**Python summary exercises**

Where is relevant add a test module to check your program and write tests using pytest format.

**Exercise 1:**

Write a Python program to combine each line from first file(‘file\_A.txt’) with the corresponding line in second file(‘file\_B.txt’).

Write the result into a new json file called ‘combination.json’

of following format:

code for example:

for giving the input files:

**file\_A.txt:**

Hello girls!

Today is Tuesday

Good luck!!!

**file\_B.txt:**

We are learning python

The best language!!:)

The output file will be:

**combination.json**

{

"line1" : ["Hello girls!", "We are learning python"],

"line2" : ["Today is Tuesday", "The best language!!:)"],

"line3" : ["Good luck!!!"],

}

The files need not have the same number of lines. See example

Function prototype: def merge(source1, source2, dest, lines1=[], lines2=[])  
source1 is the first source file  
source2 is the second source file  
dest is the destination file  
lines1 provides a list of the lines to extract from source1; the empty list represents all of the lines  
lines2 provides a list of the lines to extract from source2; the empty list represents all of the lines

You should catch errors in line numbers (if the line number exceeds what is in the file

**Exercise 2:**

## **Password Validation**

1. Write a script to check the validity of a password inputed by users.

Validation rules:

* At least 4 lower case letters, and 2 Capital letters.
* At least 1 number.
* At least 1 character from [$#!%@].
* Minimum length of 8 characters.
* Maximum length of 20 characters.

If the Password is in line with conditions, return True otherwise return False.

1. Generate a random Password based on previous rules.

**Exercise 3:**

## **Deck of Cards**

Make a python module that implements a deck of cards.

Every card has a **type** and **number**:

The **type** is one of 4 colors: red, blue, green, yellow

The **number** should be 1 - 10.

Your *Deck of Cards* holds all of the cards and exposed several methods:

1. *shuffle* - the cards should be randomly rearranged.
2. *deal* - pop a card to the player.
3. *iterator* - The cards will pop one at a time as in the example bellow.

|  |
| --- |
| from deck import Deck() mydeck = Deck()  for card in mydeck:  print(card)  # now the deck is empty |

**Exercise 4:**

## **my\_reduce**

|  |
| --- |
| my\_reduce(function, sequence[, initial]) -> value |

Apply a function of two arguments cumulatively to the items of a sequence, from left to right, so as to reduce the sequence to a single value.

Do not use reduce build function

For example:

|  |
| --- |
| my\_reduce(sum, [1, 2, 3, 4, 5]) calculates ((((1+2)+3)+4)+5). |

**Exercise 5:**

Create Rectangle, Square, Circle and Equilateral triangle classes that receive the required parameters in their i. each should also have two methods:

1. calculate\_area
2. calculate\_perimeter

Design your program structure and implement efficiently using all OOP principles (inherit where you care)