	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	110/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Guía práctica de estudio 10: Introducción a Python (II).

---





---

***Elaborado por:***

M.C. Edgar E. García Cano  
Ing. Jorge A. Solano Gálvez

***Autorizado por:***

M.C. Alejandro Velázquez Mena

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	111/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Guía práctica de estudio 10: Introducción a Python (II).

### Objetivo:

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

### Actividades:

- Aplicar estructuras de control selectivas
- Aplicar estructuras de control repetitivas
- Usar las bibliotecas estándar
- Generar una gráfica
- Ejecutar un programa desde la ventana de comandos
- Pedir datos al usuario al momento de ejecutar un programa

### Estructuras de control selectivas

if

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

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

```
obtenerMayor(5, 7)
```


```
7 es mayor que 5
```

```
obtenerMayor(7, 5) #No imprime nada
```

Se puede encadenar más de una una condición sin tener que agregar un operador booleano.

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

```
True
```

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	112/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## if-else

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

```
def obtenerMayorv2(param1,param2):
    if param1 < param2:
        return param2
    else:
        return param1
```

```
print ("El mayor es {}".format( obtenerMayorv2(4, 20) ))
```

El mayor es 20

```
print ("El mayor es {}".format( obtenerMayorv2(11, 6) ))
```


El mayor es 11

Para comparaciones simples, Python no tiene un operador ternario (x ? True : False), pero se puede emular con if-else:

```
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

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	113/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## if-elif-else

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

```
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
```

```
numeros(5)
```

```
no hay opcion
```

En algunos casos, se puede evitar la repetición de código del if-elif-else de la siguiente manera:

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

```
numeros_idiom(2)
```

```
tu numero es 2
```

```
numeros_idiom(5)
```


```
5 no es una opcion
```

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 mas grande es {}".format(obtenerMasGrande(7,13,1) ))
```

```
El mas grande es 13
```

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	114/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## 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.

```
#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:
        tmp = tmp * i
        i = i + 1
    return tmp
```

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

```
24
```

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

```
720
```

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	115/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Ciclo for

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

### Iteración en listas

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

```
1
2
3
4
5
```

```
#La función 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
```

```
#También se puede inicializar desde números 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
```

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	116/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Iteración en diccionarios

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

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

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

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

helio
carbon
hidrogeno
```

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

2
6
1
```

En algunos lenguajes de programación se crea un índice para iterar un conjunto de elementos (for (int i=0; i < elementos.size(); ++i)), sin embargo con Python se puede utilizar la función enumerate() en su lugar.

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

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

Los ciclos for pueden hacer uso del else una vez que terminan de iterar, pero no funciona si se rompe el ciclo.

```
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
```

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	117/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## 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.

Referencia del lenguaje: <https://docs.python.org/3/reference/index.html>

Bibliotecas estándar: <https://docs.python.org/3/library/>

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

x = math.cos(math.pi)

print(x)

-1.0
```

```
#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)

-1.0
```


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

x = cos(pi)

print(x)

-1.0
```



	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	118/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

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

['_doc_', '_loader_', '_name_', '_package_', '_spec_', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floo
r', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamm
a', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tr
unc']

#Para conocer cómo utilizar las funciones, se puede utilizar la función 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 más recomendada para importar módulos, ya que de esta manera se sabe de qué módulo proviene la función
import math as ma

x = ma.cos(ma.pi)

print(x)
-1.0
```

## 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++.

Fuente: <http://www.numpy.org/>


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.

Fuente: <http://www.scipy.org/>

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.

Fuente: <http://matplotlib.org/>

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.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	119/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Fuente: <http://scikit-learn.org/>

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. Fuente: <http://pandas.pydata.org/>


ANEXO 1: En esta guía se explica de manera más detallada el uso de las bibliotecas Numpy y Matplotlib.

Jupyter Notebook GitHub:

[https://github.com/eegkno/FI\\_UNAM/blob/master/02\\_Estructuras\\_de\\_datos\\_y\\_algoritmos\\_1/Anexos/Anexo\\_I.ipynb](https://github.com/eegkno/FI_UNAM/blob/master/02_Estructuras_de_datos_y_algoritmos_1/Anexos/Anexo_I.ipynb)

Jupyter Notebook Visualizador:

[http://nbviewer.jupyter.org/github/eegkno/FI\\_UNAM/blob/master/02\\_Estructuras\\_de\\_datos\\_y\\_algoritmos\\_1/Anexos/Anexo\\_I.ipynb](http://nbviewer.jupyter.org/github/eegkno/FI_UNAM/blob/master/02_Estructuras_de_datos_y_algoritmos_1/Anexos/Anexo_I.ipynb)

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	120/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Graficación

Matplotlib (<http://matplotlib.org/>) es una biblioteca usada para generar gráficas en 2D y 3D, donde cada una de las configuraciones de la gráfica es programable. En el siguiente ejemplo se mostrará la configuración básica de una gráfica.

EL API de matplotlib se encuentra en <http://matplotlib.org/api/index.html>

```
#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
```

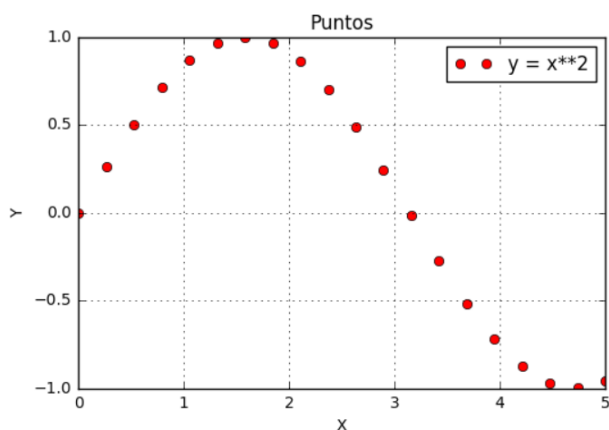
```
#Datos de entrada
x = linspace(0, 5, 20) #Generando 10 puntos entre 0 y 5
```


```
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
```



	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	121/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## 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
```


### Entrada de datos

Al igual que en otros lenguajes, también se puede pedir al usuario que introduzca ciertos datos de entrada cuando se ejecute un programa. Esto no se puede hacer desde la notebook, ya que los datos se introducen en las celdas que se van agregando a lo largo de la página, tal y como se ha venido manejando hasta ahora. Como ejemplo se va a ejecutar el archivo `lectura_datos.py` desde una ventana de comandos.

```
python lectura_datos.py
```

Al momento de ejecutar el programa, se va a pedir al usuario que introduzca su nombre, esto se logra con el siguiente código:

```
#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))
```

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	01
		Página	122/151
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Después de esto se despliega un menú donde se indican las operaciones que puede realizar el usuario, una vez que indicada la operación, se solicitan los datos necesarios para ejecutarla.

```
print ("---Calculadora---")      #Opciones para el usuario
print ("1- Sumar")
print ("2- Restar")
print ("3- Multiplicar")
print ("4- Dividir")
print ("5- Salir")
```

En la siguiente línea se solicita que el usuario especifique alguna de las operaciones, a diferencia de la primera petición, la función `input()` ahora tiene una cadena que se le despliega al usuario. A su vez, 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.

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

## Bibliografía

Tutorial oficial de Python: <https://docs.python.org/3/tutorial/>

Galería de notebooks: <https://wakari.io/gallery>

Matplotlib: <http://matplotlib.org/>