homeworks.md

How to send your homework?

1. Create solutions for your tasks.
2. Put each solution in separate python file named `<homework_number>_<task_number>.py`. For example solution for the task **3.1.** should be placed in a file `3_1.py`. If you are working with Jupyter, don't worry – after creating solutions just copy them into new files with a proper name.
3. Put all of your solutions in the folder named with your index number. For example, if your University index number is *333444*, folder should be named `333444`.
4. Compress your files using zip extension (without password protection). Finally your file should be `333444.zip`.

For example if your index is *333444* and you are sending solutions for the first homework your files should have structure (after unzipping `333444.zip`):

```
333444
├── 1_1.py
├── 1_2.py
└── 1_3.py
```

Homework 1.

1.1. Write a script (or function) that takes as an input two numbers `width` and `height` and prints a rectangle with specified size.

Examples:

- `width=5` and `height=4` should result in terminal output:

```
#####
#   #
#   #
#####
```

- `width=7` and `height=2` should result in terminal output:

```
#######
#######
```

- `width=1` and `height=1` should result in terminal output:

```
#
```

1.2. Sum all perfect squares (numbers that are equal to the square of other number – 1, 4, ..., 9, 16, 25, 36, 49, ..., 9801, 10000) in range between 0 and 10000.

1.3. Write a script for setting an alarm, which ask users whether they are employed (yes / no) and whether they are currently on vacation (yes / no). User should answer typing either `Y` for yes or `N` for no. If user specify incorrect answer (anything that is not `Y` or `N`) program should warn user about incorrect answer and ask again.

The script output `True` if user is employed and not on vacation (because these are the circumstances under which you need to set an alarm). It should output `False` otherwise. Examples:

Examples:

- setting an alarm

```
> Are you employed? (Y/N):
> y
> Incorrect answer.
> Are you employed? (Y/N):
> Y
> Are you on vacation (Y/N):
> N
> True
```

- not setting an alarm

```
> Are you employed? (Y/N):
> N
> Are you on vacation (Y/N):
> N
> False
```

Homework 2.

2.1. Write a script approximating `cos(x)` as a first five terms of Taylor series expansion (see trigonometric functions section in Wikipedia). User should provide an input value of `x` and script should output calculated value for each term as well as current sum. Output should be formatted according to scheme:

```
> Calculating cos(1.0471975511965976) as Taylor expansion...
>
> Value for k=0 is +1.0000 (current sum is 1.0000)
> Value for k=1 is -0.5483 (current sum is 0.4517)
> Value for k=2 is +0.0501 (current sum is 0.5018)
> Value for k=3 is -0.0018 (current sum is 0.5000)
> Value for k=4 is +0.0000 (current sum is 0.5000)
```

Use either the `.format()` method or f-strings.

> You might need `factorial` function from `math` module

2.2. Write two scripts `coefficients.py` and `quadratic.py`. In `coefficients.py` define three variables `a`, `b` and `c` representing coefficients of the quadratic equation `ax^2 + bx + c`. In `quadratic.py` import coefficients, solve the equation and output properly formatted the solution. For example in `coefficients.py` define:

```
a = 1
b = -10
c = 25
```

Then running `quadratic.py` should produce:

```
> One solution found x=5.000
```

Output for two solutions should look like this:

```
> Two solutions found x=-2.414 and x=0.414
```

Output for no solutions (when delta is less than zero):

```
> No solutions found
```

> Imports will work only when these two files will be placed within the same directory

2.3. Write a script that takes an input string from the user and transforms it according to three rules:

1. delete all vowels,

2. leave all consonants and place `.` (dot) after them
3. transform all uppercase letters for lowercase letters

Examples:

- input `Programming` should be transformed into `p.r.g.r.m.m.n.g.`
- input `ABACC` should be transformed into `b.c.c.`
- input `aaa` should be transformed into an empty string

> Consider only inputs consisting of lowercase and uppercase ascii letters (a-z) and (A-Z)

Homework 3.

3.1. Write a function `coins(value)` which calculate coin combinations for a given value (amount of gr). For example if we want to pay 62gr, we have to prepare 1 x 50gr coin, 1 x 10gr coin and 1 x 2gr coin. Function should return a list with 6 elements: 1st element corresponding to the number of 50gr coins, 2nd element corresponding to the number of 20gr coins, and so on up to 6th element corresponding to the number of 1gr coins.

In this example function `coins(62)` should output `[1, 0, 1, 0, 1, 0]` because we have 1 x 50gr, 0 x 20gr, 1 x 10gr, 0 x 5gr, 1 x 2gr and 0 x 1gr.

More examples:

```
coins(70) # should return [1, 1, 0, 0, 0, 0]
coins(3) # should return [0, 0, 0, 0, 1. 1]
```

> `value` will be an integer between 0 and 99

3.2.

Imagine you are a secret agent and receive secret messages from your agency. Secret message is a string of random-looking characters like `kj4656%cwj4342dm`. Your job is to determine if you should change your location based on the secret message. If the first instance letter "c" in the string is immediately followed with the number "01" (operational code for changing location) you should change your location. Write a function `decipher(secret_string)` which returns either `True` or `False` based on the decision if you should change your location.

Examples:

```
decipher('qweqw34%c013fewca') # should return True, because first occurance of 'c' is followed by '01'
decipher('qceqw34%c013fewca') # should return False, because first occurance of 'c' is followed by 'eq'
decipher('sdwe6t544^d&fda65') # should return False, because there is no 'c' in secret string
```

Homework 4.

4.1. Write a function `moving_average(array, n)` that takes two inputs: `array` which is a list of numbers and `n` which is an integer. Return new list with values being a moving average with window size `n`. Moving average is an average calculated on `n` subsequent elements. For example:

```
array = [2, 2, 4, 5, 2]
n = 3 # window size
```

In this case moving average can be calculated as:

```
2, 2, 4, 5, 2
|     |
^^^^^^^
(2+2+4)/3 = 2.67


2, 2, 4, 5, 2
```

```
   |     |
   ^^^^^^^
   (2+4+5)/3 = 3.67


2, 2, 4, 5, 2
      |     |
      ^^^^^^^
      (4+5+2)/3 = 3.67
```

so the function should oputut `[2.67, 3.67, 3.67]`.

In the cases when moving average cannot be calculated, e.g. when `n` > `len(array)`, return `None`.

4.2. The dragon's curve is a self-similar fractal which can be obtained by a recursive method.

Starting with the initial string 'Fa', at each step simultaneously perform the following operations:

- first replace `'a'` with: `'aRbFR'`
- then replace `'b'` with: `'LFaLb'`

This operations transforms the string `'Fa'` into `'FaRbFR'` at first iteration, and string `'FaRbFR'` into `'FaRbFRRLFaLbFR'` at second iteration (and so on). After desired number of iterations, remove letters `'a'` and `'b'` from the output string. You will have a string with only `'R'`, `'L'`, and `'F'`. The goal of this task is to write a function `dragon(n)` wich takes one parameter `n`, the number of iterations needed and return the string of instruction as defined above.

Examples:

- `dragon(0)` should return `'F'`
- `dragon(1)` should return `'FRFR'`
- `dragon(2)` should return `'FRFRRLFLFR'`

> *Additional note*: Output string is a set of instruction. Starting at the origin of a grid looking in the (0, 1) direction, 'F' means a step forward, 'L' and 'R' mean respectively turn left and right. After executing all instructions, the trajectory will give a beautifull self-replicating pattern called Dragon Curve.

Homework 5.

5.1. You are given an array of positive ints where every element appears two times, except one that appears only once (let's call it x) and one that appears three times (let's call it y).

Your task is to write function `appear(array)` that takes list `array` as an input and returns x * x * y.