PROGRAMMING IN PYTHON: EXERCISES MAESTRÍA EN ANÁLISIS ESTADÍSTICO Y COMPUTACIÓN (CIMAT-INEGI)

OSCAR DALMAU

Contents

1. Exercises: Class 1

1.1. String.

(1) Write a Python program to calculate the length of a string.

(2) Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.
Sample String : 'w3resource'
Expected Result : 'w3ce'
Sample String : 'w3'
Expected Result : 'w3w3'
Sample String : 'w'
Expected Result : Empty String

(3) Write a Python function that takes a list of words and returns the length of the longest one.

Date: August 21, 2020.

1.2. Lists.

(1) Write a Python program to remove duplicates from a list.

(2) Write a Python program to check a list is empty or not.

(3) Write a Python program to find the list of words that are longer than n from a given list of words.

1.3. Dictionaries.

(1) Write a Python script to concatenate following dictionaries to create a new one.

```
Sample Dictionary :
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={:50,6:60}

Expected Result :{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

(2) Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form key:value => x:x*x, ie, the values are square of keys

Sample Dictionary (n = 5) :

Expected Output : $\{1:1, 2:4, 3:9, 4:16, 5:25\}$

1.4. Tuples.

(1) Write a Python program to get the 4th element from the begining and 4th element from last of a tuple.

(2) Write a Python program to add an item in a tuple.

(3) Write a Python program to transform a tuple of pair of tuples into a dictionary

Example:

```
#Create a tuple
tuplex = ((2, "w"),(3, "r"))
# Expected Result
dictionary = {"w":2, "r":3}
```

2. Exercises: Class 2

2.1. Loops.

(1) Write a Python program to construct the following pattern, using a nested for loop.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

(2) Write a Python program to print alphabet pattern ‘E’. Expected Output:

```
*****
*
*
****
*
*
*****
```

See example below:

Example: Write a Python program to print alphabet pattern ‘G’.

Expected Output:

```
 ***
*   *
*
* ***
*   *
*   *
 ***
```

```
# python script
result=""
for row in range(0,7):
    for column in range(0,7):
        if ((column == 1 and row != 0 and row != 6)
         or ((row == 0 or row == 6) and column > 1 and column < 5)
         or (row == 3 and column > 2 and column < 6)
         or (column == 5 and row != 0 and row != 2 and row != 6)):
            result=result+"*"
        else:
            result=result+" "
    result=result+"\n"
print(result)
```

2.2. **Functions.**

(1) Write a Python function to sum all the numbers in a list.
(2) Write a Python function to compute the mean and std of all the numbers in a list.
(3) Write a Python program to print the even numbers from a given list.
(4) Write a Python function that checks whether a passed string is palindrome or not.
Note: A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., madam or nurses run.
(5) Write a Python function to check whether a number is perfect or not.

According to Wikipedia : In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum). Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself). Example : The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and $1 + 2 + 3 = 6$. Equivalently, the number 6 is equal to half the sum of all its positive divisors: $(1 + 2 + 3 + 6) / 2 = 6$. The next perfect number is $28 = 1 + 2 + 4 + 7 + 14$. This is followed by the perfect numbers 496 and 8128.

(6) Write a Python function that prints out the first n rows of Pascal's triangle. Note : Pascal's triangle is an arithmetic and geometric figure first imagined by Blaise Pascal.

Sample Pascal's triangle : $n = 5$ rows
$[1]$
$[1, 1]$
$[1, 2, 1]$
$[1, 3, 3, 1]$
$[1, 4, 6, 4, 1]$

Note: The extremes are 1 and the remainder is the two numbers above it added together

2.3. **Classes.**

(1) Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.

(2) Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

(3) Write a Python class named Complex constructed by the real and imaginary parts of a complex number.

```
# Python Program to perform addition, subtraction and
# multiplication of two complex numbers using the
# following binary  operators
# + operator overloading.
# - operator overloading.
# * operator overloading.

class Complex:
    def __init__(self, a, b):
        self.a = a
        self.b = b

     # adding two objects
    def __add__(self, other):
        return Complex(self.a + other.a, self.b + other.b)



    # subtracting two objects
    def __sub__(self, other):
        # add your code


   # multiplying two objects
    #def __mul__(self, other):
     # add your code

    # str is called when using: str(c), print(c), ‘{}’.format(c)
    def __str__(self):
        if self.b < 0 and self.b == -1:
            num = str(self.a) + ‘ - i’
        elif self.b < 0:
            num = str(self.a) + ‘ - ’ + str(-self.b) + ‘i’
        elif  self.b == 0:
            num = str(self.a)
        elif self.b > 0 and self.b == 1:
            num = str(self.a) + ‘ + i’
        else:
            num = str(self.a) + ‘ + ’ + str(self.b) + ‘i’
        return num
```

```
# Example
c1 = Complex(1,-2)
c2 = Complex(2, 3)
c3 = c1 + c2
c4 = c1 - c2
c5 = c1 * c3
print(c3)
print(c4)
print(c5)
```