	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Marco Antonio Martínez Quintana

Asignatura: Estructura de Datos y Algoritmos 1

Grupo: 16

No de Práctica(s): Practica 10

Integrante(s): Tierrablanca Oviedo Evelyn

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: Segundo Semestre

Fecha de entrega:

Observaciones:

CALIFICACIÓN: _____

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: *f*

Aplicar estructuras de control selectivas *f*
Aplicar estructuras de control repetitivas *f*
Usar las bibliotecas estándar *f*
Generar una gráfica *f*
Ejecutar un programa desde la ventana de comandos *f*
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.

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 varias 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.

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 inizializar 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
```

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
```

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/

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 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 veriedad 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.
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/

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

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 se le 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 hast 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)
```

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: '))
```