Codificación y Programación.

Quiz Capítulos #3 Y #4.

Q. 03-01. Diseñe un programa que reciba las coordenadas (x1, y1), (x2, y2) de dos puntos del usuario e imprima la distancia entre los dos puntos. Para hacer esto, implemente la función distancia (x1, y1, x2, y2). Pauta de código: consulte la ecuación para encontrar la distancia entre dos puntos.

```
In [1]: import math

x1 = float (input ('Ingresa el valor de x1: '))
x2 = float (input ('Ingresa el valor de x2: '))
y1 = float (input ('Ingresa el valor de y1: '))
y2 = float (input ('Ingresa el valor de y2: '))
distancia=math.sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))
print ('Valor de distancia: ' + repr (distancia))
print ()
```

```
Ingresa el valor de x1: 2
Ingresa el valor de x2: 10
Ingresa el valor de y1: 5
Ingresa el valor de y2: 5
Valor de distancia: 8.0
```

Q. 03-02. Un palíndromo es una oración, palabra o cadena que se lee igual al derecho o al revés. Por ejemplo, reconocer, radar o la frase “anita lava la tina”. Usemos una llamada recursiva para determinar el palíndromo. Defina una función llamada is_palindrome y escriba un programa que reciba una cadena del usuario e imprima si el palíndromo es correcto o no. Pauta de código: Llame a la función is_palindrome dentro de la función is_palindrome(función recursiva).

```
In [2]: def isPalindrome(s):
            return s == s[::-1]

        s = input("Digite una palabra: ")
        ans = isPalindrome(s)

        if ans:
            print("Yes")
        else:
            print("No")
```

```
Digite una palabra: malayalam
Yes
```

Q. 03-03. Defina una lista llamada n_list con valores de [10, 20, 30]. Utilice la función map y lambda para imprimir los resultados de duplicar, triplicar y cuadruplicar los objetos de la lista

ejemplo:mapped_numbers = list(map(lambda x: x * 2 + 3, numbers))

```
In [18]: n_list = [10, 20, 30]

         list( map(lambda x: x*2, c) )

Out[18]: [20, 40, 60]

In [19]: n_list = [10, 20, 30]

         list( map(lambda x: x*3, c) )

Out[19]: [30, 60, 90]

In [20]: n_list = [10, 20, 30]

         list( map(lambda x: x*4, c) )

Out[20]: [40, 80, 120]
```

Q. 03-04. Explique el funcionamiento del siguiente código, para ello aplique una prueba de escritorio hecha "a mano"

```
1  def calc_digit(n):
2      def final(digit):
3          return digit**n
4      return final
5
6  num_list=[]
7  for num in range(1,6):
8      num_list.append(calc_digit(num))
9      print(num_list[num - 1](num))
```

```
def calc_digit(n):
    def final(digit):
        return digit**n
    return final
num_list=[]
for num in range(1,6):
    num_list.append(calc_digit(num))
    print(num_list[num - 1](num))
```

```
1
4
27
256
3125
```

En la sucesión se observa que el primer número se obtiene de elevar la base (1) con exponente igual a su base (1) de esta manera: $1^1 = 1$ para luego al segundo número se obtiene al elevar la base del número que sigue (2) con exponente igual a su base (2): $2^2 = 4$ y así sucesivamente dentro de un for en un rango del 1 al 6.

Q. 03-05. Implemente las funciones de multiplicación (*) y división (/) de dos vectores usando los métodos especiales __mul__ y __truediv__. Suponiendo que v1 es (30, 40) y v2 es (10, 20), codifique para devolver el siguiente resultado como resultado de la multiplicación y división de dos vectores.

```
v1 * v2 =  (300,800)
v1 / v2 =  (3.0,2.0)
```

Pauta de codificación: escriba un código implementando clases que reciba 2 objetos de la clase vector2D y los opere como se ha solicitado

Hint:

```
v1 = Vector2D(30, 40)
v2 = Vector2D(10, 20)
v3 = v1 * v2
v4 = v1 / v2
print('v1 * v2 = ',v3)
print('v1 / v2 = ',v4)
```

In [64]:
```
class Vector2D:
    def __init__(self, value, value2):
        self.value = value
        self.value2 = value2

    def __mul__(self, other):
        return Vector2D(self.value * other.value, self.value2 * other.value2)
    def __truediv__(self, other):
        return Vector2D(self.value / other.value, self.value2 / other.value2)

v1 = Vector2D(30,40)
v2 = Vector2D(10,20)
v3 = v1 * v2
v4 = v1 / v2
print(v3.value,v3.value2)
print(v4.value, v4.value2)
```

```
300 800
3.0 2.0
```

Q. 04-01. La siguiente es la implementación de una pila en python. ¿Cuál será el resultado del siguiente código?

```
stack = Stack()
stack.push("Banana")
stack.push("Apple")
stack.push("Tomato")
stack.pop()
stack.push("Strawberry")
stack.push("Grapes")
stack.pop()
print(stack.stack)
```

Pauta de codificación: escriba cual es el resultado esperado de ejecutar cada línea (prueba de escritorio)

```
['Strawberry', 'Apple', 'Banana']
```

Q. 04-02. La siguiente es la implementación de una pila en python. ¿Cuál será el resultado del siguiente código?

```
stack = Stack()
items = [10 * i for i in range(1, 10)]
for item in items:
    stack.push(item)
    if (item // 10) % 2 == 0:
        stack.pop()
print(stack.stack)
```

Pauta de codificación: escriba cual es el resultado esperado de ejecutar cada línea (prueba de escritorio)

```
[90, 70, 50, 30, 10]
```

Q. 04-03. A continuación se muestra la implementación de una cola en python(queue). ¿Cuál será el resultado del siguiente código?

```
queue = Queue()
items = [10 * i for i in range(1, 11)]
for item in items:
    queue.enqueue(item)
    if (item // 10) % 2 == 0:
        queue.dequeue()
print(queue.queue)
```

Pauta de codificación: escriba cual es el resultado esperado de ejecutar cada línea (prueba de escritorio)

```
Out[78]: 5
```

Q. 04-04. ¿Cuál es el algoritmo de la siguiente función find_two()? Analice el código y escriba el resultado de la ejecución.

```
1 def find_two(nums):
2     x = y = 0
3     for i in range(1, len(nums)):
4         if nums[x] < nums[i]:
5             x = i
6         elif nums[y] > nums[i]:
7             y = i
8     return x, y
```

```
1 nums = [11, 37, 45, 26, 59, 28, 17, 53]
2 i, j = find_two(nums)
3 print(nums[i], nums[j])
```

```
59 11
```

Buscó el número mayor y el número menor de los que estaban presentes.

Q. 04-05. ¿Cuántas comparaciones debe realizar la función find_two() implementada en la pregunta anterior (Q.04-04)?

In [101]:

```
def find_two(nums):
    x = y = a = b = c = 0
    for i in range(1, len(nums)):
        a = a + 1
        if nums[x] < nums[i]:
            x = i
            b = b + 1
        elif nums[y] > nums[i]:
            y = i
            c = c + 1
    return x,y,a,b,c

nums = [11,37,45,26,59,28,17,53]

i,j, a,b,c = find_two(nums)
print(nums[i],nums[j],nums[a])
print(a,b,c)
```

```
59 11 53
7 3 0
```

Según este código, este código, el for realiza 7 repeticiones de las cuales 3 dieron ciertas la primera condicional y 0 en la segunda.

Q. 04-06. El siguiente es el código para el juego de combinación de números. Si el máximo es 100 y el número es 51, ¿cuál es la salida de count?

```
from random import randint

maximum = int(input("Enter the number of maximum: "))
number = int(input("Enter your guessing number: "))
count = 0
low, high = 1, maximum
while low < high:
    mid = (low + high) // 2
    count += 1
    if mid == number:
        print(f"Your number is {number}.")
        break
    elif mid > number:
        high = mid - 1
    else:
        low = mid + 1
print(f"Total {count} times are searched.")
```

```
Enter the number of maximum: 100
Enter your guessing number: 51
Your number is 51.
Total 6 times are searched.
```

Q. 04-07. En el código del juego de combinación de números, si el máximo es 100 y el número es 25, ¿cuál es el resultado del conteo?

```
Enter the number of maximum: 100
Enter your guessing number: 25
Your number is 25.
Total 2 times are searched.
```

Q. 04-08. Usando la función insert de la clase hash_table, ingrese la clave "Alicia en el país de las maravillas", a continuación obtenga la clave hash de buscar dicha clave dentro de la hash_table

Pauta de codificación: Ayúdese de la definición de la clase hash_table del siguiente ejemplo: https://pythondiario.com/2018/06/tabla-hash-en-python.html

In [113]:
```
class hash_table:
    def __init__(self):
        self.table = [None] * 127

    # Función hash
    def Hash_func(self, value):
        key = 0
        for i in range(0,len(value)):
            key += ord(value[i])
        return key % 127

    def Insert(self, value):
        hash = self.Hash_func(value)
        if self.table[hash] is None:
            self.table[hash] = value

    def Search(self,value):
        hash = self.Hash_func(value);
        if self.table[hash] is None:
            return None
        else:
            return hex(id(self.table[hash]))

    def Remove(self,value):
        hash = self.Hash_func(value);
        if self.table[hash] is None:
            print("No hay elementos con ese valor", value)
        else:
            print("Elemento con valor", value, "eliminado")
            self.table[hash] is None;


H = hash_table()
wo = "Alicia en el país de las maravillas"
H.Insert(wo)

print(H.Search(wo))
```

```
0x1f83ec9eab0
```

Q. 04-9. Si la nueva estantería tiene 10 compartimentos, usa el siguiente código para averiguar qué libro hay en cada compartimento.

```
table = HashTable(10)
books = [
    "The Little Prince",
    "The Old Man and the Sea",
    "The Little Mermaid",
    "Beauty and the Beast",
    "The Last Leaf",
    "Alice in WonderLand"
]
for book in books:
    key = sum(map(ord, book))
    table.put(key, book)
for key in table.table.keys():
    print(key, table.table[key])
```

```
Books#  1  : The Old Man and the Sea

Books#  2  : The Little Mermaid

Books#  3  : Beauty and the Beast

Books#  4  : The Last Leaf

Books#  5  : Alice in WonderlandEmpty

Books#  6  : Empty

Books#  7  : Empty

Books#  8  : Empty

Books#  9  : Empty
```