

Laboratorios de computación

salas A y B

<i>Profesor:</i>	<i>Marco Antonio Martinez Quintana</i>
<i>Asignatura:</i>	<i>Estructuras de datos y algoritmos I</i>
<i>Grupo:</i>	<i>17</i>
<i>No de Práctica(s):</i>	<i>10</i>
<i>Integrante(s):</i>	<i>González Cuellar Arturo</i>
<i>No. de equipo de cómputo empleado:</i>	
<i>No. de Lista o Brigada</i>	<i>16</i>
<i>Semestre:</i>	<i>2020-2</i>
<i>Fecha de entrega:</i>	<i>17 - Abril - 2020</i>
<i>Observaciones:</i>	

Calificación: _____

Introducción a Python (II).

Objetivo:

Aplicar las bases del lenguaje de programación Python en el ambiente de Jupyter notebook.

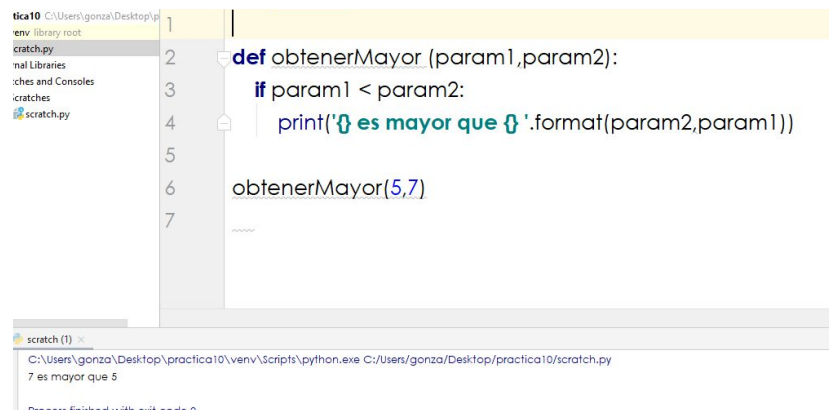
Introducción:

Se aplicaran las bases del lenguaje de programación Python para introducirnos en este lenguaje conociendo cómo interactuar en Jupyter notebook, aplicar estructuras de control selectivas y repetitivas, usar las bibliotecas estándar, generar una gráfica, ejecutar un programa desde el terminal, así como la entrada de datos, todo esto conociendo la sintaxis de este lenguaje.

Desarrollo y resultados:

if : Nos sirve para ejecutar código dependiendo de la condición que asignemos.

En este caso se puso un if en el cual se le asignaron dos números, si no se cumpliera esa condición el programa no hubiera imprimido nada.



```
1
2 def obtenerMayor (param1,param2):
3     if param1 < param2:
4         print('{} es mayor que {}'.format(param2,param1))
5
6 obtenerMayor(5,7)
7
```

scratch (1) x

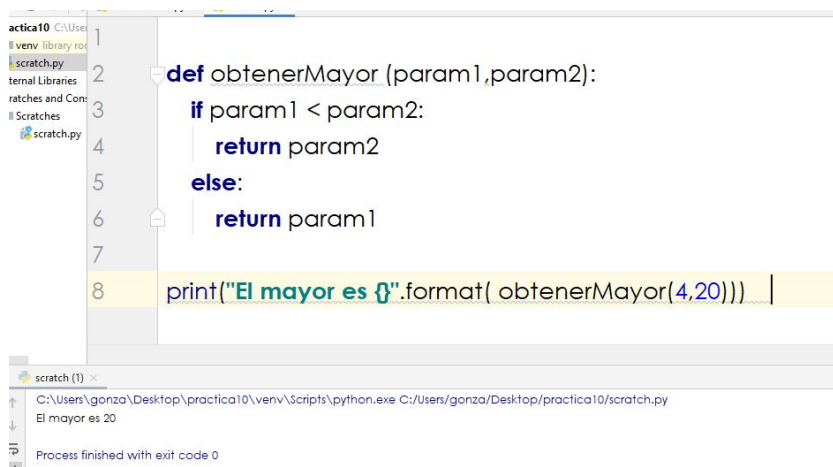
C:\Users\gonza\Desktop\practica10\venv\Scripts\python.exe C:/Users/gonza/Desktop/practica10/scratch.py

7 es mayor que 5

Process finished with exit code 0

if-else : Esta función nos sirve para que tengamos otra opción en caso de que la primera condición no se cumpla.

En este ejemplo se dan dos número en los cuales si no se cumple la primera condición del if entra al else.



```
1
2 def obtenerMayor (param1,param2):
3     if param1 < param2:
4         return param2
5     else:
6         return param1
7
8 print("El mayor es {}".format( obtenerMayor(4,20)))
```

scratch (1) x

C:\Users\gonza\Desktop\practica10\venv\Scripts\python.exe C:/Users/gonza/Desktop/practica10/scratch.py

El mayor es 20

Process finished with exit code 0

En este ejemplo la variable valor va tener el valor de param2 si el **if** es verdadero, si no tendrá el valor de param1

```

1
2 def obtenerMayor_idiom(param1,param2):
3     valor = param2 if (param1 < param2) else param1
4     return valor
5
6 print("El mayor es {}".format(obtenerMayor_idiom(11,6)))

```

scratch (1) x

C:\Users\gonza\Desktop\practica10\venv\Scripts\python.exe C:/Users/gonza/Desktop/practica10/scratch.py

El mayor es 11

Process finished with exit code 0

```

1
2 def numeros(num):
3     if num == 1:
4         print("Tu numero es 1")
5     elif num == 2:
6         print("El numero es 2")
7     elif num == 3:
8         print("El numero es 3")
9     elif num == 4:
10        print("El numero es 4")
11    else:
12        print("No hay opcion")
13
14 numeros(2)

```

scratch (1) x

C:\Users\gonza\Desktop\practica10\venv\Scripts\python.exe C:/Users/gonza/Desktop/practica10/scratch.py

El numero es 2

Process finished with exit code 0

if-elif-else: Esta nos sirve para generar varios casos, en esta caso el el Switch que conocemos en el lenguaje C.

En este ejemplo se hace un menú con opciones dependiendo el número que le asignes.

En este ejemplo se simplifica el uso del elif poniendo una lista dentro de la condición del if asignada a una variable. De esta manera se evita el uso de muchos elif y tiene el mismo resultado.

```

2 def numeros_idiom(num):
3     if num in (1,2,3,4):
4         print("Tu numero es {}".format(num))
5     else:
6         print("{} no es una opcion".format(num))
7
8 numeros_idiom(2)
9 numeros_idiom(5)

```

scratch (1) x

C:\Users\gonza\Desktop\practica10\venv\Scripts\python.exe C:/Users/gonza/Desktop/practica10/scratch.py

Tu numero es 2
5 no es una opcion

Process finished with exit code 0

```

2 def obtenerMasGrande(a,b,c):
3     if a > b:
4         if a > c:
5             return a
6         else:
7             return c
8     else:
9         if b > c:
10            return b
11        else:
12            return c
13
14 print("El mas grande es {}".format(obtenerMasGrande(7,13,1)))

```

scratch (1) x

C:\Users\gonza\Desktop\practica10\venv\Scripts\python.exe C:/Users/gonza/Desktop/practica10/scratch.py

El mas grande es 13

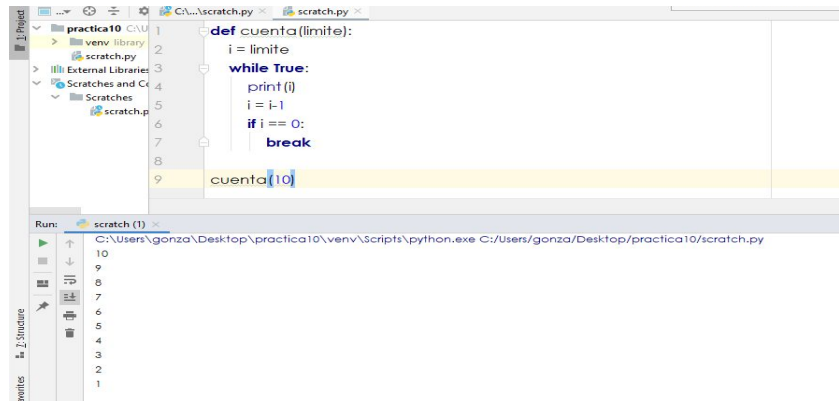
Process finished with exit code 0

En este caso se realizó una estructura selectiva anidada, en la cual se van recorriendo los if hasta que se cumpla alguna condición de las que se presentan.

Estructuras de control repetitivas

Ciclo while: Este ciclo se puede repetir tantas veces como se cumpla la condición, para que se siga ejecutando siempre debe de ser verdadera. La diferencia con la estructura if es que esta se ejecuta solo una vez.

En este ejemplo se realiza un ciclo while para hacer una cuenta regresiva a cualquier número, mientras que el límite no sea 0, el ciclo se seguirá repitiendo.



```
def cuenta(limite):
    i = limite
    while True:
        print(i)
        i = i - 1
        if i == 0:
            break
    cuenta(10)
```

The screenshot shows a Python script with a function `cuenta(limite)` that uses a `while True` loop to print numbers from `limite` down to 0. The function is called with `cuenta(10)`. The output window shows the numbers 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.



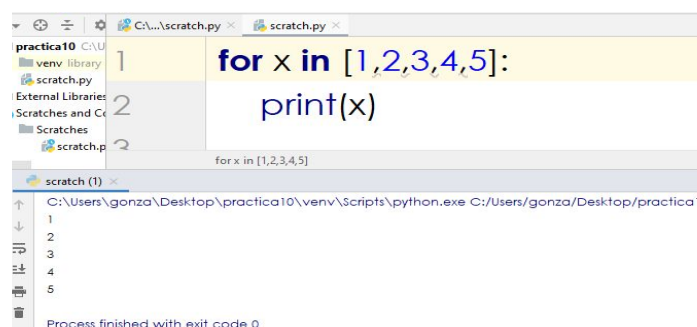
```
def factorial(n):
    i = 2
    tmp = 1
    while i <= n:
        tmp = tmp * i
        i = i + 1
    return tmp

print(factorial(4))
print(factorial(6))
```

The screenshot shows a Python script with a function `factorial(n)` that uses a `while` loop to calculate the factorial of `n`. The function is called with `print(factorial(4))` and `print(factorial(6))`. The output window shows the results 24 and 720.

En esta caso se realiza un programa que calcula el factorial de un número, este programa se apoya en un ciclo while, mientras aumenta el valor de la variable i, lo que hace que se vaya multiplicando el número para obtener su factorial.

Ciclo for: Este ciclo se utiliza principalmente para hacer iteraciones en una lista o diccionarios.



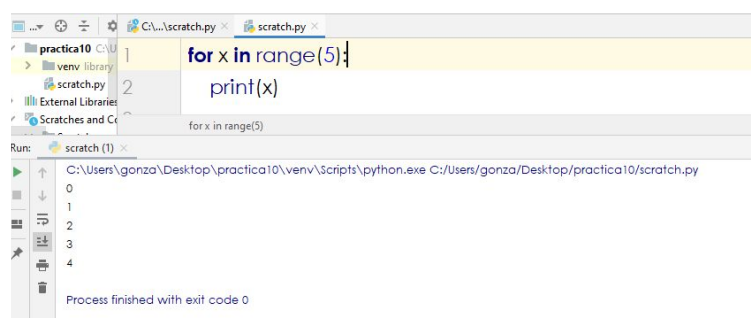
```
for x in [1,2,3,4,5]:
    print(x)
```

The screenshot shows a Python script with a `for` loop that iterates over the list `[1,2,3,4,5]` and prints each element. The output window shows the numbers 1, 2, 3, 4, 5.

Iteración en listas:

En este caso se asigna una lista a una variable, para que de esta manera se recorra la lista y así se imprima el valor que va tomando la variable x.

En este caso se realiza lo mismo que el programa pasado, solo que en este caso se hace uso de la función `range()`, a la cual se le da el rango que se irá recorriendo.

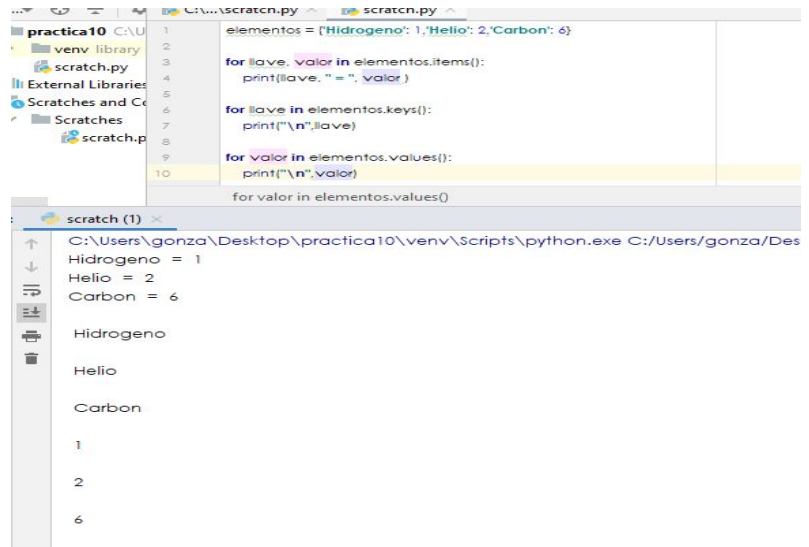


```
for x in range(5):
    print(x)
```

The screenshot shows a Python script with a `for` loop that iterates over the range `range(5)` and prints each element. The output window shows the numbers 0, 1, 2, 3, 4.

Iteración en diccionarios:

En este ejemplo se realiza un ejemplo en el cual se define un diccionario con elementos químicos y su número correspondiente, y con ayuda de ciclos for se imprimen los elementos de estos, dando como parámetro lo que se quiere imprimir, ya sea el diccionario completo, solo el nombre de los elementos o solo el número del elemento.



```
1 elementos = {'Hidrogeno': 1, 'Helio': 2, 'Carbon': 6}
2
3 for llave, valor in elementos.items():
4     print(llave, " = ", valor)
5
6 for llave in elementos.keys():
7     print("\n", llave)
8
9 for valor in elementos.values():
10    print("\n", valor)
11
12 for valor in elementos.values():
```



```
1 def cuenta_idiom (limite):
2     for i in range(limite, 0, -1):
3         print(i)
4     else:
5         print("Cuenta finalizada")
6
7 cuenta_idiom(5)
```

En este ejemplo se asigna un rango a una variable, con este se va recorriendo el parámetro y se imprime el valor que va tomando la variable, cuando esta condición termina entonces se pasa al else para imprimir que la cuenta finalizó.

Bibliotecas

Todas las funciones que podemos utilizar para nuestros programas son proporcionadas por la bibliotecas que se encuentran en la colección The Python Standard Library, la mayoría de estas son multi-plataforma.

```
#Para utilizar una biblioteca, ésta se debe de importar
import math

x = math.cos(math.pi)

print(x)
```

```
#También se pueden importar todas las funciones de la bibliotecas, de esta manera no se tiene que usar el prefijo
#de la biblioteca, que en el ejemplo anterior fue math
from math import *

x = cos(pi) #No se utiliza el prefijo math

print(x)
```

```
#Otra manera es importar sólo las funciones que se necesitan
from math import cos, pi

x = cos(pi)

print(x)
```

Además de que con las funciones `dir()`, se pueden conocer todas las funciones que contiene la biblioteca con la que se va a trabajar y con la función `help()` se puede conocer cómo utilizar una determinada función de las bibliotecas

Graficación

Para hacer graficaciones se tiene que hacer uso de la biblioteca Matplotlib, con ella se pueden generar gráficas en 2D y 3D.

Para comenzar el programa se importan las bibliotecas para poder utilizar las funciones, y se generan los puntos con los que se van a trabajar, después se definen los ejes y al final se guarda la gráfica para que esta se pueda imprimir correctamente.

```
#Esta línea se ocupa para que las gráficas que se generen queden embebidas dentro de la página
%pylab inline
```

```
#Importando las bibliotecas
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

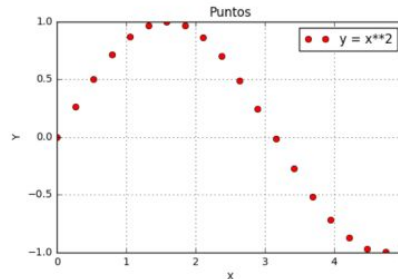
```
#Datos de entrada
x = linspace(0, 5, 20) #Generando 10 puntos entre 0 y 5
```

```
fig, ax = plt.subplots(facecolor='w', edgecolor='k')
ax.plot(x, sin(x), marker="o", color="r", linestyle='None')
```

```
ax.grid(True)
ax.set_xlabel('X') #Etiqueta del eje x
ax.set_ylabel('Y') #Etiqueta del eje y
ax.grid(True)
ax.legend(['y = x**2'])

plt.title('Puntos')
plt.show()
```

```
fig.savefig("gráfica.png") #Guardando la gráfica
```



Ejecución desde ventana de comandos

Para que podamos ejecutar el código tenemos que escribir el comando:

python nombre_archivo.py

Previamente se tiene que tener el código guardado con la extensión “.py”

Entrada de datos

Para pedir datos al usuario es necesario asignar una variable en la cual se guardará el dato solicitado, posteriormente se pone el tipo de dato que se va a almacenar y posteriormente se escribe input().

```
nombre_variable = tipo_de_dato(input())
```

Conclusión:

El desarrollo de esta práctica fue de gran importancia ya que aprendimos más funciones que podemos utilizar para hacer funcionar nuestro programa, el saber utilizar y aplicar las estructuras de control selectivas y repetitivas es importante en todo lenguaje, nosotros ya teníamos conocimiento de cómo funciona esto, pero solo nos queda aprender las funciones que se pueden aplicar en este lenguaje así como su sintaxis, además de que también es importante conocer las bibliotecas disponibles para realizar determinadas acciones y también conocer la sintaxis para la entrada de datos en nuestros programas, ya que la mayoría siempre solicita datos determinados al usuario.

Estas prácticas son sencillas de entender, pero son de vital importancia para aprender un lenguaje de programación nuevo, conforme aprendamos lo básico, podremos avanzar a realizar programas más avanzados y más optimizados.

Referencias:

Manual de prácticas Estructuras de Datos y Algoritmos I