PROGRAMMING FOR ALL – SPRING 2021 - WEEK #9 – LAB – 210526

GENERAL LAB RULES:

1. **Every student has to join lab sessions.**
2. A goal of lab assignments is to practice topics covered in class during each week. Each lab has two parts: Theory & Programming.
3. In the theory part, you are expected to read pieces of codes and to predict outputs of these codes. You should not use Python for these predictions; you should test your skills to understand codes. When everyone is ready, TA displays correct outputs.
4. In the programming part, you are expected to make programs for assigned problems. You are expected to make all programs during the lab time, however, only one specified program should be submitted to CANVAS.
5. TA has to check your lab work for the full lab credit.

TOPICS:

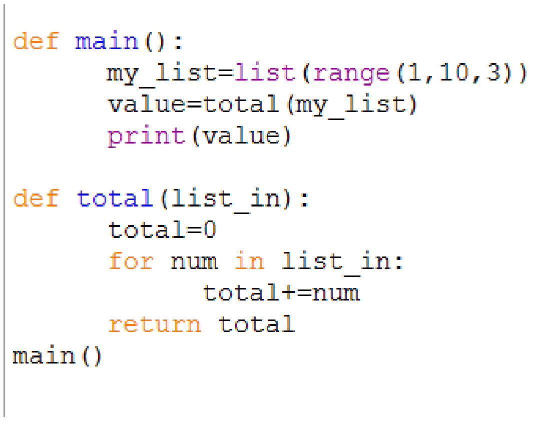
LISTS methods

PART I. – THEORY

PROGRAM OUTPUTS

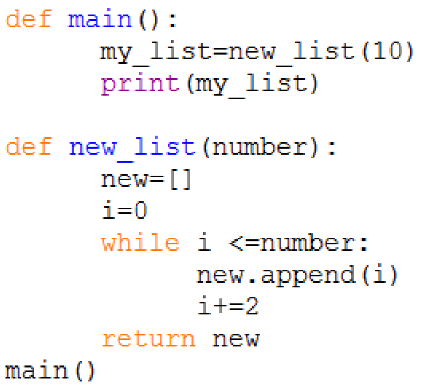
First predict and then verify the outputs of the following print statements. Don't use Python, open an empty file or use a paper to predict. TA will share correct answers with you. The goal of these exercises is to understand programming codes.

#1



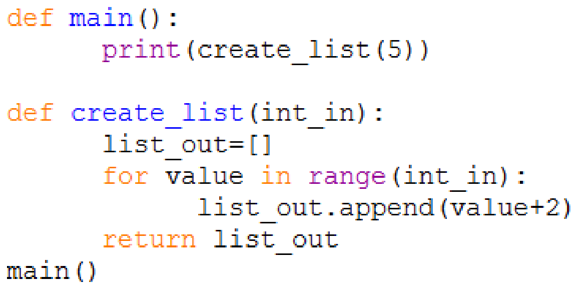
```
def main():
        my_list=list(range(1,10,3))
        value=total(my_list)
        print(value)

def total(list_in):
        total=0
        for num in list_in:
                total+=num
        return total
main()
```

#2



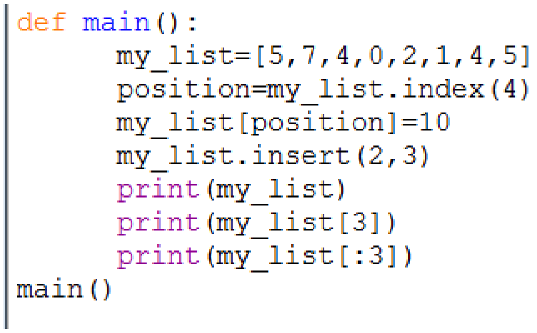
```
def main():
        my_list=new_list(10)
        print(my_list)

def new_list(number):
        new=[]
        i=0
        while i <=number:
                new.append(i)
                i+=2
        return new
main()
```

#3



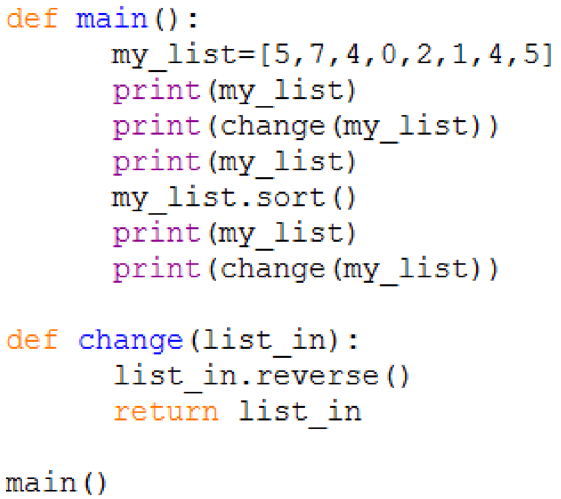
```
def main():
        print(create_list(5))

def create_list(int_in):
        list_out=[]
        for value in range(int_in):
                list_out.append(value+2)
        return list_out
main()
```

#4



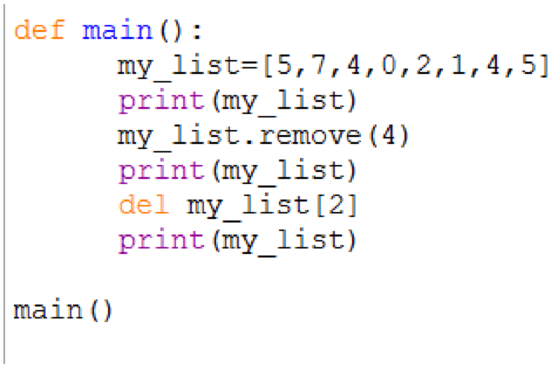
```
def main():
        my_list=[5,7,4,0,2,1,4,5]
        position=my_list.index(4)
        my_list[position]=10
        my_list.insert(2,3)
        print(my_list)
        print(my_list[3])
        print(my_list[:3])
main()
```

#5



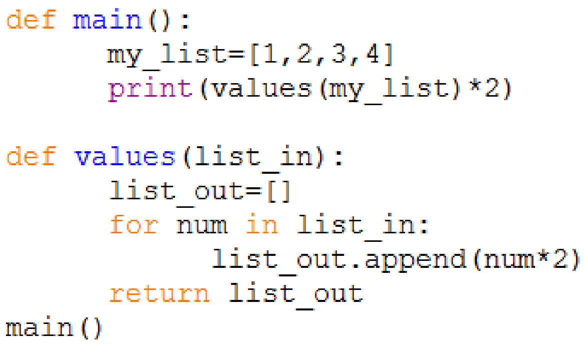
```
def main():
        my_list=[5,7,4,0,2,1,4,5]
        print(my_list)
        print(change(my_list))
        print(my_list)
        my_list.sort()
        print(my_list)
        print(change(my_list))

def change(list_in):
        list_in.reverse()
        return list_in

main()
```

#6



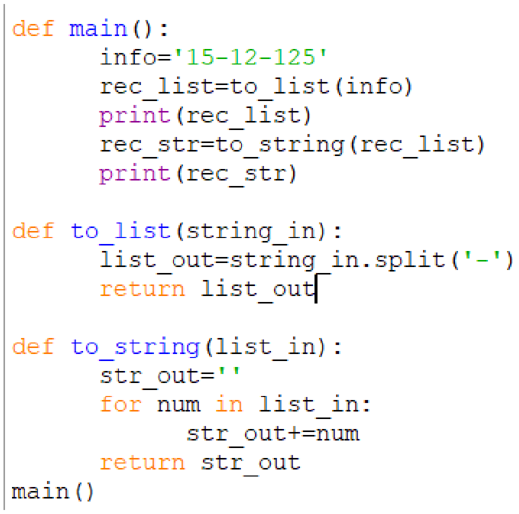
```
def main():
        my_list=[5,7,4,0,2,1,4,5]
        print(my_list)
        my_list.remove(4)
        print(my_list)
        del my_list[2]
        print(my_list)

main()
```

#7



```
def main():
        my_list=[1,2,3,4]
        print(values(my_list)*2)

def values(list_in):
        list_out=[]
        for num in list_in:
                list_out.append(num*2)
        return list_out
main()
```

#8



```
def main():
        info='15-12-125'
        rec_list=to_list(info)
        print(rec_list)
        rec_str=to_string(rec_list)
        print(rec_str)

def to_list(string_in):
        list_out=string_in.split('-')
        return list_out

def to_string(list_in):
        str_out=''
        for num in list_in:
                str_out+=num
        return str_out
main()
```

PART II. PROGRAMMING

Note: In all programs you can use only methods and tools already introduced in the class.

PROGRAM #1: TWO LISTS SUMS – must be submitted to CANVAS as lab9.py.

Start the program with: # your name.

STEP 1:

Write a value returning function two_lists with a parameter n representing the number of elements in a list, which returns two lists of random integers from 1, 50 with n elements.

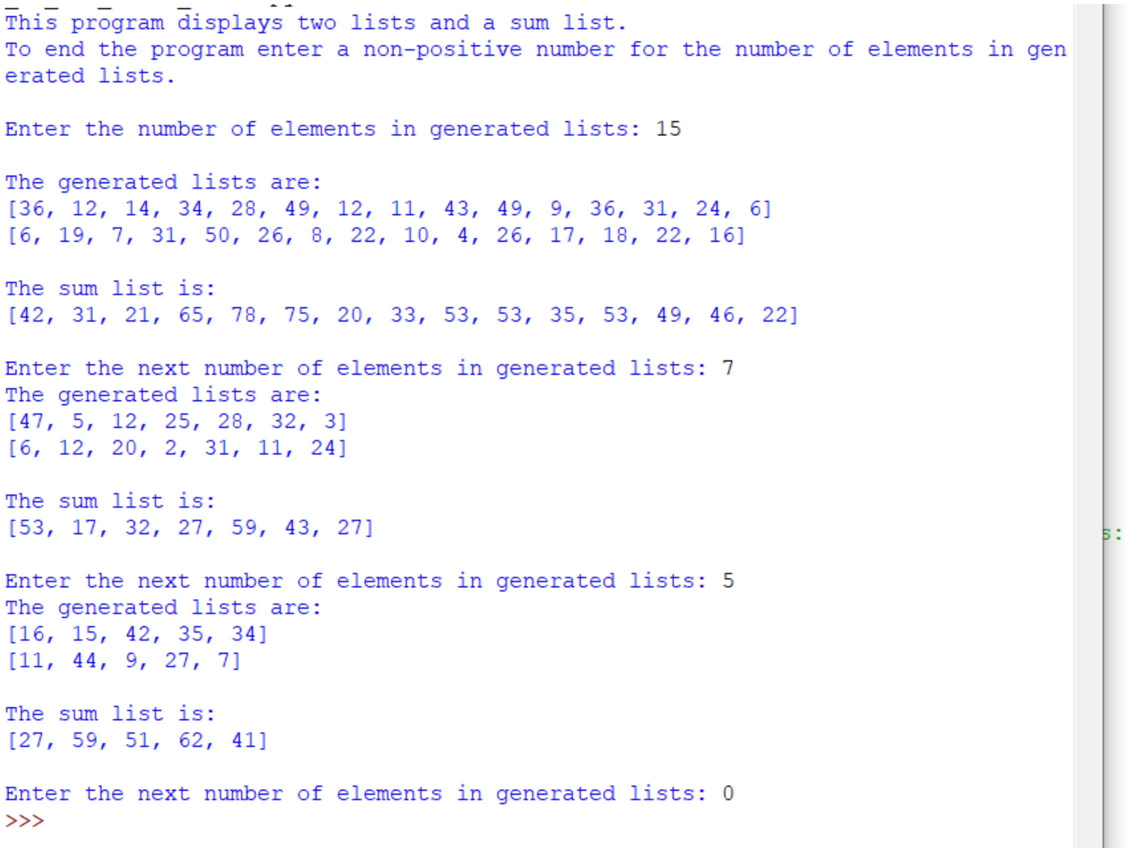
Write a value returning function sum_list with two parameters representing two lists, which returns a list, where each element is a sum of values of the parameters lists in with the same index. Example: if the parameters lists be [1, 5, 7] and [2, 6, 3], the function returns [3, 11, 10].

STEP 2:

Use the functions from the STEP 1 in the main function to create a program where a user provides value n and the program displays two generated lists of random numbers and the sum list.

The user can provide a multiple number of values n, by entering 0 or a negative value for n the program ends.

Example of the output:



```
This program displays two lists and a sum list.
To end the program enter a non-positive number for the number of elements in gen
erated lists.

Enter the number of elements in generated lists: 15

The generated lists are:
[36, 12, 14, 34, 28, 49, 12, 11, 43, 49, 9, 36, 31, 24, 6]
[6, 19, 7, 31, 50, 26, 8, 22, 10, 4, 26, 17, 18, 22, 16]

The sum list is:
[42, 31, 21, 65, 78, 75, 20, 33, 53, 53, 35, 53, 49, 46, 22]

Enter the next number of elements in generated lists: 7
The generated lists are:
[47, 5, 12, 25, 28, 32, 3]
[6, 12, 20, 2, 31, 11, 24]

The sum list is:
[53, 17, 32, 27, 59, 43, 27]

Enter the next number of elements in generated lists: 5
The generated lists are:
[16, 15, 42, 35, 34]
[11, 44, 9, 27, 7]

The sum list is:
[27, 59, 51, 62, 41]

Enter the next number of elements in generated lists: 0
>>>
```

PROGRAM #2: DIE SUM GAME

Write a program that simulates a die game.

STEP 1:

Write a value returning function `computer_roll` with a parameter (n) representing the number of rolls, which:

- Creates and returns a list of n random integers from 1 to 6;
- Returns the sum of generated numbers in the list.

STEP 2:

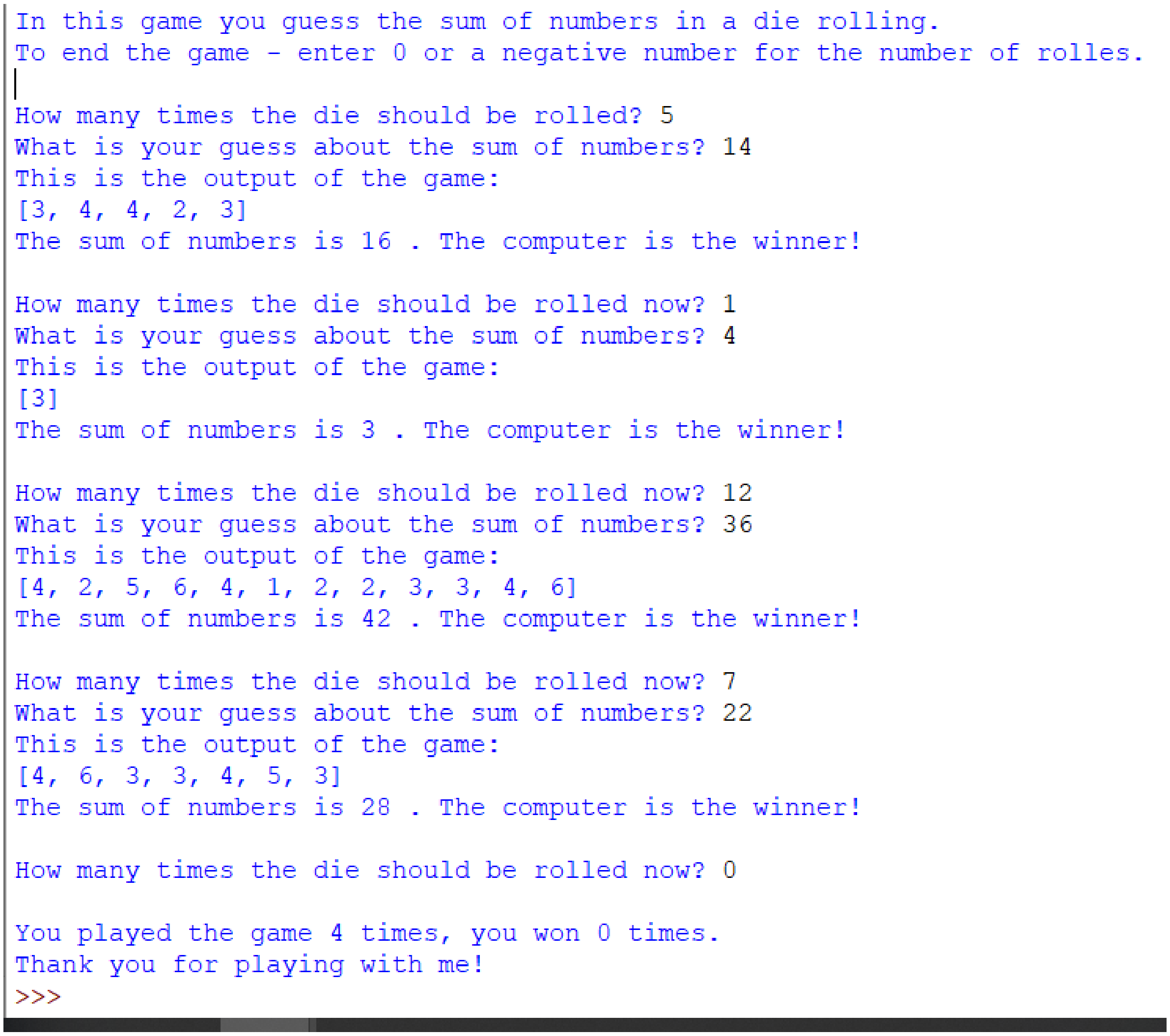
Write a program which simulates a die game between a user and the computer. The user enters a number of rolls of a die, and his/her guess about the sum of the rolled numbers.

When the user enters these values, the program calls the function `computer_roll` to display the output of the game. If the user's guess about the sum is correct, he/she is a winner, otherwise the winner is a computer.

The user can play the game as long as he/she wishes. If the user enters 0 or a negative integer for the number of times to roll the die (n), the game ends.

At the end, the program displays the number of games played and the number of wins of the user.

Example of the output:



```
In this game you guess the sum of numbers in a die rolling.
To end the game - enter 0 or a negative number for the number of rolles.

How many times the die should be rolled? 5
What is your guess about the sum of numbers? 14
This is the output of the game:
[3, 4, 4, 2, 3]
The sum of numbers is 16 . The computer is the winner!

How many times the die should be rolled now? 1
What is your guess about the sum of numbers? 4
This is the output of the game:
[3]
The sum of numbers is 3 . The computer is the winner!

How many times the die should be rolled now? 12
What is your guess about the sum of numbers? 36
This is the output of the game:
[4, 2, 5, 6, 4, 1, 2, 2, 3, 3, 4, 6]
The sum of numbers is 42 . The computer is the winner!

How many times the die should be rolled now? 7
What is your guess about the sum of numbers? 22
This is the output of the game:
[4, 6, 3, 3, 4, 5, 3]
The sum of numbers is 28 . The computer is the winner!

How many times the die should be rolled now? 0

You played the game 4 times, you won 0 times.
Thank you for playing with me!
>>>
```