Assignment for the Weekend:
Python Assignment Part 1

1. **Question 1**: General questions:
   a. Using Map with lambda function generates a third list with a single map statement that sums the integer elements of the same index of two given lists.

      lst1=[100, 200, 300, 400, 500]
      lst2=[1,10,100,1000,10000]

      Should return with a single statement : [101, 210, 400, 1400, 10500]

   b. Write a function that takes a string and returns the dictionary with each character as key and its count as value.
      For example:
      result = myfunc ("aaaaabbbbcccdde")

      Should return
      { 'a' : 5, 'b' : 4, 'c' : 3, 'd' : 2, 'e' : 1 }

   c. The dictionary given below consists of vehicles and their weights in kilograms. Construct a list of the names of vehicles with weight below 5000 kilograms. In the same list comprehension makes the key names all uppercase. Use just a single comprehension statement to achieve it.

      dict={
      "Sedan": 1500, "SUV": 2000, "Pickup": 2500, "Minivan": 1600, "Van": 2400, "Semi": 13600, "Bicycle": 7, "Motorcycle": 110
      }

      Solution should be just a statement with list comprehension like below
      List = [use comprehension to achieve the result in single statement]

2. **Question 2** Create a program to create a following form inputs as CLI inputs

Variable	type
Name	String
DOB	Date type in format 'mm/dd/yy'
Age	Integer
Hobbies	List of stings

And write to a file as json data. Give a choice to the user to quit the program or repeat the process.
Also Validate the data type from the user.

3. **Question 3**: A bracket is considered to be any one of the following characters:
   (,), {, }, [, or].

Two brackets are considered to be a *matched pair* if the opening bracket (i.e., **(, [, or {**) occurs to the left of a closing bracket (i.e., **),], or }**) *of the exact same type*. There are three types of matched pairs of brackets: **[], {}, and ()**.

A matching pair of brackets is *not balanced* if the set of brackets it encloses are not matched. For example, **{[(])}** is not balanced because the contents in between **{** and **}** are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, **(**, and the pair of parentheses encloses a single, unbalanced closing square bracket, **]**.

By this logic, we say a sequence of brackets is *balanced* if the following conditions are met:

- It contains no unmatched brackets.
- The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets.

Given a string of brackets as input, determine whether each sequence of brackets is balanced. If a string is balanced, return YES. Otherwise, return NO.

Program Description:

Complete program with the function ***isBalanced*** .

isBalanced has the following parameter(s):

- *string s*: a string of brackets [take it as an input argument from the user from CLI]

Returns

- *string:* either YES or NO

Example Input: "{}[]()[(())]" returns YES

" {}(]{" return NO

4. **Question 4 :**A *left rotation* operation on an integer array shifts each of the array's elements unit to the left. For example, if **2** left rotations are performed on array **[1,2,3,4,5]** , then the array would become **[3,4,5,1,2]**

   Note that the lowest index item moves to the highest index in a rotation. This is called a *circular array*.

Given an array of integers and a number designating the number of rotations, , perform left rotations on the array. Return the updated array to be printed as a single line of space-separated integers.

Program Description

Write a Program i.e. a python script with the function named ***rotateLeft*** in the

rotateLeft has the following parameter(s):

- ***array_list*** a list of integers to rotate
- ***num_rotate*** the number of rotations to be made

Receive these parameters from the command line as input from the user.

Returns

- the rotated array and the number of left rotations.

Python Assignment Part 2

1. Create generator with and without comprehension for getting multiples of given number upto 10.
    a. Eg. generator(5) =>> 5, 10, 15 …. 50

2. Create a scenario where following errors are handled:
    a. Custom Error implemented using class
    b. Custom Error using Exception or BaseException class using message to handle at least two of the cases.
    c. Full fledged case for exception handling using try, except, else, finally

3. Create at least 4 classes having semantic meaning (having relation to each other) so that multiple inheritance can be achieved and incorporating following things in some of them:
    a. @classmethod
    b. @staticmethod
    c. @property and setter for it
    d. Class_variable
4. Create a class for complex number implementing all the arithmetic operations and relational operations related dunder methods. Eg. __add__, __iadd__, __mult__, __eq__, __lt__, …
5. Create a Class representing the clone behavior of lists in python.
    a. Eg. List(“1234”) works same as list(“1234”)
    b. Adding 2 lists, multiple ways of instantiating using @classmethod.
6. Replicate the behaviour of range() object using :
    a. Iterator class, __iter__ and __next__
    b. Generator
    c. Generator comprehension

Python Assignment Part 3

- Create a connection to a postgres database using psycopg2
- Create a table named 'users'
    - -- columns -> id, name, dob, profession
- Create table names `address`
    - -- Columns -> id, user_id (FK -> users), permanent_address, temporary_address
- Insert dummy data in the tables using psycopg connection
- Fetch data from the joined users and address table
    - -- given user_id
    - -- given profession and permanent_address
- Update table users and add column gender
- Delete records from user whose age is less than 20 yrs

-- Note: Maintain the sql queries in `sql` folder