	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: MARCO ANTONIO MARTINEZ QUINTANA

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS I

Grupo: 17

No de Práctica(s): 10

Integrante(s): José Luis Arroyo Chavarría

No. de Equipo de cómputo empleado: 1

No. de Lista o Brigada: 5

Semestre: 2

Fecha de entrega: 07/04/2020

Observaciones:

CALIFICACIÓN: _____

Objetivo:

Aplicar las bases del lenguaje de programación en Python

Introducción:

Todos los lenguajes de programación tienen diferentes tipos de bases para hacer varios trabajos que el usuario desee y Python es la excepción y estos son algunos que utiliza:

- **If:**

La declaración IF sirve para ejecutar código dependiendo del resultado de una condición.

- **If-else:**

Este tipo de declaraciones se usan para dar una opción en el caso de que la condición no se cumpla.

- **If-elif-else**

Este tipo de declaraciones sirve para generar varios casos de prueba. En otros lenguajes es similar a case o switch

- **Ciclo while**

Un ciclo es la manera de ejecutar una o varias acciones repetidamente. A diferencia de las estructuras IF o IF-ELSE que sólo se ejecutan una vez. Para que el ciclo se ejecute, la condición siempre tiene que ser verdadera.

- **Ciclo for**

Este ciclo es el más común usado en Python, se utiliza generalmente para hacer iteraciones en una lista, diccionarios y arreglos.

- **Bibliotecas**

Todas las funcionalidades de Python son proporcionadas a través de bibliotecas que se encuentran en la colección de The Python Standard Library, la mayoría de estas bibliotecas son multi-plataforma.

- **Ejecución desde ventana de comandos**

Un código puede ser guardado en archivos de texto con la extensión '.py'. Para ejecutarlo desde la ventana de comandos se escribe el comando:

python nombre_archivo.py

- **Entrada de datos**

Al igual que en otros lenguajes, se le puede pedir al usuario que introduzca ciertos datos de entrada cuando se ejecute un programa. Pero la diferencia es que no se puede hacer desde la notebook, ya que los datos se introducen en las celdas.

- Input()

Los datos que recibe la función input() son de tipo string, por lo que se tienen que transformar a entero con la función int() para poder realizar operaciones aritméticas.

Desarrollo y resultados:

- **Código**

1. If

```
def obtenerMayor(param1, param2):
    if param1 < param2:
        print('{} es mayor que {}'.format(param2, param1))
obtenerMayor(5, 7)
                                    obtenerMayor(7, 5) #No imprime nada
```

2. If (True)

```
x = y = z = 3
if x == y == z:
    print(True)
```

3. If – else

```
def obtenerMayorv2(param1, param2):
    if param1 < param2:
        return param2
    else:
        return param1
print("El mayor es {}".format( obtenerMayorv2(4, 20) ))
print("El mayor es {}".format( obtenerMayorv2(11, 6) ))
```

4. If – else (True : Flase)

```
def obtenerMayor_idiom(param1, param2):
    #La variable valor va a tener el valor de param2 is el if es verdadero
    #de lo contrario tendra el valor de param1
    valor = param2 if (param1 < param2) else param1
```

```
    return valor
print ("El mayor es {}".format( obtenerMayor_idiom(11, 6) ))
```

5. if-elif-else

```
def numeros(num):
    if num==1:
        print ("tu numero es 1")
    elif num ==2:
        print ("el numero es 2")
    elif num ==3:
        print ("el numero es 3")
    elif num ==4:
        print ("el numero es 4")
    else:
        print ("no hay opcion")
numeros(2)
                numeros(5)
```

6. if-elif-else (2)

```
def numeros_idiom(num):
    #La tupla tiene las opciones validas
    if num in (1,2,3,4):
        print("tu numeros es {}".format(num))
    else:
        print("{} no es una opcion".format(num))
numeros_idiom(2)
                        numeros_idiom(5)
```

7. Estructura de control selectiva anidada

```
def obtenerMasGrande(a, b, c):
    if a > b:
        if a > c:
            return a
```

```
        else:
            return c
    else:
        if b > c:
            return b
        else:
            return c
print ("El más grande es {}".format(obtenerMasGrande(7,13,1) ))
```

8. While

- ```
  #Ejemplo 1
  def cuenta(limite):
      i = limite
      while True:
          print (i)
          i = i -1
          if i == 0:
              break #Rompiendo el ciclo
  cuenta(10)
  ```
- ```
  #Ejemplo 2
  def factorial(n):
      i = 2
      tmp = 1
      while i <n+1:
          tmp = tmp * i
          i = i + 1
      return tmp
  print(factorial(4))
  print(factorial(6))
  ```

9. Ciclo for

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

- #La funcion range() sirve para generar una lista
```
for x in range(5): #este caso es equivalente a range(0,5)
    print(x)
```
- #Tambien se puede inizializar desde numeros negativos
```
for x in range(-5,2):
    print(x)
```
- for num in ["uno", "dos", "tres", "cuatro"]:
```
print(num)
```

10. Iteración en diccionarios

- #Creando un diccionario

```
elementos = { 'hidrogeno': 1, 'helio': 2, 'carbon': 6}

for llave, valor in elementos.items():
    print(llave, " = ", valor)
```

- #Obteniendo solo las llaves

```
for llave in elementos.keys():
    print(llave)
```

- #Obteniendo solo los valores

```
for valor in elementos.values():
    print(valor)
```

- #Si se necesita iterar utilizando un indice

```
for idx, x in enumerate(elementos):
    print("El indice es: {} y el elemento: {}".format(idx, x))
```

- def cuenta_idiom(limite):
```
    for i in range(limite, 0, -1):
        print(i)
    else: #Corresponde al for, NO al IF
        print("Cuenta finalizada")
 cuenta_idiom(5)
```

- #Se rompe el ciclo y la sentencia else del for no se ejecuta

```
def cuenta_idiomv2(limite):
```

```
    for i in range(limite, 0, -1):
        print(i)
        if i ==3:
            break #se rompe el ciclo
    else: #Corresponde al FOR, NO al IF
        print("Cuenta finalizada")
cuenta_idiomv2(5)
```

11. Bibliotecas

- #Para utilizar una biblioteca, esta se debe de importar

```
import math
x = math.cos(math.pi)
print(x)
```

- #Tambien se pueden importar todas las funciones de la bibliotecas, de esta manera no se tiene que usar el prefijo

```
#de la biblioteca, que es el ejemplo anterior fue math
from math import *
x = cos(pi) #No se utiliza el prefijo math
print(x)
```

- #Otra manera es importar solo las funciones que necesitan

```
from math import cos, pi
x = cos(pi)
print(x)
```

- #Una vez que la biblioteca esta importada, se pueden conocer las funciones que este contiene

```
print(dir(math))
```

- #Para conocer cómo utilizar las funciones, se puede utilizar la función help

```
help(math.log)
```

- #Se puede definir un alias para llamar a las funciones que tiene la biblioteca math

```
#Esta es la forma más recomendada para importar módulos, ya que de esta manera se sabe de qué modulo proviene la función
```

```
import math as ma
x = ma.cos(ma.pi)
print(x)
```

12. Graficación

```
#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 plot
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("grafica.png") #Guardando la grafica
```

13. Ejecución desde ventana de comandos

```
python   nombre_archivo.py
```

14. Entrada de datos

- #Se pide el nombre al usuario

print("Hola, ¿cómo te llamas?")

#Se leen los datos introducidos por el usuario y se asignan a la variable nombre

nombre = input()

#Se escribe el nombre solicitado

print("Buen día {}".format(nombre))

15. Función input()

print ("---Calculadora---") #Opciones para el usuario

print ("1- Sumar")

print ("2- Restar")

print ("3- Multiplicar")

print ("4- Dividir")

print ("5- Salir")

op = int(input('Opcion: '))

Captura de pantalla

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

```
7 es mayor que 5
```

```
def obtenerMayor(param1, param2):
    if param1 < param2:
        print('{} es mayor que {}'.format(param2, param1))
obtenerMayor(7, 5) #No imprime nada
```

```
x = y = z = 3
if x == y == z:
    print(True)
```

```
True
```

```
def obtenerMayorv2(param1, param2):
    if param1 < param2:
        return param2
    else:
        return param1
print("El mayor es {}".format( obtenerMayorv2(4, 20) ))
print("El mayor es {}".format( obtenerMayorv2(11, 6) ))
```

```
El mayor es 20
El mayor es 11
```

```
def obtenerMayor_idiom(param1, param2):
    #La variable valor va a tener el valor de param2 is el if es verdadero
    #de lo contrario tendra el valor de param1
    valor = param2 if (param1 < param2) else param1
    return valor
print ("El mayor es {}".format( obtenerMayor_idiom(11, 6) ))
```

```
El mayor es 11
```

```
def numeros(num):
    if num==1:
        print ("tu numero es 1")
    elif num ==2:
        print ("el numero es 2")
    elif num ==3:
        print ("el numero es 3")
    elif num ==4:
        print ("el numero es 4")
    else:
        print ("no hay opcion")
numeros(2)
```

```
el numero es 2
```

```
def numeros(num):
    if num==1:
        print ("tu numero es 1")
    elif num ==2:
        print ("el numero es 2")
    elif num ==3:
        print ("el numero es 3")
    elif num ==4:
        print ("el numero es 4")
    else:
        print ("no hay opcion")
numeros(5)
```

```
no hay opcion
```

```
def numeros_idiom(num):
    #La tupla tiene las opciones validas
    if num in (1,2,3,4):
        print("tu numeros es {}".format(num))
    else:
        print("{} no es una opcion".format(num))
numeros_idiom(2)
```

```
tu numeros es 2
```

```
def numeros_idiom(num):
    #La tupla tiene las opciones validas
    if num in (1,2,3,4):
        print("tu numeros es {}".format(num))
    else:
        print("{} no es una opcion".format(num))
numeros_idiom(5)
```

```
5 no es una opcion
```

```
def obtenerMasGrande(a, b, c):
    if a > b:
        if a > c:
            return a
        else:
            return c
    else:
        if b > c:
            return b
        else:
            return c
print ("El mas grande es {}".format(obtenerMasGrande(7,13,1) ))
```

```
El mas grande es 13
```

```
:
#Ejemplo 1
def cuenta(limite):
    i = limite
    while True:
        print (i)
        i = i -1
        if i == 0:
            break #Rompiendo el ciclo
cuenta(10)
```

```
10
9
8
7
6
5
4
3
2
1
```

```
:
#Ejemplo 2
def factorial(n):
    i = 2
    tmp = 1
    while i <n+1:
        tmp = tmp * i
        i = i + 1
    return tmp
print(factorial(4))
print(factorial(6))
```

```
24
720
```

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

```
1
2
3
4
5
```

```
#La funcion range() sirve para generar una lista
for x in range(5): #este caso es equivalente a range(0,5)
    print(x)
```

```
0
1
2
3
4
```

```
#Tambien se puede inizializar desde numeros negativos
for x in range(-5,2):
    print(x)
```

```
-5
-4
-3
-2
-1
0
1
```

```
for num in ["uno", "dos", "tres", "cuatro"]:
    print(num)
```

```
uno
dos
tres
cuatro
```

```
#Creando un diccionario
elementos = { 'hidrogeno': 1, 'helio': 2, 'carbon': 6}

for llave, valor in elementos.items():
    print(llave, " = ", valor)
```

```
hidrogeno  =  1
helio  =  2
carbon  =  6
```

```
#Obteniendo solo las llaves
for llave in elementos.keys():
    print(llave)
```

```
hidrogeno
helio
carbon
```

```
#Obteniendo solo los valores
for valor in elementos.values():
    print(valor)
```

```
1
2
6
```

```
#Si se necesita iterar utilizando un indice
for idx, x in enumerate(elementos):
    print("El indice es: {} y el elemento: {}".format(idx, x))
```

```
El indice es: 0 y el elemento: hidrogeno
El indice es: 1 y el elemento: helio
El indice es: 2 y el elemento: carbon
```

```
def cuenta_idiom(limite):
    for i in range(limite, 0, -1):
        print(i)
    else: #Corresponde al for, NO al IF
        print("Cuenta finalizada")
cuenta_idiom(5)
```

```
5
4
3
2
1
Cuenta finalizada
```

```
#Se rompe el ciclo y la sentencia else del for no se ejecuta
def cuenta_idiomv2(limite):
    for i in range(limite, 0, -1):
        print(i)
        if i ==3:
            break #se rompe el ciclo
    else: #Corresponde al FOR, NO al IF
        print("Cuenta finalizada")
cuenta_idiomv2(5)
```

```
5
4
3
```

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

x = math.cos(math.pi)

print(x)
```

```
-1.0
```

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

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

print(x)
```

```
-1.0
```

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

x = cos(pi)

print(x)
```

```
-1.0
```

```
#Una vez que la biblioteca esta importada, se pueden conocer las funciones que este contiene
print(dir(math))
```

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmo
d', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log1
0', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

```
#Para conocer como utilizar las funciones, se puede utilizar la funcion help
help(math.log)
```

```
Help on built-in function log in module math:

log(...)
    log(x[, base])

    Return the logarithm of x to the given base.
    If the base not specified, returns the natural logarithm (base e) of x.
```

```
#Se puede definir un alias para llamar a las funciones que tiene la biblioteca math
#Esta es la forma mas recomendada para importar modulos, ya que de esta manera se sabe de que modulo proviene la funcion
import math as ma
x = ma.cos(ma.pi)
print(x)
```

```
-1.0
```

```
#Esta linea se ocupa para que las graficas que se generen queden embebidas dentro de la pagina
%pylab inline

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

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

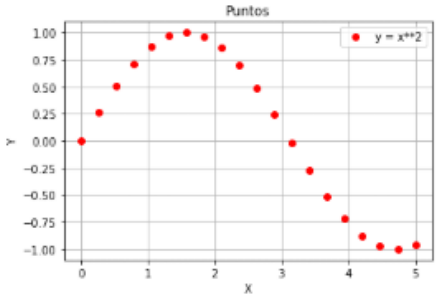
fig, ax = plt.subplots(facecolor='w', edgecolor='k')
ax.plot(x, sin(x), marker="o", color="r", linestyle='None')

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("grafica.png") #Guardando la grafica
```

```
Populating the interactive namespace from numpy and matplotlib
```

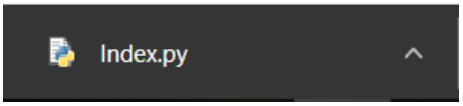


```python
#Se pide el nombre al usuario
print("Hola, ¿cómo te llamas?")
#Se leen los datos introducidos por el usuario y se asignan a la variable nombre
nombre = input()
#Se escribe el nombre solicitado
print("Buen día {}".format(nombre))
```

```
Hola, ¿cómo te llamas?
luis
Buen día luis
```

```python
print ("---Calculadora---")          #Opciones para el usuario
print ("1- Sumar")
print ("2- Restar")
print ("3- Multiplicar")
print ("4- Dividir")
print ("5- Salir")
op = int(input('Opcion: '))
```

```
---Calculadora---
1- Sumar
2- Restar
3- Multiplicar
4- Dividir
5- Salir
Opcion: 1
```



Conclusión:

Con lo aprendido en la práctica nos ayudó a comprender mejor el lenguaje Python y que en si en parte es similar a los demás leguajes en forma de hacer varias funciones o trabajos

Bibliografías y Cibergrafías:

- https://jupyter.org/try