



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): Abrego Abascal Diego

*No. de Equipo de
cómputo empleado:* -

No. de Lista o Brigada: 1

Semestre: 2

Fecha de entrega: 21/04/2020

Observaciones:

CALIFICACIÓN: _____

Introducción a Python II

Introducción

Es esta practica se revisaran las estructuras de control y cíclicas que proporciona Python, así como las principales bibliotecas usadas dentro del lenguaje.

Las estructuras de control a revisar son if, else y el-if. Su sintaxis es la siguiente:

if condición:

 instrucciones

 (Si la condición se cumple se ejecutan estas instrucciones)

elif otra_condición:

 instrucciones

 (Si la primera condición no se cumple, se evalúa esta y si es verdadera se ejecutan las instrucciones dentro del elif, si no se pasa a la siguiente parte del código)

else:

 instrucciones

 (Si no se cumplen las condiciones antes descritas se ejecutan estas instrucciones)

Las estructuras cíclicas a revisar con while y for y su sintaxis es la siguiente:

while condición:

 instrucciones

 (las instrucciones se ejecutan una y otra vez hasta que la condición se vuelva falsa o un break dentro del bucle lo detenga)

for variable_que_itera in lista_a_recorrer:

 instrucciones

 (las instrucciones se repiten una y otra vez hasta que la variable haya iterado sobre toda la lista)

Objetivo

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

Desarrollo

1. If

```
def obtenerMayor(n1,n2):  
    #compara ambas variables y si la comparacion  
    #resulta verdadera ejecuta la instruccion  
    if n1 < n2:  
        print(str(n1)+" es menor que "+str(n2))  
  
obtenerMayor(3,7)  
  
obtenerMayor(7,3)  
#no imprime nada ya que la condicion es falsa  
  
x = y = z = 3  
if x == y == z:  
    #Se muestra como se puede agregar mas de una condicion  
    print(True)  
  
input()
```

C:\windows\py.exe

3 es menor que 7
True

2. If-else

```

def obtenerMayorv2(n1,n2):
    #compara ambas variables y si la comparacion
    #resulta verdadera regresa un valor, si no regresa el otro
    if n1 < n2:
        return n2
    else:
        return n1

print("El mayor es: "+str(obtenerMayorv2(20,5)))
print("El mayor es: "+str(obtenerMayorv2(7,50)))

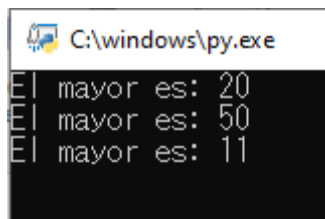
#el operador ternario se puede simular con if else

def mayorTernario(n1, n2):
    #si la condicion es verdadera asigna n2 de lo contrario, asigna n1
    valor = n2 if (n1 < n2) else n1
    return valor

print("El mayor es: "+str(mayorTernario(11,6)))

input()

```



C:\windows\py.exe

```

El mayor es: 20
El mayor es: 50
El mayor es: 11

```

```

def numeros(n):
    #Genera multiples condiciones y si no se cumple una pasa a la
    #siguiente hasta encontrar la verdadera o llegar al else
    if n==1:
        print("tu numero es 1")
    elif n==2:
        print("tu numero es 2")
    elif n==3:
        print("tu numero es 3")
    elif n==4:
        print("tu numero es 4")
    else:
        print("no hay opcion")

numeros(2)
numeros(5)

def numsTupla(n):
    if n in (1,2,3,4):
        print("Tu numero es: "+n)
    else:
        print(n+" no es una opcion valida")


numeros(3)
numeros(7)

def masgrandede3(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: "+str(masgrandede3(7,8,9)))

input()

```

 C:\windows\py.exe

```

tu numero es 2
no hay opcion
tu numero es 3
no hay opcion
El mas grande es: 9

```

```


def cuenta(limite):
    i = limite
    while True:
        #ciclo infinito while
        print(i)
        i=i-1
        if i==0:
            break
        #sin embargo el ciclo de control se rompera cuando i=0
        #ya que el if se volvera verdadero y se ejecara el break
cuenta(10)

def factorial(n):
    i=2
    tmp=1
    while i < n+1:
        #La condicion se ejecara mientras i sea menor al numero que entro +1
        #con cada ciclo de ejecucion, i aumenta en 1
        tmp=tmp*i
        i=i+1
    return tmp

print(factorial(4))

print(factorial(6))

```

 C:\windows\py.exe

```

10
9
8
7
6
5
4
3
2
1
24
720

```

```

for x in [1,2,3,4,5]:
    #se itera el valor de x automaticamente recorriendo la lista
    print (x)

#funcion range genera una lista
#se genera la lista que inicia en 0 y hasta antes del numero indicado
for x in range(5):
    print(x)

#en este caso inicia en el -5 y llega hasta el 1
for x in range(-5,2):
    print(x)

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

```



```

1
2
3
4
5
0
1
2
3
4
-5
-4
-3
-2
-1
0
1
uno
dos
tres
cuatro

```

6. Iteraciones en Diccionarios

```

#se crea el diccionario
elementos = {'hidrogeno':1, 'helio':2, 'carbon':6}

for llave, valor in elementos.items():
    #se asignan los valores del diccionario a las variables valor y llave, se
    #imprimen y despues se itera a la siguiente posicion del diccionario
    print(llave, "=", valor)

for llave in elementos.keys():
    #solo se itera en las llaves de cada elemento del diccionario
    print(llave)

for valor in elementos.values():
    #solo se itera en los valores de cada elemento del diccionario
    print(valor)

#iterando con indices
for idx, x in enumerate(elementos):
    print("El indice es: "+str(idx)+" y el elemento: "+str(x))

def cuenta_idiom(limite):
    for i in range(limite, 0, -1):
        print(i)
    else:
        #else que corresponde al for y no a un if
        print("Cuenta finalizada")

cuenta_idiom(5)

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

cuenta_idiomv2(6)

```



```

C:\windows\py.exe
hidrogeno = 1
helio = 2
carbon = 6
hidrogeno
helio
carbon
1
2
6
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
6
5
4
3

```

7. Biblioteca Math


```

import math
#from math import *           se importan todas las funciones de la biblioteca
#from math import cos,pi     se importan solo las funciones que se necesitan
#import math as ma           permite poner un alias a la biblioteca importada
#x = ma.cos(ma.pi)

x = math.cos(math.pi)

print(x)

#La funcion dir permite conocer las funciones existentes en una biblioteca
#previamente importada
print(dir(math))

#La funcion help permite conocer el como usar las funciones
help(math.log)

```

C:\windows\py.exe

```

h', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist',
e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gam
ma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma
', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 're
mainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
Help on built-in function log in module math:

log(...)
    log(x, [base=math.e])
    Return the logarithm of x to the given base.

    If the base not specified, returns the natural logarithm (base e) of x.

```

8. Graficando con matplotlib

```

%pylab inline
import math
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D as ax

x = np.arange(0, 10, 0.1)
y = np.sin(x)

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

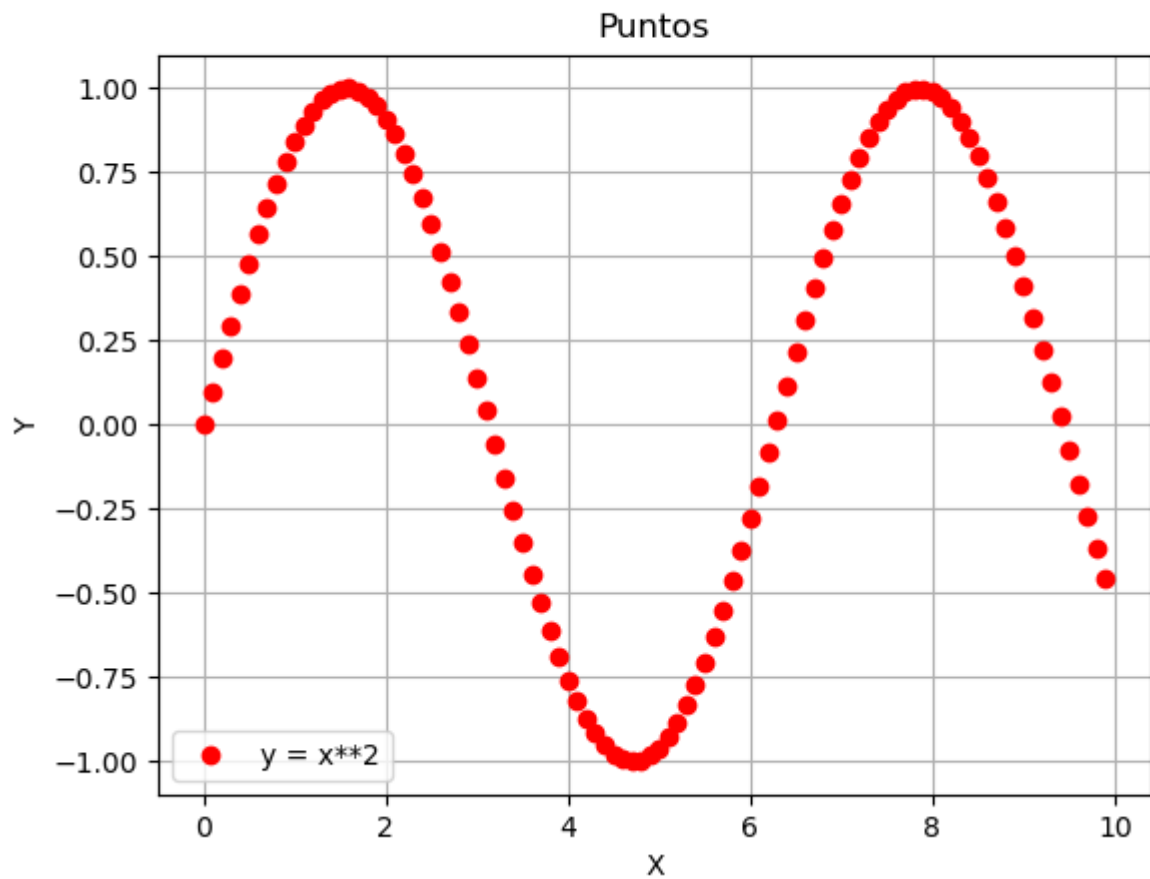
ax.grid(True)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.grid(True)
ax.legend(["y = x**2"])

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

fig.savefig("grafica.png")

```

Figure 1

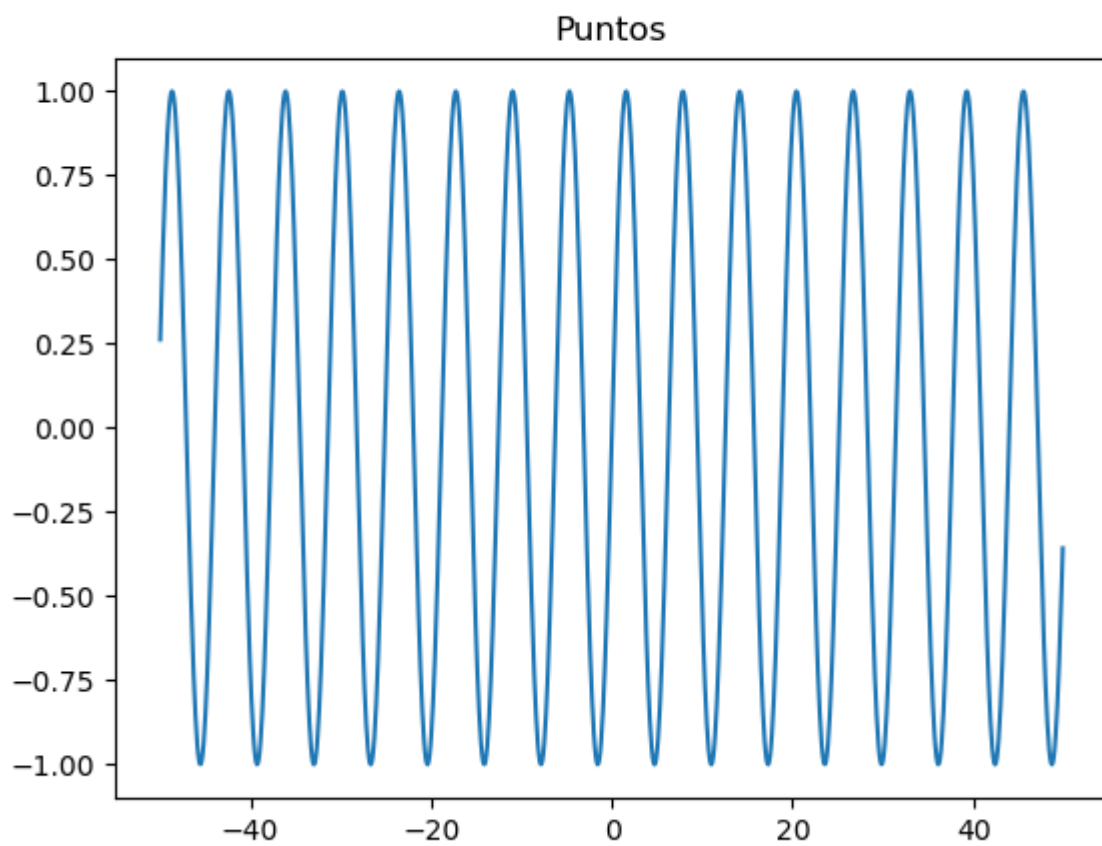


```
%pylab inline
import math
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D as ax

x = np.arange(-50, 50, 0.1)
y = np.sin(x)
plt.plot(x,y)

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

Figure 1



9. Calculadora

```
def menu():
    print("--Calculadora--\n1.Sumar\n2.Restar\n3.Multiplicar\n4.Dividir\n5.Salir\n")


def suma():
    n1 = int(input("Ingresa el numero 1: "))
    n2 = int(input("Ingresa el numero 2: "))
    print("El resultado de la suma es: "+str(n1+n2))

def resta():
    n1 = int(input("Ingresa el numero 1: "))
    n2 = int(input("Ingresa el numero 2: "))
    print("El resultado de la resta es: "+str(n1-n2))

def multi():
    n1 = int(input("Ingresa el numero 1: "))
    n2 = int(input("Ingresa el numero 2: "))
    print("El resultado de la multiplicacion es: "+str(n1*n2))

def div():
    n1 = int(input("Ingresa el numero 1: "))
    n2 = int(input("Ingresa el numero 2: "))
    print("El resultado de la division es: "+str(n1/n2))

op=0
while op != 5:
    menu()
    op = int(input("Elige una operacion: "))
    if op == 1:
        suma()
    elif op == 2:
        resta()
    elif op == 3:
        multi()
    elif op == 4:
        div()
    elif op == 5:
        print("\nHasta Luego\n")
        input()
    else:
        print("Elige una opcion valida")
```

 C:\windows\py.exe

--Calculadora--

- 1.Sumar
- 2.Restar
- 3.Multiplicar
- 4.Dividir
- 5.Salir

Elige una operacion: 1

Ingresa el numero 1: 5

Ingresa el numero 2: 6

El resultado de la suma es: 11

--Calculadora--

- 1.Sumar
- 2.Restar
- 3.Multiplicar
- 4.Dividir
- 5.Salir

Elige una operacion: 3

Ingresa el numero 1: 15

Ingresa el numero 2: 3

El resultado de la multiplicacion es: 45

--Calculadora--

- 1.Sumar
- 2.Restar
- 3.Multiplicar
- 4.Dividir
- 5.Salir

Elige una operacion: 4

Ingresa el numero 1: 80

Ingresa el numero 2: 4

El resultado de la division es: 20.0

--Calculadora--

- 1.Sumar
- 2.Restar
- 3.Multiplicar
- 4.Dividir
- 5.Salir

Elige una operacion: 5

Hasta Luego

Conclusión

En esta práctica primeramente se revisaron las estructuras de control if else que permiten al programa tomar una decisión u otra en caso de que la condición no se cumpla, a su vez el elif permite apilar muchas condiciones una detrás de otro resultando en una estructura similar a un switch y como consecuencia ayuda en gran medida a evitar la anidación de varias estructuras if-else que pueden resultar confuso si muchas de estas se llegan a anidar.

Posteriormente, se expusieron las estructuras cíclicas while y for las cuales permiten repetir instrucciones creando un bucle que se rompe cuando la condición especificada al principio se vuelve falsa. En esta sección se volvió a comprobar la versatilidad que ofrece el lenguaje ya que se podían usar distintas funciones para delimitar la instrucción for así como también este ofrece la posibilidad de adjuntarle un else al final de la ejecución del ciclo.

Finalmente, se vieron y usaron algunas de las funciones de las bibliotecas más usadas del lenguaje y entre los usos que se les dio se pudo graficar la función seno con una sencilla interfaz gráfica.

Bibliografía

- Apuntes de clase
- http://lcp02.fi-b.unam.mx/static/docs/PRACTICAS_EDA1/eda1_p10.pdf
- <https://www.python.org>