



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación  
Salas A y B

*Profesor:*

Martínez Quintana Marco Antonio

*Asignatura:*

Estructura de Datos y Algoritmos 1.

*Grupo:*

17

*No de Práctica(s):*

10

*Integrante(s):*

Ruiz Godoy Franco

*No. de Equipo de  
cómputo empleado:*

7

*No. de Lista o Brigada:*

*Semestre:*

2020-2

*Fecha de entrega:*

19 de Abril de 2020

*Observaciones:*

**CALIFICACIÓN:**

## Objetivo.

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

## Introducción.

### Estructuras de control selectivas:

#### 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.

### Estructuras de control repetitivas:

#### 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.

#### Bibliotecas más usadas

**NumPy (Numerical Python).** Es una de las bibliotecas más populares de Python, es usado para realizar operaciones con vectores o matrices de una manera eficiente. Contiene funciones de Álgebra Lineal, transformadas de Fourier, generación de números aleatorios e integración con Fortran, C y C++.

**SciPy (Scientific Python).** Es una biblioteca que hace uso de Numpy y es utilizada para hacer operaciones más avanzadas como transformadas discretas de Fourier, Álgebra Lineal, Optimización, etc.

**Matplotlib.** Esta biblioteca es usada para generar una variedad de gráficas en 2D y 3D, donde cada una de las configuraciones de la gráfica es programable. Se puede usar comando de Latex para agregar ecuaciones matemáticas a las gráficas

**Scikit Learn (Machine Learning).** Ésta biblioteca está basada en los anteriores y

contiene algoritmos de aprendizaje de máquina, reconocimiento de patrones y estadísticas para realizar clasificación, regresión, clustering, etc.

**Pandas (Manipulación de datos).** Esta biblioteca es utilizada para manipulación de datos, contiene estructuras de datos llamadas data frames que se asemejan a las hojas de cálculo y a los cuales se le puede aplicar una gran cantidad de funciones.

## Graficación:

**Matplotlib** es una biblioteca usada para generar gráficas en 2D y 3D, donde cada una de las configuraciones de la gráfica es programable.

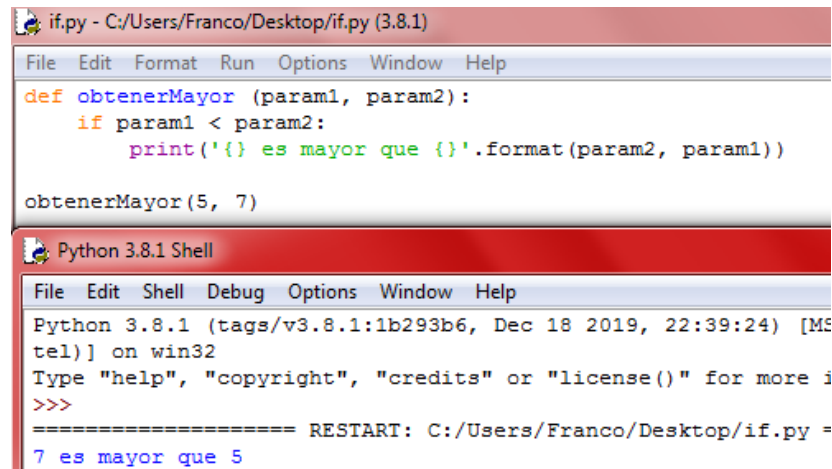
Ejecución desde ventana de comandos.

Todo el código que se ha visto hasta el momento puede ser guardado en archivos de texto plano con la extensión '.py'. Para ejecutarlo desde la ventana de comandos se escribe el comando: `python nombre_archivo.py`

## Desarrollo.

### Estructuras de control selectivas:

#### If



The screenshot shows a Python IDE window titled 'if.py - C:/Users/Franco/Desktop/if.py (3.8.1)'. The code defines a function `obtenerMayor` that takes two parameters, `param1` and `param2`. Inside the function, there is an `if` statement: `if param1 < param2:` followed by an indented `print` statement: `print('{} es mayor que {}'.format(param2, param1))`. Below the function definition, the function is called with `obtenerMayor(5, 7)`. The output window, titled 'Python 3.8.1 Shell', shows the execution result: `7 es mayor que 5`.

#### 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) ))  
print ("*****")  
  
def obtenerMayor_idiom(param3, param4):  
    valor = param4 if (param3 < param4) else param3  
    return valor  
print ("El mayor es {}".format( obtenerMayor_idiom(11, 6) ))
```

```

El mayor es 20
El mayor es 11
*****
El mayor es 11

```

## if-elif-else

```

def numeros(num):
    if num==1:
        print("Tu numero es 1")
    elif num==2:
        print("Tu numero es 2")
    elif num==3:
        print("Tu numero es 3")
    elif num==4:
        print("Tu numero es 4")
    else:
        print ("No hay opción")

numeros (2)
numeros (5)

print("*****")

def numeros_idiom(num2):
    if num2 in (1,2,3,4):
        print("Tu numero es {}".format(num2))
    else:
        print("{} No es una opcion".format(num2))

numeros_idiom(2)
numeros_idiom(5)

print("*****")
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) ))

Tu numero es 2
No hay opción
*****
Tu numero es 2
5 No es una opcion
*****
El mas grande es 13

```

## Estructuras de control repetitivas:

### Ciclo while

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

print("*****")

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))
```

### Ciclo for.

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

print("*****")
for a in range(5):
    print (a)
print("*****")

for b in range(-5,2):
    print (b)

print("*****")

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

Python 3.8.1 (tags/v3.8.1:622dcf4, May 3 2019) on win32  
Type "help", "copyright()", "credits()" or "quit()" for more  
>>>  
===== REPL  
1  
2  
3  
4  
5  
\*\*\*\*\*  
0  
1  
2  
3  
4  
\*\*\*\*\*  
-5  
-4  
-3  
-2  
-1  
0  
1  
\*\*\*\*\*  
uno  
dos  
tres  
cuatro

```

#Iteración en diccionarios
elementos = { 'hidrogeno': 1, 'helio':2, 'carbon':6}

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

print("*****")

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

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

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

print("*****")

def cuenta_idiom(limite):
    for i in range(limite, 0, -1):
        print(i)
    else:
        print("Cuenta finalizada")
cuenta_idiom(5)

print("*****")
def cuenta_idiomv2(limite):
    for i in range(limite, 0, -1):
        print(i)
        if i == 3:
            break
    else:
        print("Cuenta finalizada")
cuenta_idiomv2(5)

```

```

hidrogeno = 1
helio = 2
carbon = 6
*****
hidrogeno
helio
carbon
*****
El indice es: 0 y el elemento: hidrogeno
El indice es: 1 y el elemento: helio
El indice es: 2 y el elemento: carbon
*****
5
4
3
2
1
Cuenta finalizada
*****
5
4
3

```

## Bibliotecas.

```

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

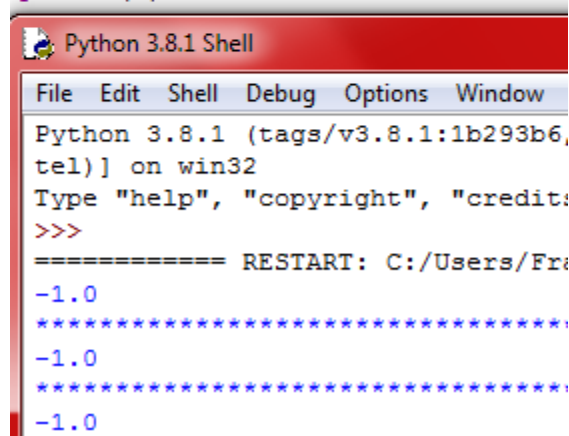
print ("*****")

from math import *
x = cos(pi)
print (x)

print ("*****")

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

```



```

Python 3.8.1 Shell
File Edit Shell Debug Options Window
Python 3.8.1 (tags/v3.8.1:1b293b6,
tel)] on win32
Type "help", "copyright", "credits
>>>
===== RESTART: C:/Users/Fra
-1.0
*****
-1.0
*****
-1.0

```

## **Conclusiones.**

Las estructuras de control selectivas, las estructuras de control repetitivas, las bibliotecas son esenciales para cualquier lenguaje de programación, algo básico que debe de saber. Ya que las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa. Con las estructuras de control se puede: De acuerdo con una condición, ejecutar un grupo u otro de sentencias. Las bibliotecas por que contienen el código objeto de muchos programas que permiten hacer cosas comunes, como leer el teclado, escribir en la pantalla, manejar números, realizar funciones matemáticas, etc.

## **Referencias.**

- García Cano, E. E. G. C., & Solano Gálvez, J. A. S. G. (2017, enero 20). Guía práctica de estudio 10: Introducción a Python (II). Recuperado 19 de abril de 2020, de [http://lcp02.fib.unam.mx/static/docs/PRACTICAS\\_EDA1/eda1\\_p10.pdf](http://lcp02.fib.unam.mx/static/docs/PRACTICAS_EDA1/eda1_p10.pdf)