is_prime, 105](#)
[utils.primes.primes_generator, 106](#)
[utils.primes.test_is_prime, 106](#)
[utils.primes.test_primes_generator, 106](#)
[kyu_8.set_alarm, 96](#)
[kyu_8.set_alarm.set_alarm, 95](#)
[kyu_8.set_alarm.test_set_alarm, 95](#)
[kyu_8.surface_area_and_volume_of_box, 92](#)
[kyu_8.surface_area_and_volume_of_box.get_size, 92](#)
[kyu_8.surface_area_and_volume_of_box.test_get_size, 92](#)
[kyu_8.swap_values, 94](#)
[kyu_8.swap_values.swap_values, 94](#)
[kyu_8.swap_values.test_swap_values, 94](#)
[kyu_8.terminal_game_move_function, 89](#)
[kyu_8.terminal_game_move_function.terminal_game_move_function, 88](#)
[kyu_8.terminal_game_move_function.test_terminal_game_move_function, 89](#)
[kyu_8.the_feast_of_many_beasts, 101](#)
[kyu_8.the_feast_of_many_beasts.feast, 100](#)
[kyu_8.the_feast_of_many_beasts.test_feast, 100](#)
[kyu_8.third_angle_of_triangle, 91](#)
[kyu_8.third_angle_of_triangle.test_third_angle_of_triangle, 90](#)
[kyu_8.third_angle_of_triangle.third_angle_of_triangle, 91](#)

INDEX

Symbols

<code>__calculate()</code> (<i>kyu_3.calculator.calculator.Calculator</i> method), 2	<code>AlternatingCaseTestCase</code> (class in <i>kyu_8.alternating_case.test_alternating_case</i>), 93
<code>__process_math_expression()</code> (<i>kyu_3.calculator.calculator.Calculator</i> method), 2	<code>anagrams()</code> (in module <i>kyu_5.where_my_anagrams_at.anagrams</i>), 24
<code>__set_level()</code> (<i>kyu_4.the_greatest_warrior.warrior.Warrior</i> method), 14	<code>AnagramsTestCase</code> (class in <i>kyu_5.where_my_anagrams_at.test_anagrams</i>), 24
<code>__set_rank()</code> (<i>kyu_4.the_greatest_warrior.warrior.Warrior</i> method), 14	<code>Animal</code> (class in <i>kyu_7.make_class.animal</i>), 73
<code>__update_experience()</code> (<i>kyu_4.the_greatest_warrior.warrior.Warrior</i> method), 14	<code>append()</code> (<i>kyu_6.default_list.default_list.DefaultList</i> method), 53
	<code>array_diff()</code> (in module <i>kyu_6.array_diff.solution</i>), 61
A	<code>ArrayDiffTestCase</code> (class in <i>kyu_6.array_diff.test_array_diff</i>), 61
<code>a()</code> (<i>kyu_6.disease_spread.epidemic_test_data.EpidemicTestData</i> property), 51	<code>assert_sudoku_by_column()</code> (in module <i>kyu_5.did_i_finish_my_sudoku.sudoku_by_column</i>), 23
<code>achievements()</code> (<i>kyu_4.the_greatest_warrior.warrior.Warrior</i> property), 14	<code>assert_sudoku_by_region()</code> (in module <i>kyu_5.did_i_finish_my_sudoku.sudoku_by_regions</i>), 23
<code>advice()</code> (in module <i>kyu_5.find_the_safest_places_in_town.advice</i>), 30	<code>assert_sudoku_by_row()</code> (in module <i>kyu_5.did_i_finish_my_sudoku.sudoku_by_row</i>), 24
<code>agents_cleanup()</code> (in module <i>kyu_5.find_the_safest_places_in_town.advice</i>), 30	
<code>all_fibonacci_numbers()</code> (in module <i>kyu_5.fibonacci_streaming.all_fibonacci_numbers</i>), 17	B
<code>AllFibonacciNumbersTestCase</code> (class in <i>kyu_5.fibonacci_streaming.test_all_fibonacci_numbers</i>), 17	<code>b()</code> (<i>kyu_6.disease_spread.epidemic_test_data.EpidemicTestData</i> property), 51
<code>alphabet_war()</code> (in module <i>kyu_5.alphabet_wars_nuclear_strike.alphabet_war</i>), 20	<code>battle()</code> (<i>kyu_4.the_greatest_warrior.warrior.Warrior</i> method), 14
<code>AlphabetWarTestCase</code> (class in <i>kyu_5.alphabet_wars_nuclear_strike.test_alphabet_war</i>), 21	<code>BattleshipFieldValidatorTestCase</code> (class in <i>kyu_3.battleship_field_validator.test_battleship_validator</i>), 6
<code>alphanumeric()</code> (in module <i>kyu_5.not_very_secure.alphanumeric</i>), 18	<code>BattleTestCase</code> (class in <i>kyu_4.the_greatest_warrior.test_battle</i>), 13
<code>AlphanumericTestCase</code> (class in <i>kyu_5.not_very_secure.test_alphanumeric</i>), 19	<code>binary_to_string()</code> (in module <i>kyu_6.binary_to_text_ascii_conversion.binary_to_string</i>), 46
	C
	<code>calc()</code> (in module <i>kyu_2.evaluate_mathematical_expression.evaluate</i>), 1

```

calc_combination_per_row_item() (in module kyu_7.easy_line.easyline), 82
CalcTestCase (class in kyu_2.evaluate_mathematical_expression.test_evaluate),
calc_days() (in module kyu_4.human_readable_duration_format.format_duration), 2
7
calculate() (in module kyu_2.evaluate_mathematical_expression.evaluate),
calc_first_number() (in module kyu_6.row_of_the_odd_triangle.odd_row), 1
49
calculate() (in module kyu_7.basic_math_add_or_subtract.calculate),
calc_first_number() (in module kyu_7.sum_of_odd_numbers.row_sum_odd_numbers), 80
81
calculate_damage() (in module kyu_6.pokemon_damage_calculator.calculate_damage),
calc_for_against() (in module kyu_5.sports_league_table_ranking.compute_ranks), 48
CalculateDamageTestCase (class in kyu_6.pokemon_damage_calculator.test_calculate_damage),
27
calc_g_cur() (in module kyu_7.find_the_longest_gap.gap), 79
48
CalculateTestCase (class in kyu_7.basic_math_add_or_subtract.test_calculate),
calc_g_max() (in module kyu_7.find_the_longest_gap.gap), 79
81
calc_gd() (in module kyu_5.sports_league_table_ranking.compute_ranks), CalculateTestCase (class in kyu_7.help_bob_count_letters_and_digits.test_count_letters_and_digits),
28
82
calc_hours() (in module kyu_4.human_readable_duration_format.format_duration), Calculator (class in kyu_3.calculator.calculator), 2
8
CalculatorTestCase (class in kyu_3.calculator.test_calculator), 3
calc_ip_range() (in module kyu_5.count_ip_addresses.ips_between), century() (in module kyu_8.century_from_year.century), 103
18
CenturyTestCase (class in kyu_8.century_from_year.test_century), 104
calc_last_number() (in module kyu_6.row_of_the_odd_triangle.odd_row), char_processor() (in module kyu_8.check_the_exam.check_exam), 98
49
calc_last_number() (in module kyu_7.sum_of_odd_numbers.row_sum_odd_numbers), check_cols() (in module kyu_5.tic_tac_toe_checker.checker), 34
81
check_diagonals() (in module kyu_5.tic_tac_toe_checker.checker), 34
calc_minutes() (in module kyu_4.human_readable_duration_format.format_duration), exam() (in module kyu_8.check_the_exam.check_exam), 98
8
calc_rank() (in module kyu_5.sports_league_table_ranking.compute_ranks), check_for_factor() (in module kyu_8.grasshopper_check_for_factor.check_for_factor),
28
97
calc_result() (in module kyu_5.count_ip_addresses.ips_between), check_root() (in module kyu_7.always_perfect.check_root), 75
18
check_rows() (in module kyu_5.tic_tac_toe_checker.checker), 34
calc_seconds() (in module kyu_4.human_readable_duration_format.format_duration), choose() (in module kyu_6.color_choice.checkchoose), 52
8
CheckchooseTestCase (class in kyu_6.color_choice.test_checkchoose), 53
calc_team_points() (in module kyu_5.sports_league_table_ranking.compute_ranks), CheckExamTestCase (class in kyu_8.check_the_exam.test_check_exam),
28
98
calc_teams_score() (in module kyu_5.sports_league_table_ranking.compute_ranks), CheckForFactorTestCase (class in kyu_8.grasshopper_check_for_factor.test_check_for_factor),
28
calc_years() (in module kyu_4.human_readable_duration_format.format_duration), 97
8
CheckRootTestCase (class in

```

kyu_7.always_perfect.test_check_root), 76
city_map_processing() (in module kyu_5.find_the_safest_places_in_town.advice), 30
clean_battlefield() (in module kyu_5.alphabet_wars_nuclear_strike.alphabet_war), 20
clean_interval() (in module kyu_4.sum_of_intervals.sum_of_intervals), 7
clean_unsheltered() (in module kyu_5.alphabet_wars_nuclear_strike.alphabet_war), 20
clean_up_string() (in module kyu_5.valid_parentheses.valid_parentheses), 22
color() (kyu_6.potion_class_101.potion.Potion property), 50
compute_ranks() (in module kyu_5.sports_league_table_ranking.compute_ranks), 28
ComputeRanksTestCase (class in kyu_5.sports_league_table_ranking.test_compute_ranks), 29
count_letters_and_digits() (in module kyu_7.help_bob_count_letters_and_digits.count_letters_and_digits), 81
count_sheeps() (in module kyu_8.counting_sheep.counting_sheep), 96
CountingSheepTestCase (class in kyu_8.counting_sheep.test_counting_sheep), 97
CountLettersInStringTestCase (class in kyu_6.count_letters_in_string.test_count_letters_in_string), 42
create_city_map() (in module kyu_5.find_the_safest_places_in_town.advice), 30
D
decipher_this() (in module kyu_6.decipher_this.solution), 58
DecipherThisTestCase (class in kyu_6.decipher_this.test_decipher_this), 58
decode_rail_fence_cipher() (in module kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding), 3
DecodingTestCase (class in kyu_3.rail_fence_cipher_encoding_and_decoding.test_decoding), 4
DefaultList (class in kyu_6.default_list.default_list), 53
DefaultListTestCase (class in kyu_6.default_list.test_default_list), 53
diagonal() (in module kyu_6.easy_diagonal.diagonal), 54
DidIFinishedSudokuTestCase (class in kyu_5.did_i_finish_my_sudoku.test_did_i_finish_sudoku), 24
digit_that_breaks_ordering_index() (in module kyu_4.next_bigger_number_with_the_same_digits.next_bigger_number_with_the_same_digits), 16
digital_root() (in module kyu_5.integers_recreation_one.solution), 35
digital_root() (in module kyu_6.sum_of_digits_digital_root.digital_root), 46
DigitalRootTestCase (class in kyu_6.sum_of_digits_digital_root.test_digital_root), 46
DirectionsReductionTestCase (class in kyu_5.directions_reduction.test_directions_reduction), 23
dirReduc() (in module kyu_5.directions_reduction.directions_reduction), 22
disemvowel() (in module kyu_7.disemvowel_trolls.disemvowel_trolls), 62
DisemvowelTestCase (class in kyu_7.disemvowel_trolls.test_disemvowel_trolls), 62
divisor_generator() (in module kyu_5.integers_recreation_one.solution), 35
domain_name() (in module kyu_5.extract_the_domain_name_from_url.extract_domain_from_url), 32
DomainNameTestCase (class in kyu_5.extract_the_domain_name_from_url.test_domain_name), 32
done_or_not() (in module kyu_5.did_i_finish_my_sudoku.is_sudoku_done), 23
down() (in module kyu_3.make_spiral.solution), 4
duplicate_encode() (in module kyu_6.duplicate_encoder.duplicate_encode), 43
DuplicateEncodeTestCase (class in kyu_6.duplicate_encoder.test_duplicate_encode), 43
E
easy_line() (in module kyu_7.easy_line.easylines), 82
EasyDiagonalTestCase (class in kyu_6.easy_diagonal.test_diagonal), 54

```

EasyLineTestCase (class in 15
    kyu_7.easy_line.test_easyline), 82
effectiveness() (in module kyu_4.next_smaller_number_with_the_same_digits.next_smaller)
    kyu_6.pokemon_damage_calculator.calculate_damage), 15
48
FindMissingNumberTestCase (class in
encode_rail_fence_cipher() (in module kyu_6.number_zoo_patrol.test_find_missing_number),
    kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding),
3
FindTheOddIntTestCase (class in
EncodingTestCase (class in kyu_6.find_the_odd_int.test_find_the_odd_int),
    kyu_3.rail_fence_cipher_encoding_and_decoding.test_encoding),
4
first_dup() (in module
encrypt_this() (in module kyu_6.first_character_that_repeats.first_character_that_repeats),
    kyu_6.encrypt_this.solution), 59
37
EncryptThisTestCase (class in first_non_consecutive() (in module
    kyu_6.encrypt_this.test_encrypt_this), 59 kyu_8.find_the_first_non_consecutive_number.first_non_consecutive)
enough() (in module 90
    kyu_8.will_there_be_enough_space.enough), first_non_repeated() (in module
    96 kyu_7.the_first_non_repeated_character_in_string.first_non_repeated_character_in_string)
EnoughTestCase (class in 72
    kyu_8.will_there_be_enough_space.test_enough), first_non_repeating_letter() (in module
    96 kyu_5.first_non_repeating_character.first_non_repeating_letter),
epidemic() (in module 27
    kyu_6.disease_spread.epidemic), 50
EpidemicTestCase (class in kyu_5.find_the_safest_places_in_town.test_advice),
    kyu_6.disease_spread.test_epidemic), 52 31
EpidemicTestData (class in FirstDupTestCase (class in
    kyu_6.disease_spread.epidemic_test_data), kyu_6.first_character_that_repeats.test_first_character_that_repeats)
51 37
evaluate() (kyu_3.calculator.calculator.Calculator FirstNonConsecutiveTestCase (class in
    method), 2 kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive)
expected() (kyu_6.disease_spread.epidemic_test_data.EpidemicTestData
    property), 51 FirstNonRepeatedTestCase (class in
experience() (kyu_4.the_greatest_warrior.warrior.Warrior kyu_7.the_first_non_repeated_character_in_string.test_first_non_repeated_character_in_string)
    property), 14 72
extend() (kyu_6.default_list.default_list.DefaultList FirstNonRepeatingLetterTestCase (class in
    method), 53 kyu_5.first_non_repeating_character.test_first_non_repeating_character)
27
F
fix_the_meerkat() (in module
factorial() (in module kyu_7.factorial.factorial), 78 kyu_8.my_head_is_at_the_wrong_end.fix_the_meerkat),
    93
FactorialTestCase (class in
    kyu_7.factorial.test_factorial), 78
FixTheMeerkatTestCase (class in
feast() (in module kyu_8.my_head_is_at_the_wrong_end.test_fix_the_meerkat),
    kyu_8.the_feast_of_many_beasts.feast), 100 94
FeastTestCase (class in flatten() (in module kyu_5.flatten.flatten), 26
    kyu_8.the_feast_of_many_beasts.test_feast), FlattenTestCase (class in
    100 kyu_5.flatten.test_flatten), 26
find_it() (in module format_duration() (in module
    kyu_6.find_the_odd_int.find_the_odd_int), kyu_4.human_readable_duration_format.format_duration),
    37 8
find_missing_number() (in module FormatDurationTestCase (class in
    kyu_6.number_zoo_patrol.missing_number), kyu_4.human_readable_duration_format.test_format_duration),
    56 9
find_x() (in module
    kyu_4.next_smaller_number_with_the_same_digits.next_smaller),

```

G

gap () (in module kyu_7.find_the_longest_gap.gap), 79
 GapTestCase (class in kyu_7.find_the_longest_gap.test_gap), 79
 gen_primes () (in module utils.primes.primes_generator), 106
 generate_hashtag () (in module kyu_5.the_hashtag_generator.hashtag_generator), 32

GenerateHashtagTestCase (class in kyu_5.the_hashtag_generator.test_generate_hashtag), 33
 GenPrimesTestCase (class in utils.primes.test_primes_generator), 106
 get_counters () (in module kyu_4.strings_mix.solution), 15
 get_first_digit_index () (in module kyu_5.string_incrementer.string_incrementer), 35

get_rails () (in module kyu_3.rail_fence_cipher_encoding_and_decoding.encoding_and_decoding), 4
 get_size () (in module kyu_8.surface_area_and_volume_of_box.get_size), 92

get_string () (in module kyu_4.human_readable_duration_format.format_duration), 8

get_sum () (in module kyu_7.beginner_series_sum_of_numbers.sum_of_numbers), 61

GetSizeTestCase (class in kyu_8.surface_area_and_volume_of_box.test_get_size), 92

goals () (in module kyu_8.grasshopper_messi_goals_function.messi_goals_function), 86

GoalsTestCase (class in kyu_8.grasshopper_messi_goals_function.test_messi_goals_function), 87

greek_comparator () (in module kyu_8.greek_sort.greek_comparator), 105

GreekComparatorTestCase (class in kyu_8.greek_sort.test_greek_comparator), 105

greet () (in module kyu_8.grasshopper_personalized_message.grasshopper_personalized_message), 86

GreetTestCase (class in kyu_8.grasshopper_personalized_message.test_grasshopper_personalized_message), 86

group_cities () (in module kyu_6.rotate_the_letters_of_each_element.group_cities), 55

GroupCitiesTestCase (class in kyu_6.rotate_the_letters_of_each_element.test_group_cities), 56

growing_plant () (in module kyu_7.growing_plant.growing_plant), 79

GrowingPlantTestCase (class in kyu_7.growing_plant.test_growing_plant), 80

H

has_subpattern () (in module kyu_6.string_subpattern_recognition_1.has_subpattern), 40

has_subpattern () (in module kyu_6.string_subpattern_recognition_2.has_subpattern), 40

has_subpattern () (in module kyu_6.string_subpattern_recognition_3.has_subpattern), 41

HasSubpatternTestCase (class in kyu_6.string_subpattern_recognition_1.test_has_subpattern), 40

HasSubpatternTestCase (class in kyu_6.string_subpattern_recognition_2.test_has_subpattern), 41

HasSubpatternTestCase (class in kyu_6.string_subpattern_recognition_3.test_has_subpattern), 41

hoop_count () (in module kyu_8.keep_up_the_hoop.hoop_count), 101

HoopCountTestCase (class in kyu_8.keep_up_the_hoop.test_hoop_count), 102

I

img (in module kyu_6.disease_spread.epidemic_test_data.EpidemicTestData property), 51

increment_string () (in module kyu_5.string_incrementer.string_incrementer), 35

insert () (in module kyu_6.default_list.default_list.DefaultList method), 53

invite_more_women () (in module kyu_7.simple_fun_152.invite_more_women), 68

InviteMoreWomenTestCase (class in kyu_7.simple_fun_152.test_invite_more_women), 69

ips_between () (in module kyu_5.count_ip_addresses.ips_between), 18

```
IpsBetweenTestCase          (class      in      module,7
    kyu_5.count_ip_addresses.test_ips_between), kyu_3.battleship_field_validator
18                                module,7
is_isogram()                (in          module kyu_3.battleship_field_validator.test_battleship_v
    kyu_7.isograms.is_isogram), 83                                module,6
is_palindrome()             (in          module kyu_3.battleship_field_validator.validator
    kyu_8.is_it_a_palindrome.is_palindrome), module,6
98                                kyu_3.calculator
is_perfect_square()         (in          module module,3
    kyu_5.integers_recreation_one.solution), kyu_3.calculator.calculator
36                                module,2
is_prime()                  (in          module kyu_3.calculator.test_calculator
    kyu_5.master_your_primes_sieve_with_memoization.primes), module,3
25                                kyu_3.make_spiral
is_prime() (in module utils.primes.is_prime), 105 module,6
is_solved()                 (in          module kyu_3.make_spiral.solution
    kyu_5.tic_tac_toe_checker.checker), 34 module,4
is_square()                 (in          module kyu_3.make_spiral.test_spiralize
    kyu_7.you_are_square.you_are_square), module,5
66                                kyu_3.rail_fence_cipher_encoding_and_decoding
is_valid() (kyu_4.validate_sudoku_with_size.sudoku.Sudoku module,4
    method), 10                                kyu_3.rail_fence_cipher_encoding_and_decoding.encod
is_valid_cell()             (in          module module,3
    kyu_3.battleship_field_validator.validator), kyu_3.rail_fence_cipher_encoding_and_decoding.test
6                                module,4
IsIsogramTestCase          (class      in      kyu_3.rail_fence_cipher_encoding_and_decoding.test
    kyu_7.isograms.test_is_isogram), 83 module,4
IsPalindromeTestCase       (class      in      kyu_4
    kyu_8.is_it_a_palindrome.test_is_palindrome), module,17
99                                kyu_4.human_readable_duration_format
IsPrimeTestCase            (class      in      module,9
    utils.primes.test_is_prime), 106 kyu_4.human_readable_duration_format.format_duratio
IsSolvedTestCase           (class      in      module,7
    kyu_5.tic_tac_toe_checker.test_checker), kyu_4.human_readable_duration_format.test_format_du
34                                module,9
J                                kyu_4.most_frequently_used_words
                                module,13
JadenCasingStringsTestCase (class      in      kyu_4.most_frequently_used_words.solution
    kyu_7.jaden_casing_strings.test_jaden_casing_strings), module,13
63                                kyu_4.most_frequently_used_words.test_top_3_words
                                module,13
K                                kyu_4.next_bigger_number_with_the_same_digits
                                module,17
KeepHydratedTestCase       (class      in      kyu_4.next_bigger_number_with_the_same_digits.next
    kyu_8.keep_hydrated.test_keep_hydrated), module,16
95                                kyu_4.next_bigger_number_with_the_same_digits.test
kyu_2                        module,17
    module,2                                kyu_4.next_smaller_number_with_the_same_digits
kyu_2.evaluate_mathematical_expression module,16
    module,2                                kyu_4.next_smaller_number_with_the_same_digits.next
kyu_2.evaluate_mathematical_expression.evaluate module,15
    module,1                                kyu_4.next_smaller_number_with_the_same_digits.test
kyu_2.evaluate_mathematical_expression.test module,16
    module,2                                kyu_4.range_extraction
kyu_3
```


module, 10	module, 11
kyu_4.range_extraction.solution	kyu_5
module, 10	module, 36
kyu_4.range_extraction.test_solution	kyu_5.alphabet_wars_nuclear_strike
module, 10	module, 21
kyu_4.snail	kyu_5.alphabet_wars_nuclear_strike.alphabet_war
module, 12	module, 20
kyu_4.snail.snail_sort	kyu_5.alphabet_wars_nuclear_strike.test_alphabet_war
module, 11	module, 21
kyu_4.snail.test_snail	kyu_5.count_ip_addresses
module, 12	module, 18
kyu_4.strings_mix	kyu_5.count_ip_addresses.ips_between
module, 15	module, 18
kyu_4.strings_mix.solution	kyu_5.count_ip_addresses.test_ips_between
module, 15	module, 18
kyu_4.strings_mix.test_mix	kyu_5.did_i_finish_my_sudoku
module, 15	module, 24
kyu_4.strip_comments	kyu_5.did_i_finish_my_sudoku.is_sudoku_done
module, 11	module, 23
kyu_4.strip_comments.solution	kyu_5.did_i_finish_my_sudoku.sudoku_by_column
module, 11	module, 23
kyu_4.strip_comments.test_solution	kyu_5.did_i_finish_my_sudoku.sudoku_by_regions
module, 11	module, 23
kyu_4.sudoku_solution_validator	kyu_5.did_i_finish_my_sudoku.sudoku_by_row
module, 10	module, 24
kyu_4.sudoku_solution_validator.test_validator	kyu_5.did_i_finish_my_sudoku.test_did_i_finish_sudoku
module, 9	module, 24
kyu_4.sudoku_solution_validator.valid_solution	kyu_5.directions_reduction
module, 10	module, 23
kyu_4.sum_by_factors	kyu_5.directions_reduction.directions_reduction
module, 13	module, 22
kyu_4.sum_by_factors.sum_for_list	kyu_5.directions_reduction.test_directions_reduction
module, 12	module, 23
kyu_4.sum_by_factors.test_sum_for_list	kyu_5.extract_the_domain_name_from_url
module, 12	module, 32
kyu_4.sum_of_intervals	kyu_5.extract_the_domain_name_from_url.extract_domain_name
module, 7	module, 32
kyu_4.sum_of_intervals.sum_of_intervals	kyu_5.extract_the_domain_name_from_url.test_domain_name
module, 7	module, 32
kyu_4.sum_of_intervals.test_sum_of_intervals	kyu_5.fibonacci_streaming
module, 7	module, 18
kyu_4.the_greatest_warrior	kyu_5.fibonacci_streaming.all_fibonacci_numbers
module, 15	module, 17
kyu_4.the_greatest_warrior.test_battle	kyu_5.fibonacci_streaming.test_all_fibonacci_numbers
module, 13	module, 17
kyu_4.the_greatest_warrior.test_warrior	kyu_5.find_the_safest_places_in_town
module, 14	module, 32
kyu_4.the_greatest_warrior.warrior	kyu_5.find_the_safest_places_in_town.advice
module, 14	module, 30
kyu_4.validate_sudoku_with_size	kyu_5.find_the_safest_places_in_town.print_agents
module, 11	module, 31
kyu_4.validate_sudoku_with_size.sudoku	kyu_5.find_the_safest_places_in_town.test_advice
module, 10	module, 31
kyu_4.validate_sudoku_with_size.test_sudoku	kyu_5.first_non_repeating_character

module, 27	module, 30
kyu_5.first_non_repeating_character.first_non_repeating_character.module, 27	kyu_5.sports_league_table_ranking.compute_ranks
kyu_5.first_non_repeating_character.test_first_non_repeating_character.module, 27	module, 27
kyu_5.flatten	kyu_5.string_incrementer
module, 27	module, 35
kyu_5.flatten.flatten	kyu_5.string_incrementer.string_incrementer
module, 26	module, 35
kyu_5.flatten.test_flatten	kyu_5.string_incrementer.test_increment_string
module, 26	module, 35
kyu_5.human_readable_time	kyu_5.sum_of_pairs
module, 20	module, 34
kyu_5.human_readable_time.make_readable	kyu_5.sum_of_pairs.sum_pairs
module, 20	module, 33
kyu_5.human_readable_time.test_make_readable	kyu_5.sum_of_pairs.test_sum_pairs
module, 20	module, 33
kyu_5.integers_recreation_one	kyu_5.the_hashtag_generator
module, 36	module, 33
kyu_5.integers_recreation_one.solution	kyu_5.the_hashtag_generator.hashtag_generator
module, 35	module, 32
kyu_5.integers_recreation_one.test_list_squared	kyu_5.the_hashtag_generator.test_generate_hashtag
module, 36	module, 33
kyu_5.master_your_primes_sieve_with_memory	kyu_5.tic_tac_toe_checker
module, 25	module, 35
kyu_5.master_your_primes_sieve_with_memory	kyu_5.tic_tac_toe_checker.checker
module, 25	module, 34
kyu_5.master_your_primes_sieve_with_memory	kyu_5.tic_tac_toe_checker.test_checker
module, 25	module, 34
kyu_5.moving_zeros_to_the_end	kyu_5.valid_parentheses
module, 22	module, 22
kyu_5.moving_zeros_to_the_end.move_zeros	kyu_5.valid_parentheses.test_valid_parentheses
module, 22	module, 21
kyu_5.moving_zeros_to_the_end.test_move_zeros	kyu_5.valid_parentheses.valid_parentheses
module, 22	module, 22
kyu_5.not_very_secure	kyu_5.where_my_anagrams_at
module, 19	module, 25
kyu_5.not_very_secure.alphanumeric	kyu_5.where_my_anagrams_at.anagrams
module, 18	module, 24
kyu_5.not_very_secure.test_alphanumeric	kyu_5.where_my_anagrams_at.test_anagrams
module, 19	module, 24
kyu_5.number_of_trailing_zeros_of_n	kyu_6
module, 26	module, 61
kyu_5.number_of_trailing_zeros_of_n.test_zeroes	kyu_6.a_rule_of_divisibility_by_13
module, 25	module, 52
kyu_5.number_of_trailing_zeros_of_n.zeros	kyu_6.a_rule_of_divisibility_by_13.test_thirt
module, 26	module, 52
kyu_5.simple_pig_latin	kyu_6.a_rule_of_divisibility_by_13.thirt
module, 20	module, 52
kyu_5.simple_pig_latin.pig_it	kyu_6.array_diff
module, 19	module, 61
kyu_5.simple_pig_latin.test_pig_it	kyu_6.array_diff.solution
module, 19	module, 61
kyu_5.sports_league_table_ranking	kyu_6.array_diff.test_array_diff

module, 61	module, 52
kyu_6.array_to_html_table	kyu_6 duplicate_encoder
module, 55	module, 44
kyu_6.array_to_html_table.to_table	kyu_6 duplicate_encoder.duplicate_encode
module, 55	module, 43
kyu_6.binary_to_text_ascii_conversion	kyu_6 duplicate_encoder.test_duplicate_encode
module, 47	module, 43
kyu_6.binary_to_text_ascii_conversion.binary_to_text_ascii_conversion	kyu_6.easy_diagonal
module, 46	module, 55
kyu_6.binary_to_text_ascii_conversion.test_binary_to_text_ascii_conversion	kyu_6.easy_diagonal.test_diagonal
module, 47	module, 54
kyu_6.casino_chips	kyu_6.encrypt_this
module, 47	module, 59
kyu_6.casino_chips.solve	kyu_6.encrypt_this.solution
module, 47	module, 59
kyu_6.casino_chips.test_solve	kyu_6.encrypt_this.test_encrypt_this
module, 47	module, 59
kyu_6.character_frequency	kyu_6.find_the_odd_int
module, 40	module, 37
kyu_6.character_frequency.character_frequency	kyu_6.find_the_odd_int.find_the_odd_int
module, 39	module, 37
kyu_6.character_frequency.test_character_frequency	kyu_6.find_the_odd_int.test_find_the_odd_int
module, 39	module, 37
kyu_6.color_choice	kyu_6.first_character_that_repeats
module, 53	module, 38
kyu_6.color_choice.checkchoose	kyu_6.first_character_that_repeats.first_character_that_repeats
module, 52	module, 37
kyu_6.color_choice.test_checkchoose	kyu_6.first_character_that_repeats.test_first_character_that_repeats
module, 53	module, 37
kyu_6.count_letters_in_string	kyu_6.format_string_of_names
module, 43	module, 60
kyu_6.count_letters_in_string.count_letters_in_string	kyu_6.format_string_of_names.solution
module, 42	module, 59
kyu_6.count_letters_in_string.test_count_letters_in_string	kyu_6.format_string_of_names.test_name_list
module, 42	module, 60
kyu_6.decipher_this	kyu_6.help_the_bookseller
module, 59	module, 49
kyu_6.decipher_this.solution	kyu_6.help_the_bookseller.stock_list
module, 58	module, 49
kyu_6.decipher_this.test_decipher_this	kyu_6.help_the_bookseller.test_stock_list
module, 58	module, 49
kyu_6.default_list	kyu_6.longest_repetition
module, 54	module, 39
kyu_6.default_list.default_list	kyu_6.longest_repetition.longest_repetition
module, 53	module, 38
kyu_6.default_list.test_default_list	kyu_6.longest_repetition.test_longest_repetition
module, 53	module, 38
kyu_6.disease_spread	kyu_6.multiples_of_3_or_5
module, 52	module, 46
kyu_6.disease_spread.epidemic	kyu_6.multiples_of_3_or_5.solution
module, 50	module, 45
kyu_6.disease_spread.epidemic_test_data	kyu_6.multiples_of_3_or_5.test_solution
module, 51	
kyu_6.disease_spread.test_epidemic	

module, 45	module, 60
kyu_6.number_zoo_patrol	kyu_6.string_subpattern_recognition_1
module, 56	module, 40
kyu_6.number_zoo_patrol.missing_number	kyu_6.string_subpattern_recognition_1.has_subpattern
module, 56	module, 40
kyu_6.number_zoo_patrol.test_find_missing_number	kyu_6.string_subpattern_recognition_1.test_has_subpattern
module, 56	module, 40
kyu_6.numericals_of_string	kyu_6.string_subpattern_recognition_2
module, 39	module, 41
kyu_6.numericals_of_string.numericals	kyu_6.string_subpattern_recognition_2.has_subpattern
module, 39	module, 40
kyu_6.numericals_of_string.test_numericals	kyu_6.string_subpattern_recognition_2.test_has_subpattern
module, 39	module, 41
kyu_6.permute_a_palindrome	kyu_6.string_subpattern_recognition_3
module, 42	module, 42
kyu_6.permute_a_palindrome.permute_a_palindrome	kyu_6.string_subpattern_recognition_3.has_subpattern
module, 42	module, 41
kyu_6.permute_a_palindrome.test_permute_a_palindrome	kyu_6.string_subpattern_recognition_3.test_has_subpattern
module, 42	module, 41
kyu_6.pokemon_damage_calculator	kyu_6.string_transformer
module, 49	module, 45
kyu_6.pokemon_damage_calculator.calculate_damage	kyu_6.string_transformer.string_transformer
module, 48	module, 44
kyu_6.pokemon_damage_calculator.test_calculate_damage	kyu_6.string_transformer.test_string_transformer
module, 48	module, 45
kyu_6.potion_class_101	kyu_6.sum_of_digits_digital_root
module, 50	module, 46
kyu_6.potion_class_101.potion	kyu_6.sum_of_digits_digital_root.digital_root
module, 50	module, 46
kyu_6.potion_class_101.test_potion	kyu_6.sum_of_digits_digital_root.test_digital_root
module, 50	module, 46
kyu_6.pyramid_array	kyu_6.unique_in_order
module, 38	module, 43
kyu_6.pyramid_array.pyramid_array	kyu_6.unique_in_order.test_unique_in_order
module, 38	module, 43
kyu_6.pyramid_array.test_pyramid_array	kyu_6.unique_in_order.unique_in_order
module, 38	module, 43
kyu_6.rotate_the_letters_of_each_element	kyu_6.vasya_clerk
module, 56	module, 44
kyu_6.rotate_the_letters_of_each_element.rotate_the_letters_of_each_element	kyu_6.vasya_clerk.test_tickets
module, 55	module, 44
kyu_6.rotate_the_letters_of_each_element.test_rotate_the_letters_of_each_element	kyu_6.vasya_clerk.test_tickets
module, 56	module, 44
kyu_6.row_of_the_odd_triangle	kyu_6.who_likes_it
module, 50	module, 58
kyu_6.row_of_the_odd_triangle.odd_row	kyu_6.who_likes_it.likes_function
module, 49	module, 57
kyu_6.row_of_the_odd_triangle.test_odd_row	kyu_6.who_likes_it.test_likes_function
module, 50	module, 58
kyu_6.sort_the_odd	kyu_6.your_order_please
module, 61	module, 57
kyu_6.sort_the_odd.solution	kyu_6.your_order_please.order
module, 60	module, 56
kyu_6.sort_the_odd.test_sort_array	kyu_6.your_order_please.test_order

module, 57	module, 76
kyu_7	kyu_7.formatting_decimal_places_1.two_decimal_places
module, 83	module, 76
kyu_7.always_perfect	kyu_7.fun_with_lists_length
module, 76	module, 71
kyu_7.always_perfect.check_root	kyu_7.fun_with_lists_length.length
module, 75	module, 70
kyu_7.always_perfect.test_check_root	kyu_7.fun_with_lists_length.node
module, 76	module, 71
kyu_7.basic_math_add_or_subtract	kyu_7.fun_with_lists_length.test_length
module, 81	module, 71
kyu_7.basic_math_add_or_subtract.calculator	kyu_7.growing_plant
module, 80	module, 80
kyu_7.basic_math_add_or_subtract.test_calculator	kyu_7.growing_plant.growing_plant
module, 81	module, 79
kyu_7.beginner_series_sum_of_numbers	kyu_7.growing_plant.test_growing_plant
module, 62	module, 80
kyu_7.beginner_series_sum_of_numbers.sum_of_numbers	kyu_7.help_bob_count_letters_and_digits
module, 61	module, 82
kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers	kyu_7.help_bob_count_letters_and_digits.count_letters
module, 62	module, 81
kyu_7.disemvowel_trolls	kyu_7.help_bob_count_letters_and_digits.test_count_letters
module, 63	module, 82
kyu_7.disemvowel_trolls.disemvowel_trolls	kyu_7.isograms
module, 62	module, 83
kyu_7.disemvowel_trolls.test_disemvowel_trolls	kyu_7.isograms.is_isogram
module, 62	module, 83
kyu_7.easy_line	kyu_7.isograms.test_is_isogram
module, 83	module, 83
kyu_7.easy_line.easylines	kyu_7.jaden_casing_strings
module, 82	module, 63
kyu_7.easy_line.test_easylines	kyu_7.jaden_casing_strings.jaden_casing_strings
module, 82	module, 63
kyu_7.factorial	kyu_7.jaden_casing_strings.test_jaden_casing_strings
module, 79	module, 63
kyu_7.factorial.factorial	kyu_7.make_class
module, 78	module, 74
kyu_7.factorial.test_factorial	kyu_7.make_class.animal
module, 78	module, 73
kyu_7.fill_the_hard_disk_drive	kyu_7.make_class.make_class
module, 72	module, 74
kyu_7.fill_the_hard_disk_drive.save	kyu_7.make_class.test_make_class
module, 71	module, 74
kyu_7.fill_the_hard_disk_drive.test_save	kyu_7.maximum_multiple
module, 72	module, 73
kyu_7.find_the_longest_gap	kyu_7.maximum_multiple.maximum_multiple
module, 79	module, 73
kyu_7.find_the_longest_gap.gap	kyu_7.password_validator
module, 79	module, 75
kyu_7.find_the_longest_gap.test_gap	kyu_7.password_validator.password
module, 79	module, 74
kyu_7.formatting_decimal_places_1	kyu_7.password_validator.test_password
module, 77	module, 74
kyu_7.formatting_decimal_places_1.test_two_decimal_places	kyu_7.power_of_three

module, 67	module, 67
kyu_7.powers_of_3.largest_power	kyu_7.sum_of_powers_of_2.sum_of_powers_of_2
module, 67	module, 66
kyu_7.powers_of_3.test_largest_power	kyu_7.sum_of_powers_of_2.test_sum_of_powers_of_2
module, 67	module, 67
kyu_7.pull_your_words_together_man	kyu_7.sum_of_triangular_numbers
module, 78	module, 68
kyu_7.pull_your_words_together_man.sentence	kyu_7.sum_of_triangular_numbers.sum_triangular_numbers
module, 77	module, 67
kyu_7.pull_your_words_together_man.test_sentence	kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers
module, 78	module, 68
kyu_7.remove_the_minimum	kyu_7.sum_of_two_lowest_int
module, 65	module, 66
kyu_7.remove_the_minimum.remove_the_minimum	kyu_7.sum_of_two_lowest_int.sum_two_smallest_int
module, 63	module, 65
kyu_7.remove_the_minimum.test_remove_the_minimum	kyu_7.sum_of_two_lowest_int.test_sum_two_smallest_int
module, 64	module, 65
kyu_7.share_prices	kyu_7.the_first_non_repeated_character_in_string
module, 75	module, 73
kyu_7.share_prices.share_price	kyu_7.the_first_non_repeated_character_in_string.first
module, 75	module, 72
kyu_7.share_prices.test_share_price	kyu_7.the_first_non_repeated_character_in_string.test
module, 75	module, 72
kyu_7.significant_figures	kyu_7.vaporcode
module, 70	module, 68
kyu_7.significant_figures.number_of_significant_figures	kyu_7.vaporcode.test_vaporcode
module, 69	module, 68
kyu_7.significant_figures.test_number_of_significant_figures	kyu_7.vaporcode.vaporcode
module, 69	module, 68
kyu_7.simple_fun_152	kyu_7.you_are_square
module, 69	module, 66
kyu_7.simple_fun_152.invite_more_women	kyu_7.you_are_square.test_you_are_square
module, 68	module, 66
kyu_7.simple_fun_152.test_invite_more_women	kyu_7.you_are_square.you_are_square
module, 69	module, 66
kyu_7.sort_out_the_men_from_boys	kyu_8
module, 70	module, 105
kyu_7.sort_out_the_men_from_boys.men_from_boys	kyu_8.alternating_case
module, 70	module, 93
kyu_7.sort_out_the_men_from_boys.test_men_from_boys	kyu_8.alternating_case.alternating_case
module, 70	module, 92
kyu_7.substituting_variables_into_strings	kyu_8.alternating_case.test_alternating_case
module, 77	module, 93
kyu_7.substituting_variables_into_strings.spaced_numbers	kyu_8.century_from_year
module, 77	module, 104
kyu_7.substituting_variables_into_strings.spaced_numbers	kyu_8.century_from_year.test_century
module, 77	module, 103
kyu_7.sum_of_odd_numbers	kyu_8.check_the_exam
module, 81	module, 98
kyu_7.sum_of_odd_numbers.row_sum_odd_numbers	kyu_8.check_the_exam.check_exam
module, 81	module, 98
kyu_7.sum_of_odd_numbers.test_row_sum_odd_numbers	kyu_8.check_the_exam.test_check_exam
module, 81	
kyu_7.sum_of_powers_of_2	

module, 98	module, 86
kyu_8.convert_string_to_an_array	kyu_8.grasshopper_summation
module, 100	module, 93
kyu_8.convert_string_to_an_array.string_to_array	kyu_8.grasshopper_summation.summation
module, 99	module, 93
kyu_8.convert_string_to_an_array.test_string_to_array	kyu_8.grasshopper_summation.test_summation
module, 100	module, 93
kyu_8.count_the_monkeys	kyu_8.greek_sort
module, 101	module, 105
kyu_8.count_the_monkeys.monkey_count	kyu_8.greek_sort.greek_comparator
module, 101	module, 105
kyu_8.count_the_monkeys.test_monkey_count	kyu_8.greek_sort.test_greek_comparator
module, 101	module, 105
kyu_8.counting_sheep	kyu_8.holiday_vi_shark_pontoon
module, 97	module, 105
kyu_8.counting_sheep.counting_sheep	kyu_8.holiday_vi_shark_pontoon.shark
module, 96	module, 104
kyu_8.counting_sheep.test_counting_sheep	kyu_8.holiday_vi_shark_pontoon.test_shark
module, 97	module, 104
kyu_8.enumerable_magic_25	kyu_8.is_it_a_palindrome
module, 103	module, 99
kyu_8.enumerable_magic_25.take	kyu_8.is_it_a_palindrome.is_palindrome
module, 102	module, 98
kyu_8.enumerable_magic_25.test_take	kyu_8.is_it_a_palindrome.test_is_palindrome
module, 102	module, 99
kyu_8.find_the_first_non_consecutive_number	kyu_8.is_your_period_late
module, 90	module, 84
kyu_8.find_the_first_non_consecutive_number.first_non_consecutive	kyu_8.is_your_period_late.is_your_period_late
module, 90	module, 83
kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive	kyu_8.is_your_period_late.test_is_your_period_late
module, 90	module, 84
kyu_8.formatting_decimal_places_0	kyu_8.keep_hydrated
module, 99	module, 95
kyu_8.formatting_decimal_places_0.test_two_decimal_places	kyu_8.keep_hydrated.keep_hydrated
module, 99	module, 94
kyu_8.formatting_decimal_places_0.two_decimal_places	kyu_8.keep_hydrated.test_keep_hydrated
module, 99	module, 95
kyu_8.grasshopper_check_for_factor	kyu_8.keep_up_the_hoop
module, 98	module, 102
kyu_8.grasshopper_check_for_factor.check_for_factor	kyu_8.keep_up_the_hoop.hoop_count
module, 97	module, 101
kyu_8.grasshopper_check_for_factor.test_check_for_factor	kyu_8.keep_up_the_hoop.test_hoop_count
module, 97	module, 102
kyu_8.grasshopper_messi_goals_function	kyu_8.logical_calculator
module, 87	module, 85
kyu_8.grasshopper_messi_goals_function.messi_goals_function	kyu_8.logical_calculator.logical_calculator
module, 86	module, 84
kyu_8.grasshopper_messi_goals_function.test_messi_goals_function	kyu_8.logical_calculator.test_logical_calculator
module, 87	module, 84
kyu_8.grasshopper_personalized_message	kyu_8.make_upper_case
module, 86	module, 88
kyu_8.grasshopper_personalized_message.grasshopper_personalize_message	kyu_8.make_upper_case.make_upper_case
module, 86	module, 88
kyu_8.grasshopper_personalized_message.test_grasshopper_personalize_message	kyu_8.make_upper_case.test_make_upper_case

module, 88
 kyu_8.multiply
 module, 86
 kyu_8.multiply.multiply
 module, 85
 kyu_8.multiply.test_multiply
 module, 85
 kyu_8.my_head_is_at_the_wrong_end
 module, 94
 kyu_8.my_head_is_at_the_wrong_end.fix_the_my_head_is_at_the_wrong_end
 module, 93
 kyu_8.my_head_is_at_the_wrong_end.test_fix_the_my_head_is_at_the_wrong_end
 module, 94
 kyu_8.remove_first_and_last_character
 module, 91
 kyu_8.remove_first_and_last_character.remove_character
 module, 91
 kyu_8.remove_first_and_last_character.test_remove_character
 module, 91
 kyu_8.remove_string_spaces
 module, 87
 kyu_8.remove_string_spaces.remove_string_spaces
 module, 87
 kyu_8.remove_string_spaces.test_remove_string_spaces
 module, 87
 kyu_8.reversed_strings
 module, 92
 kyu_8.reversed_strings.reversed_strings
 module, 91
 kyu_8.reversed_strings.test_reversed_strings
 module, 92
 kyu_8.set_alarm
 module, 96
 kyu_8.set_alarm.set_alarm
 module, 95
 kyu_8.set_alarm.test_set_alarm
 module, 95
 kyu_8.surface_area_and_volume_of_box
 module, 92
 kyu_8.surface_area_and_volume_of_box.get_surface_area_and_volume_of_box
 module, 92
 kyu_8.surface_area_and_volume_of_box.test_get_surface_area_and_volume_of_box
 module, 92
 kyu_8.swap_values
 module, 94
 kyu_8.swap_values.swap_values
 module, 94
 kyu_8.swap_values.test_swap_values
 module, 94
 kyu_8.terminal_game_move_function
 module, 89
 kyu_8.terminal_game_move_function.terminal_game_move_function
 module, 88
 kyu_8.terminal_game_move_function.test_terminal_game_move_function
 module, 88
 module, 89
 kyu_8.the_feast_of_many_beasts
 module, 101
 kyu_8.the_feast_of_many_beasts.feast
 module, 100
 kyu_8.the_feast_of_many_beasts.test_feast
 module, 100
 kyu_8.third_angle_of_triangle
 module, 91
 kyu_8.third_angle_of_triangle.test_third_angle_of_triangle
 module, 90
 kyu_8.third_angle_of_triangle.third_angle_of_triangle
 module, 91
 kyu_8.well_of_ideas_easy_version
 module, 88
 kyu_8.well_of_ideas_easy_version.test_well_of_ideas_easy_version
 module, 87
 kyu_8.will_there_be_enough_space
 module, 88
 kyu_8.will_there_be_enough_space.enough
 module, 96
 kyu_8.will_there_be_enough_space.test_enough
 module, 96
 kyu_8.will_you_make_it
 module, 103
 kyu_8.will_you_make_it.test_zero_fuel
 module, 103
 kyu_8.will_you_make_it.zero_fuel
 module, 103
 kyu_8.wolf_in_sheep_clothing
 module, 90
 kyu_8.wolf_in_sheep_clothing.test_wolf_in_sheep_clothing
 module, 89
 kyu_8.wolf_in_sheep_clothing.wolf_in_sheep_clothing
 module, 89

L

largestPower() (in module
 kyu_7.powers_of_3.largest_power), 67
 LargestPowerTestCase (class in
 kyu_7.powers_of_3.test_largest_power),
 67
 last_digit_index() (in module
 kyu_6.decipher_this.solution), 58
 left() (in module kyu_3.make_spiral.solution), 5
 length() (in module
 kyu_7.fun_with_lists_length.length), 70
 LengthTestCase (class in
 kyu_7.fun_with_lists_length.test_length),
 71
 letter_count() (in module
 kyu_6.count_letters_in_string.count_letters_in_string),
 60

letter_frequency()	(in module kyu_6.character_frequency.character_frequency),	42	men_from_boys()	(in module kyu_7.sort_out_the_men_from_boys.men_from_boys),	70
LetterFrequencyTestCase	(class in kyu_6.character_frequency.test_character_frequency),	39	MenFromBoysTestCase	(class in kyu_7.sort_out_the_men_from_boys.test_men_from_boys),	70
level()	(kyu_4.the_greatest_warrior.warrior.Warrior property),	14	mix()	(in module kyu_4.strings_mix.solution),	15
likes()	(in module kyu_6.who_likes_it.likes_function),	57	mix()	(kyu_6.potion_class_101.potion.Potion method),	50
LikesTestCase	(class in kyu_6.who_likes_it.test_likes_function),	58	MixTestCase	(class in kyu_4.strings_mix.test_mix),	15
list_squared()	(in module kyu_5.integers_recreation_one.solution),	36	module		
ListSquaredTestCase	(class in kyu_5.integers_recreation_one.test_list_squared),	36	img,	1	
litres()	(in module kyu_8.keep_hydrated.keep_hydrated),	94	kyu_2,	2	
logical_calc()	(in module kyu_8.logical_calculator.logical_calculator),	84	kyu_2.evaluate_mathematical_expression,	2	
LogicalCalculatorTestCase	(class in kyu_8.logical_calculator.test_logical_calculator),	84	kyu_2.evaluate_mathematical_expression.evaluate	1	
longest_repetition()	(in module kyu_6.longest_repetition.longest_repetition),	38	kyu_2.evaluate_mathematical_expression.test_eva	2	
LongestRepetitionTestCase	(class in kyu_6.longest_repetition.test_longest_repetition),	38	kyu_3,	7	
M			kyu_3.battleship_field_validator,	7	
make_class()	(in module kyu_7.make_class.make_class),	74	kyu_3.battleship_field_validator.test_battleshi	6	
make_readable()	(in module kyu_5.human_readable_time.make_readable),	20	kyu_3.battleship_field_validator.validator,	6	
make_upper_case()	(in module kyu_8.make_upper_case.make_upper_case),	88	kyu_3.calculator,	3	
MakeClassTestCase	(class in kyu_7.make_class.test_make_class),	74	kyu_3.calculator.calculator,	2	
MakeReadableTestCase	(class in kyu_5.human_readable_time.test_make_readable),	20	kyu_3.calculator.test_calculator,	3	
MakeUpperCaseTestCase	(class in kyu_8.make_upper_case.test_make_upper_case),	88	kyu_3.make_spiral,	6	
max_multiple()	(in module kyu_7.maximum_multiple.maximum_multiple),	73	kyu_3.make_spiral.solution,	4	
			kyu_3.make_spiral.test_spiralize,	5	
			kyu_3.rail_fence_cipher_encoding_and_decoding,	4	
			kyu_3.rail_fence_cipher_encoding_and_decoding.e	3	
			kyu_3.rail_fence_cipher_encoding_and_decoding.t	4	
			kyu_3.rail_fence_cipher_encoding_and_decoding.t	4	
			kyu_4,	17	
			kyu_4.human_readable_duration_format,	9	
			kyu_4.human_readable_duration_format.format_dur	7	
			kyu_4.human_readable_duration_format.test_forma	9	
			kyu_4.most_frequently_used_words,	13	
			kyu_4.most_frequently_used_words.solution,	13	
			kyu_4.most_frequently_used_words.test_top_3_wor	13	
			kyu_4.next_bigger_number_with_the_same_digits,	17	
			kyu_4.next_bigger_number_with_the_same_digits.n		

16	20
kyu_4.next_bigger_number_with_the_same_digits, 16	kyu_5.alphabet_wars_nuclear_strike.test_alphab
17	21
kyu_4.next_smaller_number_with_the_same_digits, 17	kyu_5.count_ip_addresses, 18
16	kyu_5.count_ip_addresses.ips_between,
kyu_4.next_smaller_number_with_the_same_digits, 16	18s.next_smaller,
15	kyu_5.count_ip_addresses.test_ips_between,
kyu_4.next_smaller_number_with_the_same_digits, 15	18s.test_next_smaller,
16	kyu_5.did_i_finish_my_sudoku, 24
kyu_4.range_extraction, 10	kyu_5.did_i_finish_my_sudoku.is_sudoku_done,
kyu_4.range_extraction.solution, 10	23
kyu_4.range_extraction.test_solution,	kyu_5.did_i_finish_my_sudoku.sudoku_by_column,
10	23
kyu_4.snail, 12	kyu_5.did_i_finish_my_sudoku.sudoku_by_regions,
kyu_4.snail.snail_sort, 11	23
kyu_4.snail.test_snail, 12	kyu_5.did_i_finish_my_sudoku.sudoku_by_row,
kyu_4.strings_mix, 15	24
kyu_4.strings_mix.solution, 15	kyu_5.did_i_finish_my_sudoku.test_did_i_finish_
kyu_4.strings_mix.test_mix, 15	24
kyu_4.strip_comments, 11	kyu_5.directions_reduction, 23
kyu_4.strip_comments.solution, 11	kyu_5.directions_reduction.directions_reduction
kyu_4.strip_comments.test_solution,	22
11	kyu_5.directions_reduction.test_directions_redu
kyu_4.sudoku_solution_validator, 10	23
kyu_4.sudoku_solution_validator.test_vali	kyu_5.extract_the_domain_name_from_url,
9	32
kyu_4.sudoku_solution_validator.valid_soku	kyu_5.extract_the_domain_name_from_url.extract_
10	32
kyu_4.sum_by_factors, 13	kyu_5.extract_the_domain_name_from_url.test_dom
kyu_4.sum_by_factors.sum_for_list,	32
12	kyu_5.fibonacci_streaming, 18
kyu_4.sum_by_factors.test_sum_for_list,	kyu_5.fibonacci_streaming.all_fibonacci_numbers
12	17
kyu_4.sum_of_intervals, 7	kyu_5.fibonacci_streaming.test_all_fibonacci_nu
kyu_4.sum_of_intervals.sum_of_intervals,	17
7	kyu_5.find_the_safest_places_in_town,
kyu_4.sum_of_intervals.test_sum_of_interval	32
7	kyu_5.find_the_safest_places_in_town.advice,
kyu_4.the_greatest_warrior, 15	30
kyu_4.the_greatest_warrior.test_battle,	kyu_5.find_the_safest_places_in_town.printAgen
13	31
kyu_4.the_greatest_warrior.test_warrior,	kyu_5.find_the_safest_places_in_town.test_advic
14	31
kyu_4.the_greatest_warrior.warrior,	kyu_5.first_non_repeating_character,
14	27
kyu_4.validate_sudoku_with_size, 11	kyu_5.first_non_repeating_character.first_non_r
kyu_4.validate_sudoku_with_size.sudoku,	27
10	kyu_5.first_non_repeating_character.test_first_
kyu_4.validate_sudoku_with_size.test_sudoku,	27
11	kyu_5.flatten, 27
kyu_5, 36	kyu_5.flatten.flatten, 26
kyu_5.alphabet_wars_nuclear_strike,	kyu_5.flatten.test_flatten, 26
21	kyu_5.human_readable_time, 20
kyu_5.alphabet_wars_nuclear_strike.alphab	kyu_5.human_readable_time.make_readable,

20 kyu_5.tic_tac_toe_checker, 35
 kyu_5.human_readable_time.test_make_readable, 35
 20 kyu_5.tic_tac_toe_checker.checker, 34
 kyu_5.integers_recreation_one, 36 kyu_5.tic_tac_toe_checker.test_checker, 34
 kyu_5.integers_recreation_one.solution, 34
 35 kyu_5.valid_parentheses, 22
 kyu_5.integers_recreation_one.test_list_square, 22
 36 kyu_5.valid_parentheses.test_valid_parentheses, 21
 kyu_5.master_your_primes_sieve_with_memory, 22
 25 kyu_5.valid_parentheses.valid_parentheses, 22
 kyu_5.master_your_primes_sieve_with_memory, 25
 25 kyu_5.where_my_anagrams_at, 25
 kyu_5.master_your_primes_sieve_with_memory, 21
 25 kyu_5.where_my_anagrams_at.test_primes, 21
 kyu_5.moving_zeros_to_the_end, 22 kyu_5.where_my_anagrams_at.test_anagrams, 24
 kyu_5.moving_zeros_to_the_end.move_zeros, 22
 22 kyu_6, 61
 kyu_5.moving_zeros_to_the_end.test_move_zeros, 22
 22 kyu_6.a_rule_of_divisibility_by_13, 53
 kyu_5.not_very_secure, 19 kyu_6.a_rule_of_divisibility_by_13.test_thirt, 52
 kyu_5.not_very_secure.alphanumeric, 18 kyu_6.a_rule_of_divisibility_by_13.thirt, 52
 kyu_5.not_very_secure.test_alphanumeric, 19 kyu_6.array_diff, 61
 kyu_5.number_of_trailing_zeros_of_n, 26 kyu_6.array_diff.solution, 61
 kyu_5.number_of_trailing_zeros_of_n.test_kyu_6, 25
 25 kyu_6.array_diff.test_array_diff, 61
 kyu_5.number_of_trailing_zeros_of_n.zeros, 26 kyu_6.array_to_html_table, 55
 26 kyu_6.array_to_html_table.to_table, 55
 kyu_5.simple_pig_latin, 20 kyu_6.binary_to_text_ascii_conversion, 47
 kyu_5.simple_pig_latin.pig_it, 19 kyu_6.binary_to_text_ascii_conversion.binary_to_text_ascii_conversion, 46
 kyu_5.simple_pig_latin.test_pig_it, 19 kyu_6.binary_to_text_ascii_conversion.test_binary_to_text_ascii_conversion, 47
 kyu_5.sports_league_table_ranking, 30 kyu_6.casino_chips, 47
 kyu_5.sports_league_table_ranking.compute_rank, 27 kyu_6.casino_chips.solve, 47
 kyu_5.sports_league_table_ranking.test_compute_rank, 29 kyu_6.casino_chips.test_solve, 47
 29 kyu_6.character_frequency, 40
 kyu_5.string_incrementer, 35 kyu_6.character_frequency.character_frequency, 39
 kyu_5.string_incrementer.string_incrementer, 35 kyu_6.character_frequency.test_character_frequency, 39
 35 kyu_6.color_choice, 53
 kyu_5.string_incrementer.test_incrementer, 35 kyu_6.color_choice.checkchoose, 52
 35 kyu_6.color_choice.test_checkchoose, 53
 kyu_5.sum_of_pairs, 34 kyu_6.count_letters_in_string, 43
 kyu_5.sum_of_pairs.sum_pairs, 33 kyu_6.count_letters_in_string.count_letters_in_string, 42
 kyu_5.sum_of_pairs.test_sum_pairs, 33 kyu_6.count_letters_in_string.test_count_letters_in_string, 42
 kyu_5.the_hashtag_generator, 33 kyu_6.decipher_this, 59
 kyu_5.the_hashtag_generator.hashtag_generator, 32 kyu_6.decipher_this.solution, 58
 kyu_5.the_hashtag_generator.test_hashtag_generator, 33 kyu_6.decipher_this.test_decipher_this, 58

kyu_6.default_list, 54	kyu_6.number_zoo_patrol, 56
kyu_6.default_list.default_list, 53	kyu_6.number_zoo_patrol.missing_number,
kyu_6.default_list.test_default_list,	56
53	kyu_6.number_zoo_patrol.test_find_missing_numbe
kyu_6.disease_spread, 52	56
kyu_6.disease_spread.epidemic, 50	kyu_6.numericals_of_string, 39
kyu_6.disease_spread.epidemic_test_data,	kyu_6.numericals_of_string.numericals,
51	39
kyu_6.disease_spread.test_epidemic,	kyu_6.numericals_of_string.test_numericals,
52	39
kyu_6 duplicate_encoder, 44	kyu_6.permute_a_palindrome, 42
kyu_6 duplicate_encoder duplicate_encode	kyu_6.permute_a_palindrome.permute_a_palindrome
43	42
kyu_6 duplicate_encoder.test_duplicate_encoder,	kyu_6.permute_a_palindrome.test_permute_a_palin
43	42
kyu_6.easy_diagonal, 55	kyu_6.pokemon_damage_calculator, 49
kyu_6.easy_diagonal.diagonal, 54	kyu_6.pokemon_damage_calculator.calculate_damag
kyu_6.easy_diagonal.test_diagonal,	48
54	kyu_6.pokemon_damage_calculator.test_calculate_
kyu_6.encrypt_this, 59	48
kyu_6.encrypt_this.solution, 59	kyu_6.potion_class_101, 50
kyu_6.encrypt_this.test_encrypt_this,	kyu_6.potion_class_101.potion, 50
59	kyu_6.potion_class_101.test_potion,
kyu_6.find_the_odd_int, 37	50
kyu_6.find_the_odd_int.find_the_odd_int,	kyu_6.pyramid_array, 38
37	kyu_6.pyramid_array.pyramid_array,
kyu_6.find_the_odd_int.test_find_the_odd_int,	38
37	kyu_6.pyramid_array.test_pyramid_array,
kyu_6.first_character_that_repeats,	38
38	kyu_6.rotate_the_letters_of_each_element,
kyu_6.first_character_that_repeats.first_cha	kyu_6.rotate_the_letters_of_each_element.group
37	56
kyu_6.first_character_that_repeats.test_fir	kyu_6.rotate_the_letters_of_each_element.test_g
37	56
kyu_6.format_string_of_names, 60	kyu_6.row_of_the_odd_triangle, 50
kyu_6.format_string_of_names.solution,	kyu_6.row_of_the_odd_triangle.odd_row,
59	kyu_6.row_of_the_odd_triangle.test_odd_row,
kyu_6.format_string_of_names.test_namelist,	50
60	kyu_6.sort_the_odd, 61
kyu_6.help_the_bookseller, 49	kyu_6.sort_the_odd.solution, 60
kyu_6.help_the_bookseller.stock_list,	kyu_6.sort_the_odd.test_sort_array,
49	60
kyu_6.help_the_bookseller.test_stock_list,	kyu_6.string_subpattern_recognition_1,
49	kyu_6.string_subpattern_recognition_1.has_subpa
kyu_6.longest_repetition, 39	kyu_6.string_subpattern_recognition_1.test_has_
kyu_6.longest_repetition.longest_repetition,	40
38	kyu_6.string_subpattern_recognition_2,
kyu_6.longest_repetition.test_longest_repeti	41
38	
kyu_6.multiples_of_3_or_5, 46	
kyu_6.multiples_of_3_or_5.solution,	
45	
kyu_6.multiples_of_3_or_5.test_solution,	

40
 kyu_6.string_subpattern_recognition_2.test_string_subpattern_recognition_2, 62
 41
 62
 kyu_6.string_subpattern_recognition_3, kyu_7.easy_line, 83
 42
 kyu_7.easy_line.easyline, 82
 kyu_6.string_subpattern_recognition_3.has_subpattern, kyu_7.easylines.test_easylines, 82
 41
 kyu_7.factorial, 79
 kyu_6.string_subpattern_recognition_3.test_string_subpattern_recognition_3, kyu_7.factorial.test_factorial, 78
 41
 kyu_7.factorial.test_factorial, 78
 kyu_6.string_transformer, 45
 kyu_7.fill_the_hard_disk_drive, 72
 kyu_6.string_transformer.string_transformer, kyu_7.fill_the_hard_disk_drive.save, 71
 44
 kyu_7.fill_the_hard_disk_drive.test_save, 72
 45
 kyu_6.sum_of_digits_digital_root, 46
 kyu_7.find_the_longest_gap, 79
 kyu_6.sum_of_digits_digital_root.digitalkyu_7.find_the_longest_gap.gap, 79
 46
 kyu_7.find_the_longest_gap.test_gap, 79
 kyu_6.sum_of_digits_digital_root.test_digital_root, 79
 46
 kyu_7.formatting_decimal_places_1, 77
 kyu_6.unique_in_order, 43
 kyu_7.formatting_decimal_places_1.test_two_decimal_places_1, 76
 43
 kyu_6.unique_in_order.test_unique_in_order, kyu_7.formatting_decimal_places_1.two_decimal_places_1, 76
 43
 kyu_6.vasya_clerk, 44
 kyu_7.fun_with_lists_length, 71
 kyu_6.vasya_clerk.test_tickets, 44
 kyu_7.fun_with_lists_length.length, 70
 kyu_6.vasya_clerk.tickets, 44
 kyu_7.fun_with_lists_length.node, 71
 kyu_6.who_likes_it, 58
 kyu_7.fun_with_lists_length.test_length, 71
 57
 kyu_6.who_likes_it.likes_function, kyu_7.growing_plant, 80
 58
 kyu_6.who_likes_it.test_likes_function, kyu_7.growing_plant.growing_plant, 79
 kyu_6.your_order_please, 57
 kyu_7.growing_plant.test_growing_plant, 80
 kyu_6.your_order_please.order, 56
 kyu_6.your_order_please.test_order, 57
 kyu_7.help_bob_count_letters_and_digits, 82
 kyu_7, 83
 kyu_7.always_perfect, 76
 kyu_7.help_bob_count_letters_and_digits.count_letters_and_digits, 81
 kyu_7.always_perfect.check_root, 75
 kyu_7.always_perfect.test_check_root, kyu_7.help_bob_count_letters_and_digits.test_count_letters_and_digits, 82
 76
 kyu_7.basic_math_add_or_subtract, 81
 kyu_7.isograms, 83
 kyu_7.basic_math_add_or_subtract.calculate, kyu_7.isograms.is_isogram, 83
 80
 kyu_7.isograms.test_is_isogram, 83
 kyu_7.basic_math_add_or_subtract.test_calculate, kyu_7.jaden_casing_strings, 63
 81
 kyu_7.jaden_casing_strings.jaden_casing_strings, 63
 kyu_7.beginner_series_sum_of_numbers, 62
 kyu_7.jaden_casing_strings.test_jaden_casing_strings, 61
 kyu_7.beginner_series_sum_of_numbers.sum_of_numbers, 61
 kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers, kyu_7.make_class, 74
 62
 kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers, kyu_7.make_class.make_class, 74
 kyu_7.make_class, 74
 kyu_7.make_class.test_make_class, 74
 kyu_7.disemvowel_trolls, 63
 kyu_7.make_class.test_make_class, 74
 kyu_7.disemvowel_trolls.disemvowel_trolls, kyu_7.maximum_multiple, 73

kyu_7.maximum_multiple.maximum_multiple,	66	
73		kyu_7.sum_of_powers_of_2.test_sum_of_powers_of_
kyu_7.password_validator,75	67	
kyu_7.password_validator.password,		kyu_7.sum_of_triangular_numbers,68
74		kyu_7.sum_of_triangular_numbers.sum_triangular_
kyu_7.password_validator.test_password,	67	
74		kyu_7.sum_of_triangular_numbers.test_sum_triang
kyu_7.powers_of_3,67	68	
kyu_7.powers_of_3.largest_power,67		kyu_7.sum_of_two_lowest_int,66
kyu_7.powers_of_3.test_largest_power,		kyu_7.sum_of_two_lowest_int.sum_two_smallest_in
67	65	
kyu_7.pull_your_words_together_man,		kyu_7.sum_of_two_lowest_int.test_sum_two_smalle
78	65	
kyu_7.pull_your_words_together_man.sentence,	73	kyu_7.the_first_non_repeated_character_in_strin
77	73	
kyu_7.pull_your_words_together_man.test_sentence,	72	kyu_7.the_first_non_repeated_character_in_strin
78	72	
kyu_7.remove_the_minimum,65		kyu_7.the_first_non_repeated_character_in_strin
kyu_7.remove_the_minimum.remove_the_minimum,	72	
63		kyu_7.vaporcode,68
kyu_7.remove_the_minimum.test_remove_the_minim		kyu_7.vaporcode.test_vaporcode,68
64		kyu_7.vaporcode.vaporcode,68
kyu_7.share_prices,75		kyu_7.you_are_square,66
kyu_7.share_prices.share_price,75		kyu_7.you_are_square.test_you_are_square,
kyu_7.share_prices.test_share_price,	66	
75		kyu_7.you_are_square.you_are_square,
kyu_7.significant_figures,70	66	
kyu_7.significant_figures.number_of_signifi	8, 105	
69		kyu_8.alternating_case,93
kyu_7.significant_figures.test_number_of_s		kyu_8.alternating_case.alternating_case,
69	92	
kyu_7.simple_fun_152,69		kyu_8.alternating_case.test_alternating_case,
kyu_7.simple_fun_152.invite_more_women,	93	
68		kyu_8.century_from_year,104
kyu_7.simple_fun_152.test_invite_more_women,		kyu_8.century_from_year.century,103
69		kyu_8.century_from_year.test_century,
kyu_7.sort_out_the_men_from_boys,70	104	
kyu_7.sort_out_the_men_from_boys.men_from_boys,		kyu_8.check_the_exam,98
70		kyu_8.check_the_exam.check_exam,98
kyu_7.sort_out_the_men_from_boys.test_men_from_boys,		kyu_8.check_the_exam.test_check_exam,
70	98	
kyu_7.substituting_variables_into_strings,		kyu_8.padded_numbers.string_to_an_array,
77	100	
kyu_7.substituting_variables_into_strings.padded_numbers,		kyu_8.padded_numbers.string_to_an_array.string_to_arra
77	99	
kyu_7.substituting_variables_into_strings.padded_numbers,		kyu_8.padded_numbers.string_to_an_array.test_string_to
77	100	
kyu_7.sum_of_odd_numbers,81		kyu_8.count_the_monkeys,101
kyu_7.sum_of_odd_numbers.row_sum_odd_num		kyu_8.count_the_monkeys.monkey_count,
81	101	
kyu_7.sum_of_odd_numbers.test_row_sum_odd_nu		kyu_8.count_the_monkeys.test_monkey_count,
81	101	
kyu_7.sum_of_powers_of_2,67		kyu_8.counting_sheep,97
kyu_7.sum_of_powers_of_2.sum_of_powers_of_2,		kyu_8.counting_sheep.counting_sheep,

96	98
kyu_8.counting_sheep.test_counting_sheep	kyu_8.is_it_a_palindrome.test_is_palindrome,
97	99
kyu_8.enumerable_magic_25, 103	kyu_8.is_your_period_late, 84
kyu_8.enumerable_magic_25.take, 102	kyu_8.is_your_period_late.is_your_period_late,
kyu_8.enumerable_magic_25.test_take,	83
102	kyu_8.is_your_period_late.test_is_your_period_late,
kyu_8.find_the_first_non_consecutive_number, 84	
90	kyu_8.keep_hydrated, 95
kyu_8.find_the_first_non_consecutive_number,	kyu_8.keep_hydrated.test_keep_hydrated,
90	94
kyu_8.find_the_first_non_consecutive_number,	kyu_8.keep_hydrated.test_keep_hydrated,
90	95
kyu_8.formatting_decimal_places_0,	kyu_8.keep_up_the_hoop, 102
99	kyu_8.keep_up_the_hoop.hoop_count,
kyu_8.formatting_decimal_places_0.test_two_decimal_places,	
99	kyu_8.keep_up_the_hoop.test_hoop_count,
kyu_8.formatting_decimal_places_0.two_decimal_places,	102
99	kyu_8.logical_calculator, 85
kyu_8.grasshopper_check_for_factor,	kyu_8.logical_calculator.logical_calculator,
98	84
kyu_8.grasshopper_check_for_factor.check_for_factor,	kyu_8.logical_calculator.test_logical_calculator,
97	84
kyu_8.grasshopper_check_for_factor.test_kyu_8_fake_factor,	88
97	kyu_8.make_upper_case.make_upper_case,
kyu_8.grasshopper_messi_goals_function,	88
87	kyu_8.make_upper_case.test_make_upper_case,
kyu_8.grasshopper_messi_goals_function.messi_goals_function,	88
86	kyu_8.multiply, 86
kyu_8.grasshopper_messi_goals_function.test_result_is_in_the_right_order,	85
87	kyu_8.multiply.test_multiply, 85
kyu_8.grasshopper_personalized_message,	kyu_8.my_head_is_at_the_wrong_end,
86	94
kyu_8.grasshopper_personalized_message.grasshopper_personalized_message,	kyu_8.my_head_is_at_the_wrong_end.test_fix_the_meerkat,
86	93
kyu_8.grasshopper_personalized_message.test_grasshopper_is_at_the_wrong_end,	94
86	
kyu_8.grasshopper_summation, 93	kyu_8.remove_first_and_last_character,
kyu_8.grasshopper_summation.summation,	91
93	kyu_8.remove_first_and_last_character.remove_character,
kyu_8.grasshopper_summation.test_summation,	91
93	kyu_8.remove_first_and_last_character.test_remove_character,
kyu_8.greek_sort, 105	91
kyu_8.greek_sort.greek_comparator,	kyu_8.remove_string_spaces, 87
105	kyu_8.remove_string_spaces.remove_string_spaces,
kyu_8.greek_sort.test_greek_comparator,	87
105	kyu_8.remove_string_spaces.test_remove_string_spaces,
kyu_8.holiday_vi_shark_pontoon, 105	87
kyu_8.holiday_vi_shark_pontoon.shark,	kyu_8.reversed_strings, 92
104	kyu_8.reversed_strings.reversed_strings,
kyu_8.holiday_vi_shark_pontoon.test_shark,	91
104	kyu_8.reversed_strings.test_reversed_strings,
kyu_8.is_it_a_palindrome, 99	92
kyu_8.is_it_a_palindrome.is_palindrome,	kyu_8.set_alarm, 96

```

kyu_8.set_alarm.set_alarm, 95
kyu_8.set_alarm.test_set_alarm, 95
kyu_8.surface_area_and_volume_of_box, monkey_count () (in module
92 kyu_8.count_the_monkeys.monkey_count),
kyu_8.surface_area_and_volume_of_box.get_size, 101
92 MonkeyCountTestCase (class in
kyu_8.surface_area_and_volume_of_box.test_get_size, kyu_8.count_the_monkeys.test_monkey_count),
92 101
kyu_8.swap_values, 94 move () (in module kyu_8.terminal_game_move_function.terminal_game_
kyu_8.swap_values.swap_values, 94 88
kyu_8.swap_values.test_swap_values, move_zeros () (in module
94 kyu_5.moving_zeros_to_the_end.move_zeros),
kyu_8.terminal_game_move_function, 22
89 MoveTestCase (class in
kyu_8.terminal_game_move_function.terminal_game_move_function.test_terminal_game_move_
88 89
kyu_8.terminal_game_move_function.test_move_zeros, MoveZerosTestCase (class, in
89 kyu_5.moving_zeros_to_the_end.test_move_zeros),
kyu_8.the_feast_of_many_beasts, 101 22
kyu_8.the_feast_of_many_beasts.feast, multiply () (in module kyu_8.multiply.multiply), 85
100 MultiplyTestCase (class in
kyu_8.the_feast_of_many_beasts.test_feast, kyu_8.multiply.test_multiply), 85
100
kyu_8.third_angle_of_triangle, 91 N
kyu_8.third_angle_of_triangle.test_third_angle_of_triangle, (in module kyu_9.disease_spread.EpidemicTestData
90 property), 51
kyu_8.third_angle_of_triangle.third_angle_of_triangle, (in module
91 kyu_6.format_string_of_names.solution),
kyu_8.well_of_ideas_easy_version, 88 59
kyu_8.well_of_ideas_easy_version.test_well_of_ideas_easy_version, NameListTestCase (class in
87 kyu_6.format_string_of_names.test_namelist),
kyu_8.well_of_ideas_easy_version.well_of_ideas_easy_version, 60
88 next_bigger () (in module
kyu_8.will_there_be_enough_space, 96 kyu_4.next_bigger_number_with_the_same_digits.next_bigger),
kyu_8.will_there_be_enough_space.enough, 16
96 next_greater_digit_index () (in module
kyu_8.will_there_be_enough_space.test_enough, kyu_4.next_bigger_number_with_the_same_digits.next_bigger),
96 16
kyu_8.will_you_make_it, 103 next_smaller () (in module
kyu_8.will_you_make_it.test_zero_fuel, kyu_4.next_smaller_number_with_the_same_digits.next_smaller)
103 15
kyu_8.will_you_make_it.zero_fuel, NextBiggerTestCase (class in
103 kyu_4.next_bigger_number_with_the_same_digits.test_next_bigger)
kyu_8.wolf_in_sheep_clothing, 90 17
kyu_8.wolf_in_sheep_clothing.test_wolf_in_sheep_clothing, NextSmallerTestCase (class in
89 kyu_4.next_smaller_number_with_the_same_digits.test_next_smaller)
kyu_8.wolf_in_sheep_clothing.wolf_in_sheep_clothing, 16
89 no_space () (in module
utils, 106 kyu_8.remove_string_spaces.remove_string_spaces),
utils.log_func, 106 87
utils.primes, 106 Node (class in kyu_7.fun_with_lists_length.node), 71
utils.primes.is_prime, 105 normalize_string () (in module
utils.primes.primes_generator, 106 kyu_2.evaluate_mathematical_expression.evaluate),
utils.primes.test_is_prime, 106 1

```

normalize_string() (in module 84
 kyu_7.significant_figures.number_of_sigfigs), permute_a_palindrome() (in module
 69 kyu_6.permute_a_palindrome.permute_a_palindrome),
 NoSpaceTestCase (class in 42
 kyu_8.remove_string_spaces.test_remove_string_spaces), testPalindromeTestCase (class in
 87 kyu_6.permute_a_palindrome.test_permute_a_palindrome),
 number_of_sigfigs() (in module 42
 kyu_7.significant_figures.number_of_sigfigs), pig_it() (in module kyu_5.simple_pig_latin.pig_it),
 69 19
 NumberOfSigFigsTestCase (class in PigItTestCase (class in
 kyu_7.significant_figures.test_number_of_sigfigs), kyu_5.simple_pig_latin.test_pig_it), 19
 69 pop() (kyu_6.default_list.default_list.DefaultList
 method), 53
 numericals() (in module
 kyu_6.numericals_of_string.numericals), Potion (class in kyu_6.potion_class_101.potion), 50
 39 PotionTestCase (class in
 NumericalsTestCase (class in kyu_6.potion_class_101.test_potion), 50
 kyu_6.numericals_of_string.test_numericals), powers() (in module
 39 kyu_7.sum_of_powers_of_2.sum_of_powers_of_2),
 66
O PrimesTestCase (class in
 odd_row() (in module kyu_5.master_your_primes_sieve_with_memoization.test_primes),
 kyu_6.row_of_the_odd_triangle.odd_row), 25
 49 print_log() (in module utils.log_func), 106
 odd_row() (in module print_map() (in module
 kyu_7.sum_of_odd_numbers.row_sum_odd_numbers), kyu_5.find_the_safest_places_in_town.print_agents),
 81 31
 OddRowTestCase (class in process_brakets() (in module
 kyu_6.row_of_the_odd_triangle.test_odd_row), kyu_2.evaluate_mathematical_expression.evaluate),
 50 1
 OddRowTestCase (class in process_duplicate_minus() (in module
 kyu_7.sum_of_odd_numbers.test_row_sum_odd_numbers), kyu_2.evaluate_mathematical_expression.evaluate),
 81 1
 order() (in module kyu_6.your_order_please.order), process_math_expression() (in module
 56 kyu_2.evaluate_mathematical_expression.evaluate),
 OrderTestCase (class in 1
 kyu_6.your_order_please.test_order), 57 process_not_played_games() (in module
 other_angle() (in module kyu_5.sports_league_table_ranking.compute_ranks),
 kyu_8.third_angle_of_triangle.third_angle_of_triangle), 29
 91 pyramid() (in module
 OtherAngleTestCase (class in kyu_6.pyramid_array.pyramid_array), 38
 kyu_8.third_angle_of_triangle.test_third_angle_of_triangle), PyramidTestCase (class in
 90 kyu_6.pyramid_array.test_pyramid_array),
 38
P pytestmark (kyu_5.count_ip_addresses.test_ips_between.IpsBetweenTest
 attribute), 18
 password() (in module
 kyu_7.password_validator.password), 74
 PasswordTestCase (class in **R**
 kyu_7.password_validator.test_password), randint() (in module
 74 kyu_7.remove_the_minimum.test_remove_the_minimum),
 period_is_late() (in module 64
 kyu_8.is_your_period_late.is_your_period_late), random_list() (kyu_7.remove_the_minimum.test_remove_the_minimum
 83 static method), 64
 PeriodIsLateTestCase (class in rank() (kyu_4.the_greatest_warrior.warrior.Warrior
 kyu_8.is_your_period_late.test_is_your_period_late), property), 14

140 Index


```

    kyu_4.strings_mix.solution), 15
sort_results() (in module kyu_8.grasshopper_summation.test_summation), 93
    kyu_6.rotate_the_letters_of_each_element.group_bits(), 55
SortArrayTestCase (class in kyu_6.sort_the_odd.test_sort_array), 60
    kyu_6.sort_the_odd.test_sort_array), 60
spiralize() (in module kyu_3.make_spiral.solution), 5
    kyu_6.sort_the_odd.test_sort_array), 60
SpiralizeTestCase (class in kyu_3.make_spiral.test_spiralize), 5
    kyu_6.sort_the_odd.test_sort_array), 60
stock_list() (in module kyu_6.help_the_bookseller.stock_list), 49
    kyu_6.sort_the_odd.test_sort_array), 60
StockListTestCase (class in kyu_6.help_the_bookseller.test_stock_list), 49
    kyu_6.sort_the_odd.test_sort_array), 60
string_to_array() (in module kyu_8.convert_string_to_an_array.string_to_array), 99
    kyu_6.sort_the_odd.test_sort_array), 60
string_transformer() (in module kyu_6.string_transformer.string_transformer), 44
    kyu_6.sort_the_odd.test_sort_array), 60
StringIncrementerTestCase (class in kyu_5.string_incrementer.test_increment_string), 35
    kyu_6.sort_the_odd.test_sort_array), 60
StringToArrayTestCase (class in kyu_8.convert_string_to_an_array.test_string_to_array), 100
    kyu_6.sort_the_odd.test_sort_array), 60
StringTransformerTestCase (class in kyu_6.string_transformer.test_string_transformer), 45
    kyu_6.sort_the_odd.test_sort_array), 60
Sudoku (class in kyu_4.validate_sudoku_with_size.sudoku), 10
    kyu_6.sort_the_odd.test_sort_array), 60
SudokuTestCase (class in kyu_4.validate_sudoku_with_size.test_sudoku), 11
    kyu_6.sort_the_odd.test_sort_array), 60
sum_for_list() (in module kyu_4.sum_by_factors.sum_for_list), 12
    kyu_6.sort_the_odd.test_sort_array), 60
sum_of_intervals() (in module kyu_4.sum_of_intervals.sum_of_intervals), 7
    kyu_6.sort_the_odd.test_sort_array), 60
sum_pairs() (in module kyu_5.sum_of_pairs.sum_pairs), 33
    kyu_6.sort_the_odd.test_sort_array), 60
sum_triangular_numbers() (in module kyu_7.sum_of_triangular_numbers.sum_triangular_numbers), 67
    kyu_6.sort_the_odd.test_sort_array), 60
sum_two_smallest_numbers() (in module kyu_7.sum_of_two_lowest_int.sum_two_smallest_int), 65
    kyu_6.sort_the_odd.test_sort_array), 60
SumForListTestCase (class in kyu_4.sum_by_factors.test_sum_for_list), 12
    kyu_6.sort_the_odd.test_sort_array), 60
summation() (in module kyu_8.grasshopper_summation.summation), 93
    kyu_6.sort_the_odd.test_sort_array), 60
SummationTestCase (class in kyu_8.grasshopper_summation.test_summation), 93
    kyu_6.sort_the_odd.test_sort_array), 60
SumOfIntervalsTestCase (class in kyu_4.sum_of_intervals.test_sum_of_intervals), 7
    kyu_6.sort_the_odd.test_sort_array), 60
SumOfNumbersTestCase (class in kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers), 62
    kyu_6.sort_the_odd.test_sort_array), 60
SumOfPowerOfTwoTestCase (class in kyu_7.sum_of_powers_of_2.test_sum_of_powers_of_2), 67
    kyu_6.sort_the_odd.test_sort_array), 60
SumPairsTestCase (class in kyu_5.sum_of_pairs.test_sum_pairs), 33
    kyu_6.sort_the_odd.test_sort_array), 60
SumTriangularNumbersTestCase (class in kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers), 68
    kyu_6.sort_the_odd.test_sort_array), 60
SumTwoSmallestNumbersTestCase (class in kyu_7.sum_of_two_lowest_int.test_sum_two_smallest_numbers), 65
    kyu_6.sort_the_odd.test_sort_array), 60
swap_values() (in module kyu_8.swap_values.swap_values), 94
    kyu_6.sort_the_odd.test_sort_array), 60
SwapValuesTestCase (class in kyu_8.swap_values.test_swap_values), 94
    kyu_6.sort_the_odd.test_sort_array), 60
T
    kyu_8.convert_string_to_an_array.test_string_to_array), 100
TakeTestCase (class in kyu_8.enumerable_magic_25.test_take), 102
    kyu_6.sort_the_odd.test_sort_array), 60
TakeTestCase (class in kyu_8.enumerable_magic_25.test_take), 102
    kyu_6.sort_the_odd.test_sort_array), 60
test_agents_cleanup() (in module kyu_5.find_the_safest_places_in_town.test_advice.FirstAdviceTest), 31
    kyu_6.sort_the_odd.test_sort_array), 60
test_all_fibonacci_numbers() (in module kyu_5.fibonacci_streaming.test_all_fibonacci_numbers.AllFibonacci), 17
    kyu_6.sort_the_odd.test_sort_array), 60
test_alphabet_war() (in module kyu_5.alphabet_wars_nuclear_strike.test_alphabet_war.AlphabetWar), 21
    kyu_6.sort_the_odd.test_sort_array), 60
test_alphanumeric() (in module kyu_5.not_very_secure.test_alphanumeric.AlphanumericTestCase), 19
    kyu_6.sort_the_odd.test_sort_array), 60
test_alternating_case() (in module kyu_8.alternating_case.test_alternating_case.AlternatingTestCase), 93
    kyu_6.sort_the_odd.test_sort_array), 60
test_anagrams() (in module kyu_5.where_my_anagrams_at.test_anagrams.Anagrams), 24
    kyu_6.sort_the_odd.test_sort_array), 60
test_array_diff_function() (in module kyu_6.array_diff.test_array_diff.ArrayDiffTestCase), 61
    kyu_6.sort_the_odd.test_sort_array), 60
test_battle() (in module kyu_4.the_greatest_warrior.test_battle.BattleTestCase), 13
    kyu_6.sort_the_odd.test_sort_array), 60
test_binary_to_string() (in module kyu_6.binary_to_text_ascii_conversion.test_binary_to_string.BinaryToTextAsciiConversion), 13
    kyu_6.sort_the_odd.test_sort_array), 60

```

```

        method), 47
test_calc() (kyu_2.evaluate_mathematical_expression.test_evaluate_mathematical_expression.method), 2
test_calc_combinations_per_row() (kyu_7.easy_line.test_easylines.EasyLineTestCase method), 82
test_calculate() (kyu_7.basic_math_add_or_subtract.test_calculate_add_or_subtract.method), 81
test_calculate() (kyu_7.help_bob_count_letters_and_digits.test_count_letters_and_digits.CalculateTestCase method), 82
test_calculate_damage() (kyu_6.pokemon_damage_calculator.test_calculate_damage.CalculateDamageTestCase method), 48
test_calculator() (kyu_3.calculator.test_calculator.CalculatorTestCase method), 3
test_century() (kyu_8.century_from_year.test_century.CenturyTestCase method), 104
test_check_exam() (kyu_8.check_the_exam.test_check_exam.CheckExamTestCase method), 98
test_check_for_factor_false() (kyu_8.grasshopper_check_for_factor.test_check_for_factor.CheckForFactorTestCase method), 97
test_check_for_factor_true() (kyu_8.grasshopper_check_for_factor.test_check_for_factor.CheckForFactorTestCase method), 97
test_check_root() (kyu_7.always_perfect.test_check_root.CheckRootTestCase method), 76
test_checkchoose() (kyu_6.color_choice.test_checkchoose.CheckchooseTestCase method), 53
test_count_letters_in_string() (kyu_6.count_letters_in_string.test_count_letters_in_string.CountLettersInStringTestCase method), 42
test_counting_sheep() (kyu_8.counting_sheep.test_counting_sheep.CountingSheepTestCase method), 97
test_counting_sheep_bad_input() (kyu_8.counting_sheep.test_counting_sheep.CountingSheepTestCase method), 97
test_counting_sheep_empty_list() (kyu_8.counting_sheep.test_counting_sheep.CountingSheepTestCase method), 97
test_counting_sheep_mixed_list() (kyu_8.counting_sheep.test_counting_sheep.CountingSheepTestCase method), 97
test_create_city_map() (kyu_5.find_the_safest_places_in_town.test_advice.FirstAdviceTestCase method), 31
test_decipher_this() (kyu_6.decipher_this.test_decipher_this.DecipherThisTestCase method), 58
test_decoding() (kyu_3.rail_fence_cipher_encoding_and_decoding.test_decoding.method), 4
test_default_list_append() (kyu_6.default_list.test_default_list.DefaultListTestCase method), 53
test_default_list_basic() (kyu_6.default_list.test_default_list.DefaultListTestCase method), 54
test_default_list_pop() (kyu_6.default_list.test_default_list.DefaultListTestCase method), 54
test_default_list_remove() (kyu_6.default_list.test_default_list.DefaultListTestCase method), 54
test_digital_root() (kyu_6.sum_of_digits_digital_root.test_digital_root.DigitalRootTestCase method), 98
test_directions_reduction() (kyu_5.directions_reduction.test_directions_reduction.DirectionsReductionTestCase method), 97
test_disemvowel() (kyu_7.disemvowel_trolls.test_disemvowel_trolls.DisemvowelTestCase method), 62
test_domain_name() (kyu_5.extract_the_domain_name_from_url.test_domain_name.DomainNameTestCase method), 32
test_done_or_not() (kyu_5.did_i_finish_my_sudoku.test_did_i_finish_sudoku.DidIFinishSudokuTestCase method), 97
test_duplicate_encode() (kyu_6.duplicate_encoder.test_duplicate_encode.DuplicateEncoderTestCase method), 43
test_easy_diagonal() (kyu_6.easy_diagonal.test_diagonal.EasyDiagonalTestCase method), 54
test_easy_line() (kyu_7.easy_line.test_easylines.EasyLineTestCase method), 82
test_exception() (kyu_7.easy_line.test_easylines.EasyLineTestCase method), 82
test_encrypt_this() (kyu_3.rail_fence_cipher_encoding_and_decoding.test_encrypt_this.method), 4
test_encrypt_this() (kyu_5.find_the_safest_places_in_town.test_advice.FirstAdviceTestCase method), 31
test_enough() (kyu_8.will_there_be_enough_space.test_enough.EnoughSpaceTestCase method), 96
test_epidemic() (kyu_6.disease_spread.test_epidemic.EpidemicTestCase method), 97

```

method), 52
test_factorial() (kyu_7.factorial.test_factorial.FactorialTestCase method), 79
method), 78
test_feast() (kyu_8.the_feast_of_many_beasts.test_feast.FeastTestCase method), 100
method), 106
test_find_missing_number() (kyu_6.number_zoo_patrol.test_find_missing_number.FindMissingNumberTestCase method), 56
method), 106
test_first_alpha_only() (kyu_6.first_character_that_repeats.test_first_character_that_repeats.FirstAlphaOnlyTestCase method), 37
test_first_dup_mixed() (kyu_6.first_character_that_repeats.test_first_character_that_repeats.FirstDupMixedTestCase method), 37
test_first_dup_none() (kyu_6.first_character_that_repeats.test_first_character_that_repeats.FirstDupNoneTestCase method), 37
test_first_no_alpha() (kyu_6.first_character_that_repeats.test_first_character_that_repeats.FirstNoAlphaTestCase method), 37
test_first_non_consecutive_large_list() (kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive_large_list.FirstNonConsecutiveLargeListTestCase method), 90
test_first_non_consecutive_negative() (kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive_negative.FirstNonConsecutiveNegativeTestCase method), 90
test_first_non_consecutive_none() (kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive_none.FirstNonConsecutiveNoneTestCase method), 90
test_first_non_consecutive_positive() (kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive_positive.FirstNonConsecutivePositiveTestCase method), 90
test_first_non_repeated() (kyu_7.the_first_non_repeated_character_in_string.test_first_non_repeated.FirstNonRepeatedTestCase method), 72
test_first_non_repeating_letter() (kyu_5.find_the_safest_places_in_town.test_advice.FirstAdviceTestCase method), 31
test_first_non_repeating_letter() (kyu_5.first_non_repeating_character.test_first_non_repeating_letter.FirstNonRepeatingLetterTestCase method), 27
test_first_space() (kyu_6.first_character_that_repeats.test_first_character_that_repeats.FirstSpaceTestCase method), 37
test_fix_the_meerkat() (kyu_8.my_head_is_at_the_wrong_end.test_fix_the_meerkat.FixTheMeerkatTestCase method), 94
test_flatten() (kyu_5.flatten.test_flatten.FlattenTestCase method), 26
test_flatten() (kyu_5.integers_recreation_one.test_list_squared_integers.OneTestCase method), 36
test_format_duration() (kyu_4.human_readable_duration_format.test_format_duration.FormatDurationTestCase method), 9
test_gap() (kyu_7.find_the_longest_gap.test_gap.GapTestCase method), 79
test_gen_primes_negative() (kyu_8.gen_primes.test_primes_generator.GenPrimesTestCase method), 106
test_gen_primes_positive() (kyu_8.gen_primes.test_primes_generator.GenPrimesTestCase method), 106
test_generate_hashtag() (kyu_5.generate_hashtag.test_generate_hashtag.GenerateHashtagTestCase method), 33
test_get_size() (kyu_8.surface_area_and_volume_of_box.test_get_size.SurfaceAreaAndVolumeOfBoxTestCase method), 9
test_get_sum_equal_numbers() (kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers.SumOfNumbersTestCase method), 6
test_get_sum_negative_numbers() (kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers.SumOfNumbersTestCase method), 6
test_get_sum_positive_numbers() (kyu_7.beginner_series_sum_of_numbers.test_sum_of_numbers.SumOfNumbersTestCase method), 6
test_goals() (kyu_8.grasshopper_messi_goals_function.test_messi_goals.GrasshopperMessiGoalsFunctionTestCase method), 87
test_group_cities() (kyu_8.find_the_first_non_consecutive_number.test_first_non_consecutive_group_cities.FirstNonConsecutiveGroupCitiesTestCase method), 56
test_growing_plant() (kyu_7.growing_plant.test_growing_plant.GrowingPlantTestCase method), 80
test_has_subpattern() (kyu_5.find_the_safest_places_in_town.test_advice.FirstAdviceTestCase method), 40
test_has_subpattern() (kyu_5.first_non_repeating_character.test_first_non_repeating_letter.FirstNonRepeatingLetterTestCase method), 41
test_has_subpattern() (kyu_6.find_the_first_non_repeating_letter.test_has_subpattern.HasSubpatternTestCase method), 41
test_hoop_count_negative() (kyu_8.keep_up_the_hoop.test_hoop_count.HoopCountTestCase method), 102
test_hoop_count_positive() (kyu_8.keep_up_the_hoop.test_hoop_count.HoopCountTestCase method), 102
test_horizontally() (in module kyu_4.sudoku_solution_validator.valid_solution), 10
test_if_team_registered() (in module

```

    kyu_5.sports_league_table_ranking.compute_ranking.test_length_none()
29    (kyu_7.fun_with_lists_length.test_length.LengthTestCase
test_increment_string()    method), 71
    (kyu_5.string_incrementer.test_increment_string.StringIncrementerTestCase
    method), 35
test_invite_more_women_negative()    (kyu_6.character_frequency.test_character_frequency.LetterFrequency
    method), 39
    (kyu_7.simple_fun_152.test_invite_more_women.InviteMoreWomenTestCase
    method), 69
test_invite_more_women_positive()    (kyu_6.character_frequency.test_character_frequency.LetterFrequency
    method), 39
    (kyu_7.simple_fun_152.test_invite_more_women.InviteMoreWomenTestCase
    method), 69
test_ips_between()    (kyu_6.character_frequency.test_character_frequency.LetterFrequency
    method), 39
    (kyu_5.count_ip_addresses.test_ips_between.IpsBetweenTestCase
    method), 18
test_is_isogram()    (kyu_6.who_likes_it.test_likes_function.LikesTestCase
    method), 58
    (kyu_7.isograms.test_is_isogram.IsIsogramTestCase
    method), 83
test_is_palindrome()    (kyu_8.logical_calculator.test_logical_calculator.LogicalCalculator
    method), 84
    (kyu_8.is_it_a_palindrome.test_is_palindrome.IsPalindromeTestCase
    method), 99
test_is_prime_negative()    (kyu_8.logical_calculator.test_logical_calculator.LogicalCalculator
    method), 84
    (utils.primes.test_is_prime.IsPrimeTestCase
    method), 106
test_is_prime_positive()    (kyu_8.logical_calculator.test_logical_calculator.LogicalCalculator
    method), 85
    (utils.primes.test_is_prime.IsPrimeTestCase
    method), 106
test_is_solved() (kyu_5.tic_tac_toe_checker.test_checker.IsSolvedTestCase
    method), 34
test_is_square_25()    (kyu_8.logical_calculator.test_logical_calculator.LogicalCalculator
    method), 85
    (kyu_7.make_class.test_make_class.MakeClassTestCase
    method), 74
    (kyu_7.you_are_square.test_you_are_square.YouAreSquareTestCase
    method), 66
test_is_square_26()    (kyu_8.logical_calculator.test_logical_calculator.LogicalCalculator
    method), 85
    (kyu_5.human_readable_time.test_make_readable.MakeReadableTestCase
    method), 20
    (kyu_7.you_are_square.test_you_are_square.YouAreSquareTestCase
    method), 66
test_is_square_four()    (kyu_8.make_upper_case.test_make_upper_case.MakeUpperCaseTestCase
    method), 88
    (kyu_7.you_are_square.test_you_are_square.YouAreSquareTestCase
    method), 66
test_is_square_negative_numbers()    (kyu_7.sort_out_the_men_from_boys.test_men_from_boys.MenFromBoysTestCase
    method), 70
    (kyu_7.you_are_square.test_you_are_square.YouAreSquareTestCase
    method), 66
test_is_square_negative_test()    (kyu_8.count_the_monkeys.test_monkey_count.MonkeyCountTestCase
    method), 101
    (kyu_7.you_are_square.test_you_are_square.YouAreSquareTestCase
    method), 66
test_is_square_zero()    (kyu_8.move_zeros_to_the_end.test_move_zeros.MoveZerosTestCase
    method), 22
    (kyu_7.you_are_square.test_you_are_square.YouAreSquareTestCase
    method), 66
test_keep_hydrated()    (kyu_5.moving_zeros_to_the_end.test_move_zeros.MoveZerosTestCase
    method), 22
    (kyu_8.keep_hydrated.test_keep_hydrated.KeepHydratedTestCase
    method), 95
test_largest_power()    (kyu_8.multiply.test_multiply.MultiplyTestCase
    method), 85
    (kyu_7.powers_of_3.test_largest_power.LargestPowerTestCase
    method), 67
test_length() (kyu_6.format_string_of_names.test_namelist.NamelistTestCase
    method), 60
    (kyu_7.fun_with_lists_length.test_length.LengthTestCase
    method), 71
    (kyu_8.bigger_number_with_the_same_digits.test_next_bigger_number_with_the_same_digits.test_next_bigger
    method), 17

```

test_next_smaller() (kyu_4.next_smaller_number_with_the_same_digits.test_next_smaller.NextSmallerTestCase method), 64
 test_number_of_sigfigs() (kyu_7.significant_figures.test_number_of_sigfigs.NumberOfSigFigsTestCase method), 64
 test_numericals() (kyu_6.numericals_of_string.test_numericals.NumericalsTestCase method), 92
 test_odd_row() (kyu_6.row_of_the_odd_triangle.test_odd_row.OddRowTestCase method), 50
 test_order() (kyu_6.your_order_please.test_order.OrderTestCase method), 57
 test_other_angle() (kyu_8.third_angle_of_triangle.test_third_angle_of_triangle.OtherAngleTestCase method), 90
 test_password() (kyu_7.password_validator.test_password_validator.PasswordValidatorTestCase method), 74
 test_period_is_late_negative() (kyu_8.is_your_period_late.test_is_your_period_late.PeriodIsLateTestCase method), 84
 test_period_is_late_positive() (kyu_8.is_your_period_late.test_is_your_period_late.PeriodIsLateTestCase method), 84
 test_permute_a_palindrome_empty_string() (kyu_6.permute_a_palindrome.test_permute_a_palindrome.PermutePalindromeTestCase method), 42
 test_permute_a_palindrome_negative() (kyu_6.permute_a_palindrome.test_permute_a_palindrome.PermutePalindromeTestCase method), 42
 test_permute_a_palindrome_positive() (kyu_6.permute_a_palindrome.test_permute_a_palindrome.PermutePalindromeTestCase method), 42
 test_pig_it() (kyu_5.simple_pig_latin.test_pig_it.PigItTestCase method), 19
 test_potion() (kyu_6.potion_class_101.test_potion.PotionTestCase method), 50
 test_powers() (kyu_7.sum_of_powers_of_2.test_sum_of_powers_of_2.SumOfPowersOfTwoTestCase method), 67
 test_primes() (kyu_5.master_your_primes_sieve_with_memoisation.test_master_your_primes_sieve_with_memoisation.MasterYourPrimesSieveWithMemoisationTestCase method), 25
 test_pyramid() (kyu_6.pyramid_array.test_pyramid_array.PyramidTestCase method), 38
 test_remove_char() (kyu_8.remove_first_and_last_character.test_remove_char.RemoveCharTestCase method), 91
 test_remove_smallest() (kyu_7.remove_the_minimum.test_remove_the_minimum.RemoveSmallestTestCase method), 64
 test_remove_smallest_empty_list() (kyu_7.remove_the_minimum.test_remove_the_minimum.RemoveSmallestTestCase method), 64
 test_remove_smallest_one_element_list() (kyu_7.remove_the_minimum.test_remove_the_minimum.RemoveSmallestTestCase method), 64
 test_reversed_strings() (kyu_8.reversed_strings.test_reversed_strings.ReversedStringsTestCase method), 92
 test_reversed_strings_empty() (kyu_8.reversed_strings.test_reversed_strings.ReversedStringsTestCase method), 92
 test_reversed_strings_one_char() (kyu_8.reversed_strings.test_reversed_strings.ReversedStringsTestCase method), 92
 test_row_sum_odd_numbers() (kyu_8.third_angle_of_triangle.test_third_angle_of_triangle.OddRowSumOddNumbersTestCase method), 81
 test_save() (kyu_7.fill_the_hard_disk_drive.test_save.SaveTestCase method), 72
 test_sentence() (kyu_7.pull_your_words_together_man.test_sentence.SentenceTestCase method), 78
 test_set_alarm() (kyu_6.set_alarm.test_set_alarm.SetAlarmTestCase method), 95
 test_shark_alive_1() (kyu_8.holiday_vi_shark_pontoon.test_shark.SharkTestCase method), 104
 test_shark_alive_2() (kyu_8.holiday_vi_shark_pontoon.test_shark.SharkTestCase method), 104
 test_shark_bait() (kyu_8.holiday_vi_shark_pontoon.test_shark.SharkTestCase method), 104
 test_snail() (kyu_4.snail.test_snail.SnailTestCase method), 12
 test_solve() (kyu_6.casino_chips.test_solve.SolveTestCase method), 47
 test_solve_league_table_ranking() (kyu_6.casino_chips.test_solve.SolveTestCase method), 47


```

        method), 29
test_something() (kyu_6.find_the_odd_int.test_find_the_odd_int.FindTheOddIntTestCase divisibility_by_13.test_thirt.ThirdTestC
        method), 37
test_something() (kyu_8.remove_string_spaces.test_remove_string_spaces.NoSpaceTestCase.test_tickets.TicketsTestCase
        method), 87
test_sort_array() (kyu_6.sort_the_odd.test_sort_array.SortArrayTestCase test_to_jaden_case_negative()
        method), 60
test_spiralize() (kyu_3.make_spiral.test_spiralize.SpiralizeTestCase (kyu_7.jaden_casing_strings.test_jaden_casing_strings.JadenCas
        method), 5
test_stock_list() (kyu_6.help_the_bookseller.test_stock_list.StockListTestCase (kyu_7.jaden_casing_strings.test_jaden_casing_strings.JadenCas
        method), 49
test_string_to_array() (kyu_8.convert_string_to_an_array.test_string_to_array.StringToArrayTestCase test_jaden_case_positive()
        method), 100
test_string_transformer() (kyu_6.string_transformer.test_string_transformer.StringTransformerTestCase (kyu_7.jaden_casing_strings.test_jaden_casing_strings.JadenCas
        method), 45
test_sub_grids() (in module kyu_4.sudoku_solution_validator.valid_solution), test_unique_in_order() (kyu_4.top_3_words()
        method), 99
test_sudoku_class() (kyu_4.validate_sudoku_with_size.test_sudoku.SudokuTestCase (kyu_4.top_3_words()
        method), 11
test_sum_for_list() (kyu_4.sum_by_factors.test_sum_for_list.SumForListTestCase (kyu_4.top_3_words()
        method), 12
test_sum_of_intervals() (kyu_4.sum_of_intervals.test_sum_of_intervals.SumOfIntervalsTestCase (kyu_4.top_3_words()
        method), 7
test_sum_pairs() (kyu_5.sum_of_pairs.test_sum_pairs.SumPairsTestCase (kyu_4.top_3_words()
        method), 33
test_sum_triangular_numbers_big_number() (kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers.SumTriangularNumbersTestCase
        method), 68
test_sum_triangular_numbers_negative_numbers() (kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers.SumTriangularNumbersTestCase
        method), 68
test_sum_triangular_numbers_positive_numbers() (kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers.SumTriangularNumbersTestCase
        method), 68
test_sum_triangular_numbers_zero() (kyu_7.sum_of_triangular_numbers.test_sum_triangular_numbers.SumTriangularNumbersTestCase
        method), 68
test_sum_two_smallest_numbers() (kyu_7.sum_of_two_lowest_int.test_sum_two_smallest_numbers.SumTwoSmallestNumbersTestCase
        method), 65
test_summation() (kyu_8.grasshopper_summation.test_summation.SummationTestCase (kyu_4.the_greatest_warrior.test_warrior.WarriorTestCase
        method), 93
test_swap_values() (kyu_8.swap_values.test_swap_values.SwapValuesTestCase (kyu_4.the_greatest_warrior.test_warrior.WarriorTestCase
        method), 94
test_take() (kyu_8.enumerable_magic_25.test_take.TakeTestCase method), 87

```

test_well_publish()	(kyu_8.well_of_ideas_easy_version.test_well_of_ideas_easy_version.method),	87
test_well_series()	(kyu_8.well_of_ideas_easy_version.test_well_of_ideas_easy_version.method),	87
test_zero_fuel()	(kyu_8.will_you_make_it.test_zero_fuel.ZeroFuelTestCase.method),	103
test_zeros()	(kyu_5.number_of_trailing_zeros_of_n.test_zeros.ZerosTestCase.method),	25
third()	(in module kyu_6.a_rule_of_divisibility_by_13.third),	52
ThirdTestCase	(class in kyu_6.a_rule_of_divisibility_by_13.test_third),	52
tickets()	(in module kyu_6.vasya_clerk.tickets),	44
TicketsTestCase	(class in kyu_6.vasya_clerk.test_tickets),	44
tm()	(kyu_6.disease_spread.epidemic_test_data.EpidemicTestData.property),	51
to_alternating_case()	(in module kyu_8.alternating_case.alternating_case),	92
to_table()	(in module kyu_6.array_to_html_table.to_table),	55
toJadenCase()	(in module kyu_7.jaden_casing_strings.jaden_casing_strings),	63
Top3WordsTestCase	(class in kyu_4.most_frequently_used_words.test_top_3_words),	13
top_3_words()	(in module kyu_4.most_frequently_used_words.solution),	13
training()	(kyu_4.the_greatest_warrior.warrior.Warrior.method),	14
two_decimal_places()	(in module kyu_7.formatting_decimal_places_1.two_decimal_places),	76
two_decimal_places()	(in module kyu_8.formatting_decimal_places_0.two_decimal_places),	99
TwoDecimalPlacesTestCase	(class in kyu_7.formatting_decimal_places_1.test_two_decimal_places),	76
TwoDecimalPlacesTestCase	(class in kyu_8.formatting_decimal_places_0.test_two_decimal_places),	99
unique_in_order()	(in module kyu_6.unique_in_order.unique_in_order),	43

```

14
well() (in module kyu_8.well_of_ideas_easy_version.well_of_ideas_easy_version),
88
WellTestCase (class in
    kyu_8.well_of_ideas_easy_version.test_well_of_ideas_easy_version),
87
word_processor() (in module
    kyu_5.simple_pig_latin.pig_it), 19

```

Y

```

YouAreSquareTestCase (class in
    kyu_7.you_are_square.test_you_are_square),
66

```

Z

```

zero_fuel() (in module
    kyu_8.will_you_make_it.zero_fuel), 103
ZeroFuelTestCase (class in
    kyu_8.will_you_make_it.test_zero_fuel),
103
zeros() (in module
    kyu_5.number_of_trailing_zeros_of_n.zeros),
26
ZerosTestCase (class in
    kyu_5.number_of_trailing_zeros_of_n.test_zeros),
25

```