Laboratorio I

PhD. Alejandro Paredes

FC-UNI

Entrega de trabajos

- En cada entrega se entrega un solo archivo de texto que contenga el script python.

- El script debe contar con tres partes fundamentales :
  A) Encabezado : Esta parte es fija (será especificada por el profesor)
  B) Cuerpo : Implementación del método en particular
  C) Salida (output): escritura de archivos y produción gráficas (ps,eps o pdf).

Entrega de trabajos

- El script debe estar nombrado : x_y_z.py
  x=primer apellido del alumno(a).
  y=primer nombre del alumno(a).
  z=nombre del trabajo que se entrega (LAB1,LAB2,LAB3,LAB4).

- Si no se cumplen las especificiones se disminuira **5 puntos** sobre la nota obtenida.

Encabezado script python


```
#! /usr/bin/python3

from numpy import *

import matplotlib.pyplot as plt  # from pylab import plot,show

import warnings
warnings.filterwarnings("ignore")
```

Interpolación polinomial

Se tiene un conjunto de medidas y_k a ciertos instantes de tiempo t_k con $k = 1, 2, 3, \ldots m$

$$(y_1, t_{,1}); (y_2, t_{,2}); \ldots (y_m, t_{,m})$$

Debemos encontar la función interpolante $f : \mathbb{R} \to \mathbb{R}$ tal que

$$f(t_k) = y_k \qquad k = 1, 2, 3, \ldots, m$$

Condiciones adicionales:

- Pendiente en ciertos puntos.
- Suavidad, monotonicidad, convexidad de la función interpolante.
- Sólo consideraremos el caso unidimensional $f(x)$.

Interpolación polinomial

Ventajas:

- Hacer pasar una curva suave sobre los datos discretos.
- Hallar el valor de una medida entre dos datos.
- Tener acceso a la derivada y a la integral del conjunto de datos.

Desventajas:

- Pobres resultados con datos sujetos a errores significantes.
- Dificultad para capturar discontinuidades.

Interpolación polinomial

Bases de funciones

Conjunto de funciones $\{\phi_1(t), \phi_2(t), \ldots, \phi_n(t)\}$ base de $E(\mathbb{R} : \mathbb{R})$

$$f(t) = \sum_{j=1}^{n} \lambda_j \phi_j(t)$$

Imponemos que $f(x)$ pase por los datos

$$f(t_k) = y_k = \sum_{j=1}^{n} \lambda_j \phi_j(t_k) \qquad k = 1, 2, \ldots, m$$

$$\mathbf{A}\lambda = \mathbf{y}$$

$$\begin{bmatrix} \phi_1(t_1) & \phi_2(t_1) & \ldots & \phi_n(t_1) \\ \phi_1(t_2) & \phi_2(t_2) & \ldots & \phi_n(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(t_n) & \phi_2(t_n) & \ldots & \phi_n(t_n) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Interpolación polinomial

Monomios

$$\begin{aligned} \phi_j(t) &= t^{j-1} \quad j = 1, 2, \ldots, n \\ f_{n-1}(t) &= \lambda_1 t + \lambda_2 t + \cdots + \lambda_n t^{n-1} \end{aligned}$$

Ejemplo: $(-2, -27), (0, -1), (1, 0)$

$$\mathbf{A}\lambda = \mathbf{y}$$

$$\begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 4 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} -27 \\ -1 \\ 0 \end{bmatrix}$$

Resolviendo para $\lambda = (-1, 5, -4)$

$$f_2(t) = -1 + 5t - 4t^2$$

Interpolación polinomial

$$\ell_j(t) = \frac{\prod_{k=1,k\neq j}^{n} (t - t_k)}{\prod_{k=1,k\neq j}^{n} (t_j - t_k)} \qquad \ell_j(t_i) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \qquad j = 1, 2, \ldots, n$$

La matriz $\mathbf{A}$ en $\mathbf{A}\lambda = \mathbf{y}$ resulta ser la matriz identidad.

$$f_{n-1} = y_1\ell_1 + y_2\ell_2 + \cdots + y_n\ell_n$$

Aqui el trabajo sólo es escribir los polinomios.

Ejemplo: $(-2, -27), (0, -1), (1, 0)$

$$f_2(x)(t) = -27\frac{t(t-1)}{-2(-2-1)} + (-1)\frac{(t+2)(t-1)}{2(-1)}$$

Interpolación polinomial

Polinomios de Newton

$$\pi_0 = 1 \quad \pi_j = \prod_{k=0}^{j-1}(t - t_k) \quad j = 1, 2, \ldots, n$$

$$f_{n-1}(t) = x_1 + x_2(t - t_1) + x_3(t - t_1)(t - t_2)$$
$$+ \cdots + x_n(t - t_1)(t - t_2)\cdots(t - t_{n-1})$$

Nota:

Para $i < j, \pi_j(t_i) = 0$ entonces $\mathbf{A}$ es una matriz triangular L.

Ejemplo: $(-2, -27), (0, -1), (1, 0)$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 3 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -27 \\ -1 \\ 0 \end{bmatrix}$$

$x = (-27, 13 - 4)$

$$f_2(t) = -27 + 13(t + 2) - (t + 2)t$$

Trabajo

- Formar grupos de 3 personas. Cada miembro es responsable de una interpolación: monomial, Lagrange, Newton.
- Hoy - 6 puntos: Entregar las matrices (monomial-Newton) y los coeficientes (Lagrange) correspondientes al conjunto de datos asignado.
- Sábado (06.05) - 14 puntos:

  - Python : entrada matriz (aumentada) y salida vector con coeficientes de interpolación.

  - Graficar de conjunto de datos asignado y tres interpolaciones.

Trabajo

Conjunto de datos 1

$T\ (°C)$	150	160	170	180	190	200	210
$R\,(\%)$	35.5	37.8	43.6	45.7	47.3	50.1	51.2

Conjunto de datos 2

$Q\ (l/h)$	500	700	900	1100	1300	1500	1700	1900
$N\ \ (w)$	365	361.6	370.64	379.68	384.46	395.5	395.95	397

Conjunto de datos 3

$T\ (s)$	200	400	650	1100	1900	2300
C	5.04	4.36	3.45	2.37	1.32	0.71

Sustitución directa (forward)

- Matriz con dimensión pequeña.
- Matriz A: triangular inferior L.

$$\begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \dots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{aligned} x_1 &= b_1/a_{11}, \quad x_2 = (b_2 - a_{21}x_1)/a_{22} \\ x_n &= (b_n - a_{n1}x_1 - a_{n2}x_2 \cdots - a_{n(n-1)}x_{(n-1)})/a_{nn} \\ x_n &= (b_n - \sum_{j=1}^{n-1} a_{nj}x_j)/a_{nn} \\ x_k &= (b_k - \sum_{j=1}^{k-1} a_{kj}x_j)/a_{kk} \end{aligned}$$

Sustitución directa (forward)

Algoritmo: Dada una matriz triangular inferior A de dimensión n y un vector $\mathbf{b}$,

$$\begin{aligned} &\text{for } k = 1 : n \\ &\quad x_k = \frac{b_k - \sum_{j=1}^{k-1} a_{kj} x_j}{a_{kk}} \\ &\text{end} \end{aligned}$$

- Los elementos de la diagonal no puden ser cero.
- Hay problemas si los elementos de la diagonal son próximos a cero.

Sustitución inversa (backward)

- Matriz de dimension pequenña.
- Matriz A: triangular superior U

$$\begin{bmatrix} a_{11} & \cdots & a_{1(n-1)} & a_{1n} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{(n-1)(n-1)} & a_{(n-1)n} \\ 0 & 0 & 0 & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{aligned} x_n &= b_n/a_{nn} \\ x_{n-1} &= (b_{n-1} - a_{(n-1)n}x_n)/a_{(n-1)(n-1)} \\ x_{n-2} &= (b_{n-2} - a_{(n-2)n}x_n - a_{(n-2)(n-1)}x_{n-1})/a_{(n-2)(n-2)} \\ &\vdots \\ x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3 \cdots - a_{1n}x_n)/a_{11} \\ x_k &= (b_k - \sum_{j=k+1}^{n} a_{kj}x_j)/a_{kk} \end{aligned}$$

Sustitución inversa.

Algoritmo: Dada una matriz triangular superior A de dimesión n y un vector $\mathbf{b}$,

$$\text{for } k = n : -1 : 1$$
$$x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{kj} x_j}{a_{kk}}$$
$$\text{end}$$

- El bucle debe comenzar a partir de $k = n - 1$
- Se debe inicializar $x_n = b_n / A_{nn}$

Eliminación de Gauss

- Matriz de dimensión pequeña.
- Matriz A : no es ni L ni U.
- Los coeficientes de la primera columna no son cero ni proximos a cero.
- El objetivo es llevar la matriz A a un matriz U y luego realizar una sustitución backwards.

Método eliminación de Gauss

Solución : $[3 \quad 4 \quad -2]$

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 5 \\ 4 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 2 \end{bmatrix} \rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{array}\right]$$

Transformar un sistema lineal en U o L

- F2 -2F1

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 4 & 0 & 5 & 2 \end{array}\right]$$

- F3-4F1

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{array}\right]$$

- F3+4F2

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 13 & -26 \end{array}\right]$$

- F3/13

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -2 \end{array}\right]$$

Algoritmo eliminación Gauss:

```
DOFOR k = 1, n − 1
 DOFOR i = k + 1, n
   factor = a_{i,k} / a_{k,k}
   DOFOR j = k + 1 to n
     a_{i,j} = a_{i,j} − factor · a_{k,j}
   END DO
   b_i = b_i − factor · b_k
 END DO
END DO
x_n = b_n / a_{n,n}
DOFOR i = n − 1, 1, −1
  sum = b_i
  DOFOR j = i + 1, n
   sum = sum − a_{i,j} · x_j
  END DO
  x_i = sum / a_{i,i}
END DO
```

Curso introductorio de Python

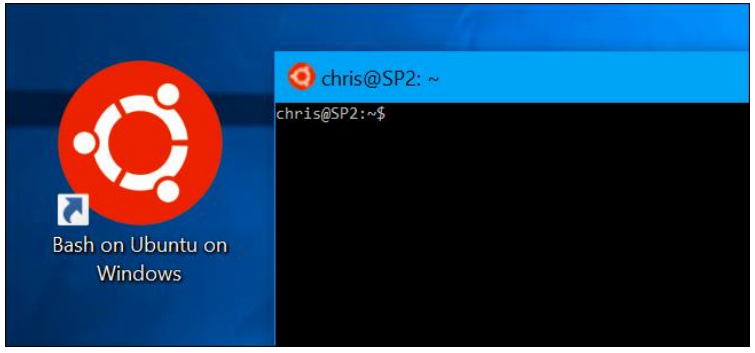
Acceso al SO Linux

- Formatear la computadora e instalar Ubuntu.

- Si tienes Windows 10:

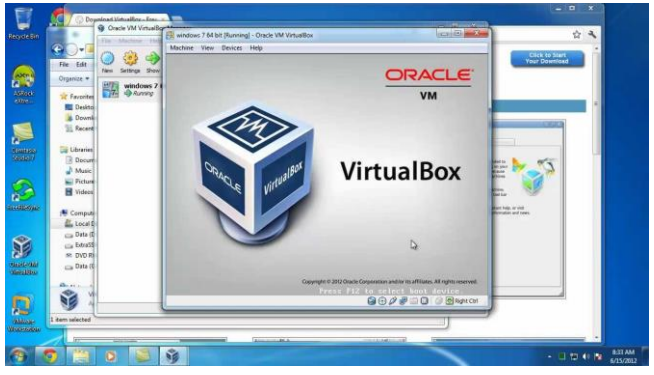
  Windows Subsystem for Linux

  Bash en Ubuntu en Windows

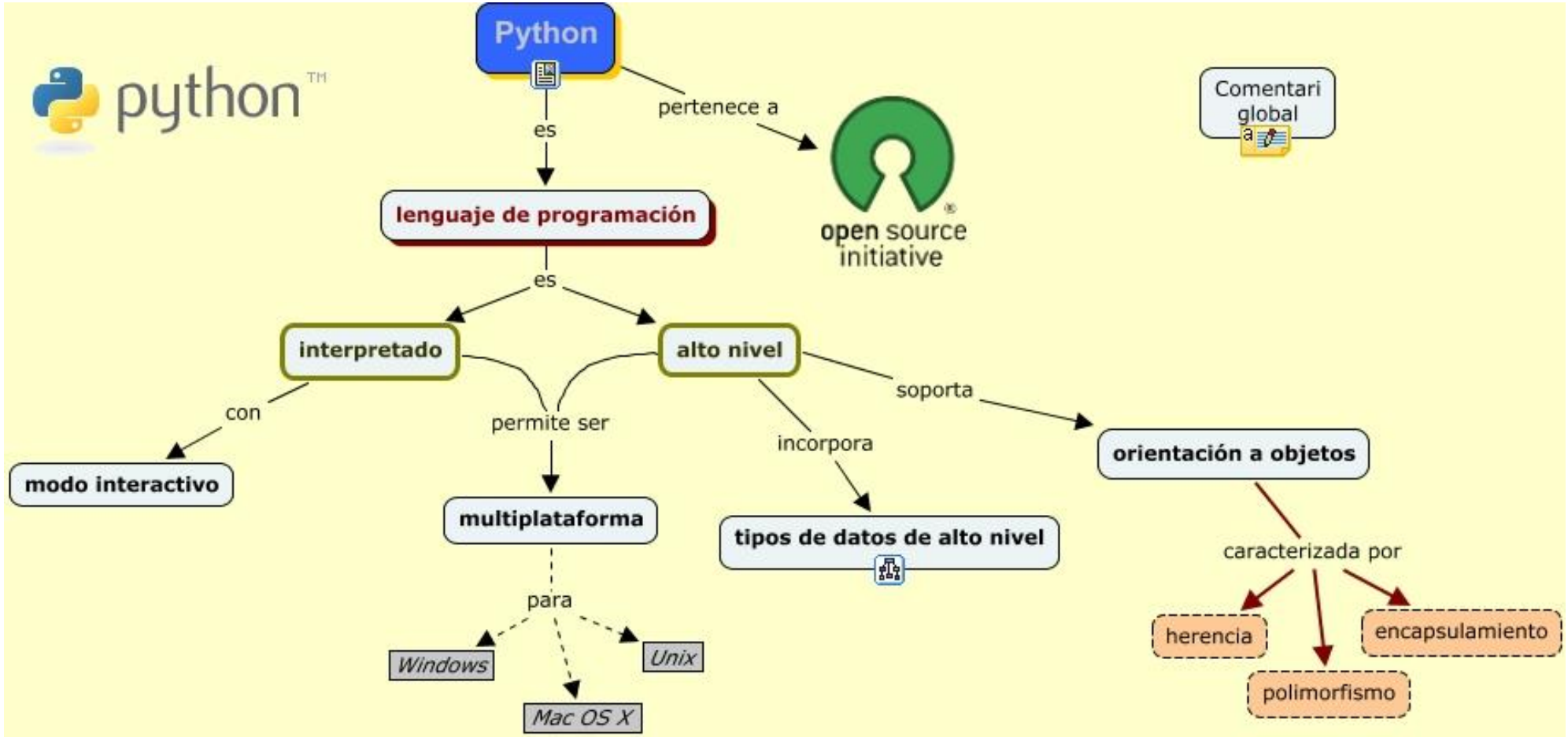


- Si tienes Windows < 10:

  Virtual box (genera una maquina virtual)



Python


python™
Python
es
pertenece a
open source initiative
Comentari global
lenguaje de programación
es
interpretado
alto nivel
con
modo interactivo
permite ser
multiplataforma
incorpora
tipos de datos de alto nivel
soporta
orientación a objetos
caracterizada por
herencia
encapsulamiento
polimorfismo
para
Windows
Unix
Mac OS X

Python

- Interprete de commandos (no se necesita compilar).
- Multipropósito: desarrollo web, desarrollo de juegos, desarrollo de software.
- Aprendizaje rápido e inutitivo.
- Python maneja una sintaxis identada (con márgenes) de carácter **obligatorio**.
- Para separar los bloques de código en Python se debe tabular hacia dentro.
- Python es *case sentive*: hace diferencia entre mayúsculas y minusculas.

Instalar Python en Ubuntu Bash

- sudo apt-get install python3 (Instala python3)
- sudo apt-get install python-pip python3-pip (herramienta para inst. paquetes python)
- sudo pip3 install numpy (Instala librería numpy de Python)
- sudo pip3 install matplotlib (Instala librería matplotlib de Python)

Antes de ejecutar un archivo Python (ejemplo.py) se ejecuta la línea

chmod +x ejemplo.py

Python: Programación básica

Tipos de variables

- Integer : Son números enteros positivos, negativos o cero. Números como el 1, 0, -2834 son admitidos.
- Float : Son números reales. Números como 3.14.15 ó -6.63 X 10^{-34} ó 1.0 son admitidos.
- Complex : Números complejos. Números com 1+2**j** ó -3.5 +0.5j son admitidos. Notar que en python la unidad compleja es **j** y no **i**.
- String : almacena letras en forma de una cadena de caracteres.

Python: Programación básica

Asignación de variables

Integer : x= 1
Float : x=1.0

Float :x=float (1)

Complex : x =1.5 +0j (variable compleja puramente real)
x = complex (1.5) (mismo resultado)
y= complex (1.0,6.0) → y=1.0 +6.0 j

String : x =‘esta es un string’

Python: Programación básica

Output : Imprimir en pantalla

Script_1

```
x=1
y=2
z=2 +3j

print(x)
print ('Este es x=',x,' Este es y= ',y ,'Este es z= ',z)
```

Input : Se entra por teclado el valor de la variable x

Script_2

```
x= input('entra el valor de x: ')
print ('El valor ingresado es',x)
```

Python: Programación básica

Aritmética

Suma : x+y
Resta : x –y
Multiplicación : x*y
División : x/y
Potencia x**y (x elevado a la potencia y)

Se recomienda el uso de paréntesis

Ejemplos

x+2*y <> $x - 2y$
x – y/2 <> $x - \frac{1}{2}y$
3*x**2 <> $3x^2$
x/2*y <> $\frac{1}{2}xy$

Python: Programación básica

Funciones paquetes y modulos

- Python posee funciones integradas.
- Las funciones se encuentran dentro de los paquetes.
- Cuando los paquetes son muy grandes son divididos en módulos.

from math import log (llama al paquete math y da acceso a la función log)
from math import log,exp,sin,cos,sqrt,pi,e (llama al paquete math y las función log,exp,etc)
from math import * (llama al paquete math y todas las funciones contenidas)
from numpy.linalg import inv (llama al paquete numpy, el modulo linalg y la función inv)

Python: Programación básica

Funciones integradas:

log : logaritmo natural
log10: logaritmo base 10
exp : función exponenecial.
sin, cos, tan : funciones seno, coseno y tangente. El argumento debe estar en radianes.
asin,acos,atan : funciones seno^{-1}, coseno^{-1} y tangente^{-1}.
sinh,cosh,tanh : funciones senohyperbólico, cosenohiperbólico y tangentehyperbólica.
sqrt : raiz cuadrada

Python: Programación básica

Comentarios

Para poder hacer al Código de fácil lectura para otros usuarios o inclusive para el desarrolador es común comentar los códigos.

Script_3

```
from math import sin,cos,pi
# Ask the user for the values of the radius and angle
r = float(input("Enter r: "))    # radius
d = float(input("Enter theta in degrees: "))
```

Todo texto que se escribe despues del simbolo # es un comentario. Python ignorará cualquier texto despues del simbolo #.

Python: sentencias if y whyle

La sentencia if

Script_4

```
x = int(input("Enter a whole number no greater than ten: "))
if x>10:
     print("You entered a number greater than ten.")
     print("Let me fix that for you.")
     x = 10
print("Your number is",x)
```

Identación:

- Se debe usar espacios o tabulaicones.
- Se sugiere usar 4 espacios.
- No se recomienda mezclar espacios con tabulaciones.

Python: sentencias if y whyle

Diferentes condicones que se pueden utilizer en la sentecia if

if x==1 : verifica si $x = 1$. Note el doble signo igual.
if x>1 : verifica si $x > 1$
if x>=1 : verifica si $x \geq 1$
if x<1 : verifica si $x < 1$
if x<=1 : verifica si $x \leq 1$
if x!=1 : verifica si $x \neq 1$

Python: sentencias if y whyle

Se pueden utilizar dos condiciones en una sola sentencia

Script_5

Una condición o la otra

```
x = int(input("Enter a whole number: "))
if x>10 or x<1:
    print("Your number is either too big or too small.")
```

Una condición y la otra

```
x = int(input("Enter a whole number: "))
if x<=10 and x>=1:
    print ("Your number is just right.")
```

Python: sentencias if y whyle

Script_6

Sentencia else

```
if x>10:
     print("Your number is greater than ten.")
else:
     print("Your number is fine. Nothing to see here.")
```

Sentencia elif

```
if x>10:
     print("Your number is greater than ten.")
elif x>9:
        print("Your number is OK, but you're cutting it close.")
else:
       print ("Your number is fine. Move along. ")
```

Python: sentencias if y whyle

Sentencia whyle

Las ordenes se ejecutan mientras se verifique la condición (x>10)

Script_7

```
x = int(input("Enter a whole number no greater than ten: "))
while x>10:
        print ("This is greater# than ten. Please try again. ")
        x = int(input("Enter a whole number no greater than ten: "))
print("Your number is",x)
```

Python: Lisas y arreglos

Lista : es una disposición de cantidades una despues de la otra separadas por comas. No es necesario que las cantidades sean del mismo tipo.

Ejemplo

r = [1, 1, 2, 3, 5, 8, 13, 21]
b =[1 , 2 . 5, 3+4. 6 j]

Los elementos de la lista se pueden obtener a partir de expresiones algebraicas.

r = [2*x, x+y, z/sqrt(X**2+y**2)]

Python: Lisas y arreglos

- Los elementos de la lista son accesibles y están enumerados desde zero.
- Los elementos de la lista se pueden operar y ser argumentos de funciones.

Script_8

```
from math import sqrt
r = [ 1. 0, 1. 5, -2. 2 ]
print(r[0])    # imprime primer elemnto de la lista r
length= sqrt( r[0]**2 + r[1]**2 + r[2]**2)
print(length)
```

Se puede añadir un elemento al final de la lista

```
r = [ 1.0, 1.5, -2.2]
x = 0.8
r.append(2*x+1)
print(r)
```

Python: Lisas y arreglos

Arreglos (arrays) Son muy similares a las listas ya que Tambien es un conjunto ordenado de valores. Sin embargo tienen algunas diferencias:

- El número de elementos de un array es fijo. No se puede anadir elementos.
- Todos los elementos de un array deben ser del mismo tipo. No se puede cambiar el tipo de los elementos una vez creado el array.
- Los arrays pueden ser de dimension dos y entenderse como una matriz. Tambien puden tener mayores dimensiones.
- Se pueden hacer operaciones aritméticas con los arrays.
- Las operaciones con arrays son significativamente más rápidas.

Python: Lisas y arreglos

Para crear un array de dimension 4 lleno de ceros

```
from numpy import zeros
a = zeros(4,float)
print(a)
```

Resultado [0. 0. 0. 0.]

Para crear un array de dimension 3X4 lleno de ceros

```
a=zeros([3,4],float)
print(a)
```

Resultado :[[0. 0. 0. 0 .]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

Script_09

Python: Lisas y arreglos

Creamos un array lleno de ceros y luego asignamos valores

Scritpt_10

```
from numpy import zeros

a=zeros([2,2],int)

a[0,1]=1


a[1,0] = -1
print(a)
```

Resultado:

[[0 1]
 [-1 O]]

Python: Lisas y arreglos

Leer un array desde un archivo de texto

El archivo data.txt contiene lo siguiente

1.0
1. 5
-2.2
2.6

Ejecutamos
Script_11

```
from numpy import loadtxt
a= loadtxt(“data.txt",float)
print(a)
```

Resultado

[1.0 1.5 -2.2 2.6]

Python: Lisas y arreglos

Si data.text contiene

1 2 3 4
3 4 5 6
5 6 7 8

Entonces

```
from numpy import loadtxt
a= loadtxt(“data.txt",float)
print(a)
```

Imprimirá un array de dimension 3X4

Python: Lisas y arreglos

Las funciones shape y size dan información sobre el tamaño y la forma del array

Script_12

```
a= array([[1,2,3], [4,5,6]] ,int)
print(a.size)
print(a.shape)
```

El resultado será

6

(2, 3)

Python: Lisas y arreglos

Algunos detalles a tener encuenta con python

Script_13

```
from numpy import array
a= array([1,1] ,int)
b = a
a[0] = 2
print(a)
print(b)
```

Resultado

[2 1]
[2 1]

```
from numpy import array,copy
a= array([1,1] ,int)
b = copy(a)
a[O] = 2
print(a)
print(b)
```

Resultado

[2 1]
[1 1]

Python: Bucles for

Un bucle for es un bucle donde el iterador toma ordenadamente valores a partir de una lista.

Script_14

```
r = [ 1, 3, 5]
for n in r:
      print(n)
      print(2*n)
print("Finished")
```

Resultado

1
2
3
6
5
10
Fnished

Python: Bucles for

Se puede crear una lista a partir de una orden y luego usarla en el bucle for

Script_15

```
r = range(5)

for n in r:
      print("Hola")
```

Resultado

Hola
Hola
Hola
Hola
Hola

La función range(5) crea la lista [0,1,2,3,4]

Python: Bucles for

La forma más comun de usar range en un bucle for

Script_16

```
for n in range(5):
    print(n**2)
```

Resultado

0
1
4
9
16

Variantes de range

range(5) da [0, 1, 2, 3, 4]
range(2,8) da [2, 3, 4, 5, 6, 7]
range(2,20,3) da [2, 5, 8, 11, 14, 17]
range(20,2,-3) da [20, 17, 14, 11, 8, 5]

Python: Funciones definidas por usuario

Se pueden definir funciones que den como resultado un número

El resultado de una función tambien puede ser un array

Script_17

```
def factorial(n):
      f = 1.0
      for k in range(1,n+1):
          f *= k
      return f
```

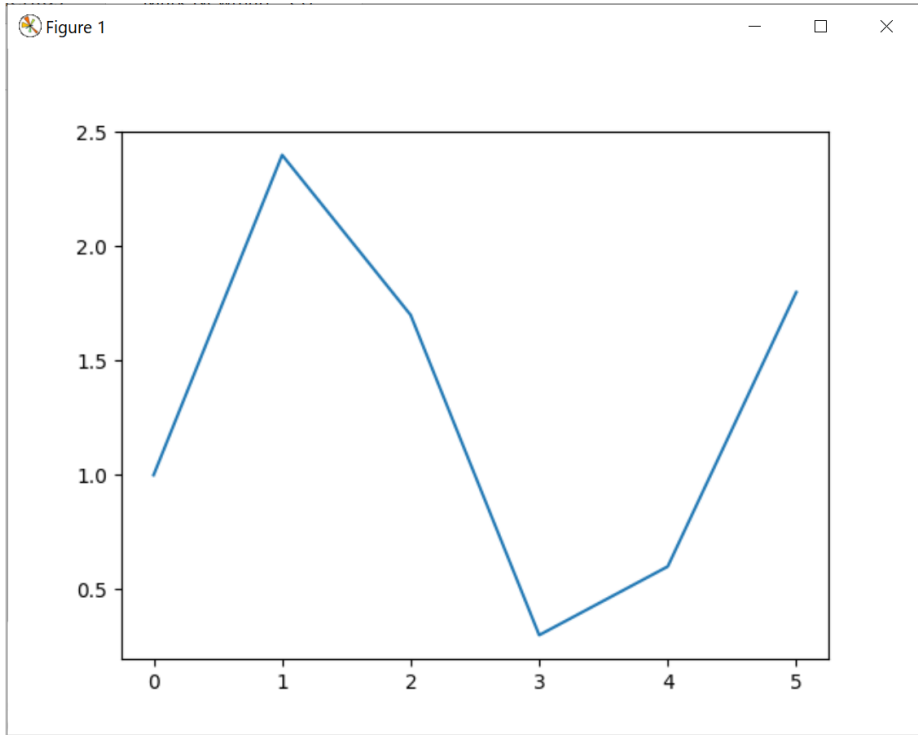
```
def cartesian(r,theta):
      x = r*cos(theta)
      y = r*sin(theta)
      position = [x,y]
      return position
```

Python: Graficas y visualización

Para graficar se llaman a las funciones plot y show del paquete pylab

Script_18

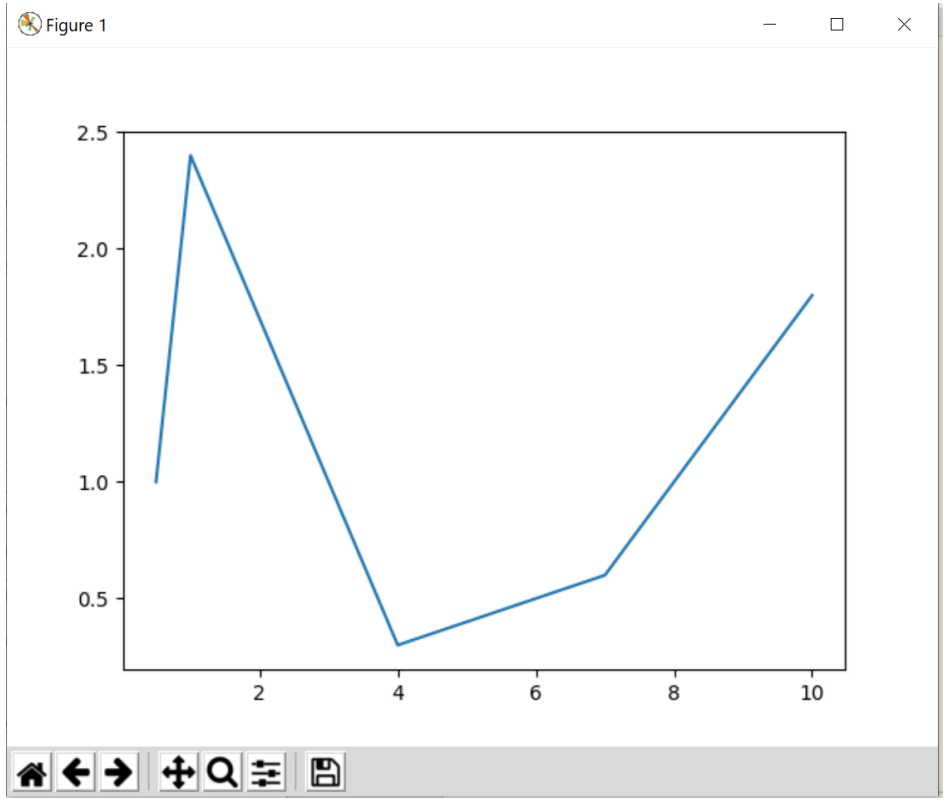
```
from pylab import plot,show
y = [ 1.0, 2.4, 1.7, 0 .3, 0.6, 1.8]
plot (y)
show()
```



Python: Graficas y visualización

Se puede definir los puntos de las abscisas y ordenadas

```
from pylab import plot,show
x = [ 0. 5, 1. 0, 2. 0, 4. 0, 7. 0, 10. 0 ]
y = [ 1.0, 2.4, 1.7, 0.3, 0.6, 1.8]
plot(x,y)
show()
```



Python: Graficas y visualización

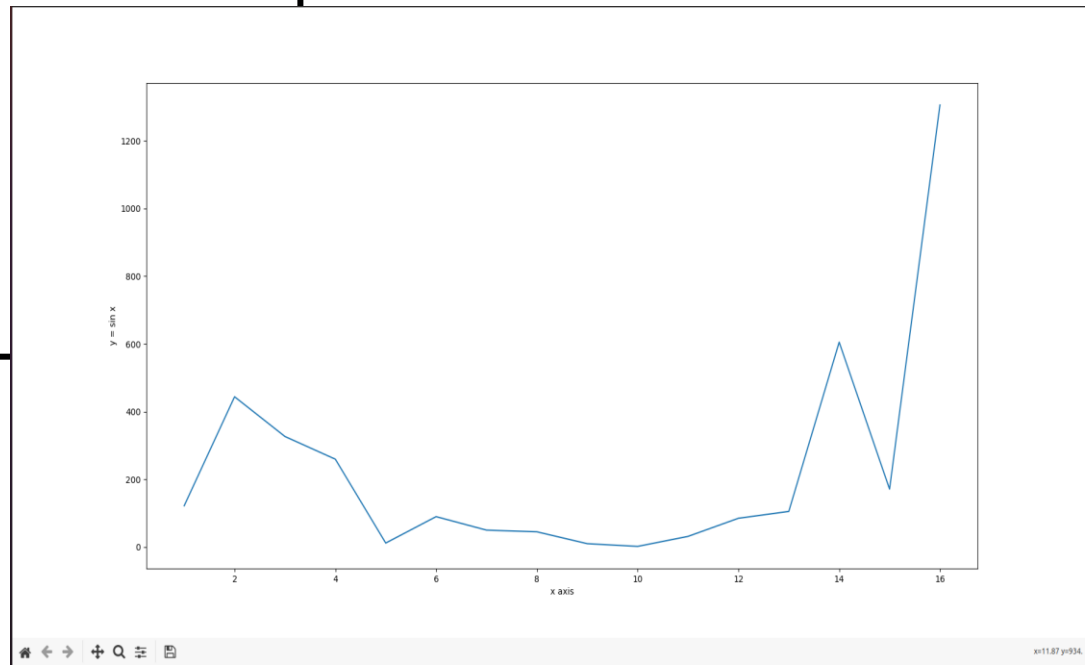
Graficar el archivo plot.text

1	122.189988
2	444.0270889
3	326.7205784
4	259.7201836
5	12.15722275
6	90.15900043
7	1005.310014

.....

Ejecutamos

```
from numpy import loadtxt
from pylab import plot,show
data= loadtxt("plot.text",float)
x = data[: ,0]
y = data [: , 1]
xlabel("x axis")
ylabel("y = sin x")
plot(x,y)
show ()
```



Modulos Python

- Son archivos Python con la extensión .py. Ejemplo mi_modulo.py
- Implementan un conjunto de funciones.
- Son importados desde otros módulos usando el comando **import**.
- Se recomienda que genere un módulo por cada tema abordado en el curso.

Modulos Python


```
#!/usr/bin/python3
from numpy import *
################################
from math import *
################################
import warnings
warnings.filterwarnings("ignore")
################################
def factorial(n):
    f = 1.0
    for k in range(1,n+1):
        f *= k
    return f

def multiplication(num1,num2):

    return num1*num2
```

```
#!/usr/bin/python3
from numpy import *
################################
from math import *
################################
import warnings
warnings.filterwarnings("ignore")
################################
from mi_modulo import *
############################

a=multiplication(3,2)

print(factorial(3),a)
```

Buenas prácticas de programación

- Hacer comentarios.
- No usar funciones que hagan un trabajo muy particular.
- Los nombres de las variables deben tener relación con el dato que representan.
- Mantener un script limpio, ordenado y mínimo.
- No invertir mucho tiempo en la optimización del script. El primer objetivo es que funione !!!