

pyladies



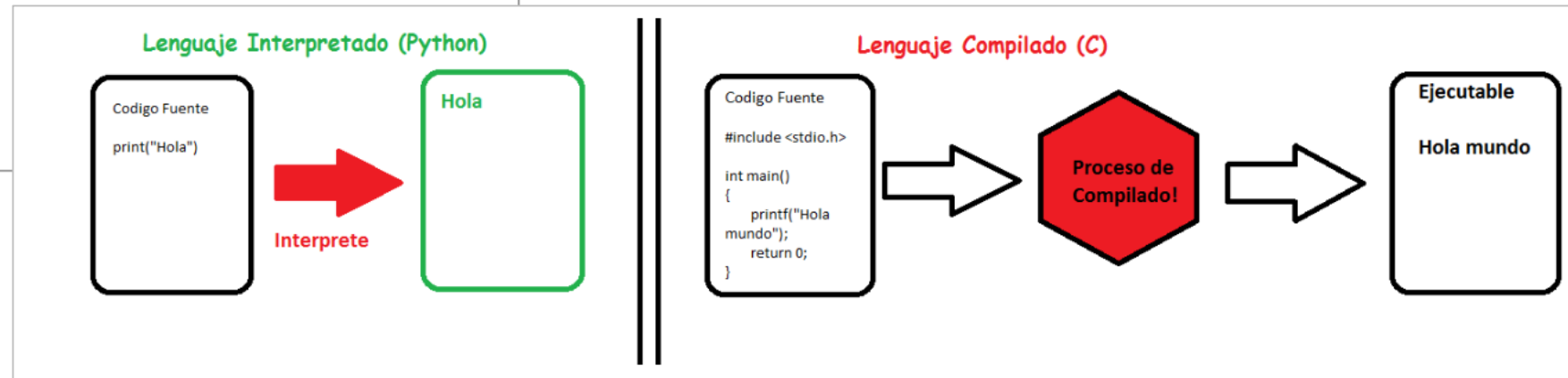
Taller Aprendiendo Python III

ESTHER BALBUENA

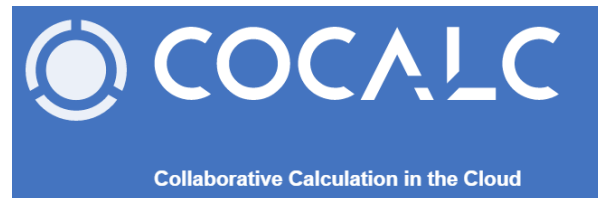
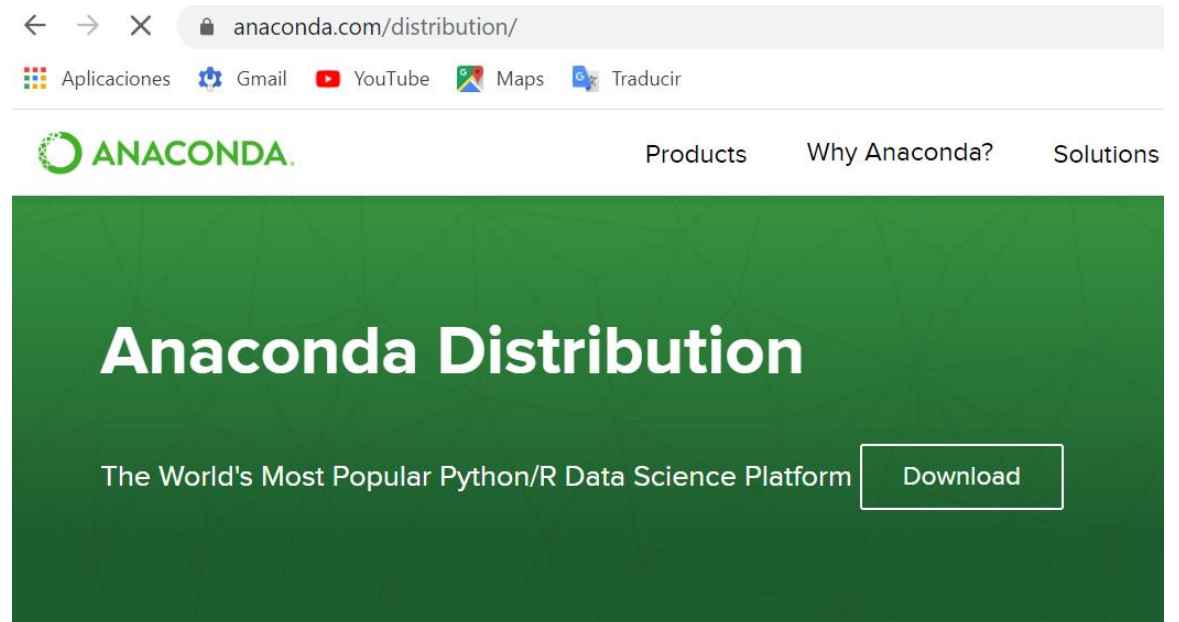
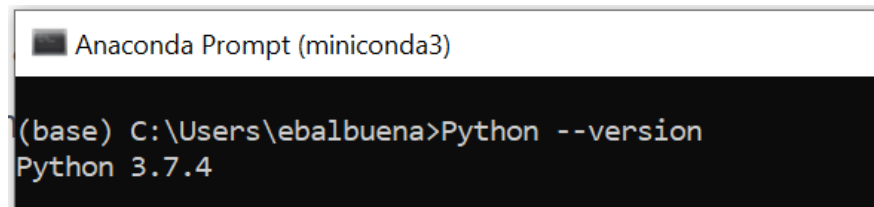
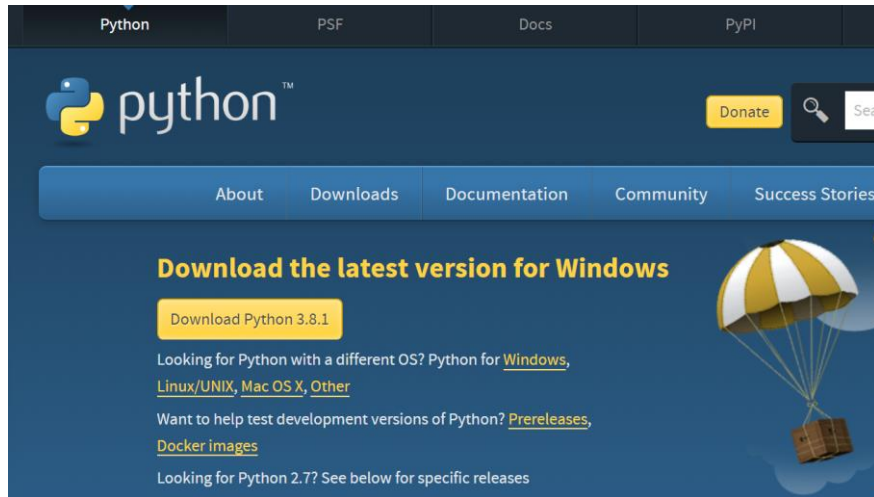
Fundamentos de Python

Python es un lenguaje de programación multiplataforma e interpretado.

- **Multiparadigma:**
Soporta orientación a objetos, programación imperativa y programación funcional.
- **Interpretado**
- **Tipado dinámico**
- **Multiplataforma**
- **Open source**



<https://www.python.org/downloads/>



Elementos del lenguaje



VARIABLES

- Una variable es un espacio para almacenar datos modificables, en la memoria de un ordenador.

Sintaxis:

```
nombre_de_la_variable = valor_de_la_variable
```

- Cada variable, tiene un nombre y un valor, el cual define a la vez, el tipo de datos de la variable.
- Existe un tipo de “variable”, denominada **constante**, la cual se utiliza para definir valores fijos, que no requieran ser modificados.

```
mi_variable = 20  
print (mi_variable)
```

```
mi_variable = 'Aprendiendo Python'  
print (mi_variable)
```

```
# Esto es un comentario de una sola línea
```

```
"""Y este es un comentario  
de varias líneas"""
```

Tipos de datos básicos

Cadena de texto (string):	<pre>mi_cadena = "Hola Mundo!" mi_cadena_multilinea = "" Esta es una cadena de varias lineas ""</pre>
---------------------------	-----------------------------------------------------------------------------------------------------------------------

Número entero:	<pre>edad = 35</pre>
----------------	----------------------

Número entero octal:	<pre>edad = 043</pre>
----------------------	-----------------------

Número entero hexadecimal:	<pre>edad = 0x23</pre>
----------------------------	------------------------

Número real:	<pre>precio = 7435.28</pre>
--------------	-----------------------------

Booleano (verdadero / Falso):	<pre>verdadero = True falso = False</pre>
-------------------------------	-----------------------------------------------

- Num1 = 10
- Num2 = 0x10
- Num3 = hex(num1)
- Num4 = int (num2)
- 10/3

Num1 : 10
Num2 : 16
Num3 : 0xa
Num4 : 16

Devuelve que tipo es:

```
type( a )
```

Operaciones con cadenas

Funcionalidad	Código
Convertir una variable a string	<code>str(variable)</code>
Unir cadenas	<code>Cadena 1= 'Hola'</code> <code>Cadena2 = 'amigo'</code> <code>Cadena3 = cadena1 + cadena2</code>
Conocer si un string contiene a otro	<code>Cadena1 = 'Peruanisimo'</code> <code>Cadena2= 'si'</code> <code>contenida = cadena2 in cadena1</code> <code>print contenida</code>
Longitud de una cadena	<code>Longitud = len (cadena1)</code>
Repetir una cadena	<code>cadena= 'mio'</code> <code>print (cadena * 3)</code>
Extraer subcadena	<code>Cadena = 'Arequipa'</code> <code>subcadena = cadena [1:3]</code>

Caracteres

```
cadena='abcdefg'
caracter=cadena[0]
print caracter ,
caracter=cadena[1]
print caracter ,
caracter=cadena[-1]
print caracter ,

caracter=cadena[0]
numero=ord(caracter)

numero=int(cadena)
caracter=chr(numero)
```

Las posiciones en la cadena inician en CERO

Un índice negativo se lee del final al principio

Operadores relacionales y lógicos

Símbolo	Significado	Ejemplo	Resultado
<code>==</code>	Igual que	<code>5 == 7</code>	Falso
<code>!=</code>	Distinto que	<code>rojo != verde</code>	Verdadero
<code><</code>	Menor que	<code>8 < 12</code>	Verdadero
<code>></code>	Mayor que	<code>12 > 7</code>	Falso
<code><=</code>	Menor o igual que	<code>12 <= 12</code>	Verdadero
<code>>=</code>	Mayor o igual que	<code>4 >= 5</code>	Falso

Operador	Ejemplo	Resultado*
and (y)	<code>5 == 7 and 7 < 12</code>	<code>0 y 0</code> Falso
	<code>9 < 12 and 12 > 7</code>	<code>1 y 1</code> Verdadero
	<code>9 < 12 and 12 > 15</code>	<code>1 y 0</code> Falso
or (o)	<code>12 == 12 or 15 < 7</code>	<code>1 o 0</code> Verdadero
	<code>7 > 5 or 9 < 12</code>	<code>1 o 1</code> Verdadero
xor (o excluyente)	<code>4 == 4 xor 9 > 3</code>	<code>1 o 1</code> Falso
	<code>4 == 4 xor 9 < 3</code>	<code>1 o 0</code> Verdadero

Operadores aritméticos

Símbolo	Significado	Ejemplo	Resultado
+	Suma	$a = 10 + 5$	a es 15
-	Resta	$a = 12 - 7$	a es 5
-	Negación	$a = -5$	a es -5
*	Multiplicación	$a = 7 * 5$	a es 35
**	Exponente	$a = 2 ** 3$	a es 8
/	División	$a = 12.5 / 2$	a es 6.25
//	División entera	$a = 12.5 // 2$	a es 6.0
%	Modulo	$a = 27 \% 4$	a es 3

```
monto_bruto = 175
tasa_interes = 12
monto_interes = monto_bruto * tasa_interes / 100
tasa_bonificacion = 5
importe_bonificacion = monto_bruto * tasa_bonificacion / 100
monto_netto = (monto_bruto - importe_bonificacion) + monto_interes
```


Tipos de datos complejos

DICCIONARIO

Los diccionarios permiten utilizar una clave para declarar y acceder a un valor

Definición

```
diccionario = {'clave_1': 'valor_1', 'clave_2': 'valor_2', 'clave_7': 'valor_7'}
```

Obtener uno de los valores

```
print (diccionario['clave_2'])
```

Modificar uno de los valores

```
diccionario['clave_1'] = 'otro Valor'
```

Agregar nuevo valor

```
diccionario['clave_8'] = 'nuevo valor'
```

Quitar un valor

```
del (diccionario['clave_2'])
```

Estructura de control

Condicional:	if semaforo == verde:
	print ("Cruzar la calle")
if	else:
	print ("Esperar")

Iterativo:	anio = 2001
Bucle while	while anio <= 2012:
	print ("Informes del Ano", str(anio))
	anio += 1

Iterativo:	mi_lista = ['Juan', 'Antonio', 'Pedro', 'Herminio']	mi_tupla = ('rosa', 'verde', 'celeste', 'amarillo')
Bucle for	for nombre in mi_lista:	for color in mi_tupla:
	print (nombre)	print (color)

Funciones

```
def función_de_ejemplo  
(parámetro1, parámetro2=24):  
    código  
    código  
    ...
```

Parámetros o argumentos son valores que la función espera recibir cuando sea invocada.

- Pueden ser ninguno, uno o más parámetros.
- Es opcional darle un valor por defecto.

Al llamar a una función, siempre se le deben pasar sus argumentos en el mismo orden en el que los espera, excepto si se pasa los argumentos como keywords.

```
def saludar (nombre, mensaje ='Hola'):  
    print (mensaje,nombre)  
saludar(mensaje="Buen día", nombre="Juancho")
```

Los parámetros pueden estar contenidos en:

- Una lista o tupla
- Diccionario

```
def calcular(importe, descuento):  
    return importe - (importe * descuento / 100)  
datos = [1500, 10]  
print (calcular(*datos))
```

```
def calcular(importe, descuento):  
    return importe - (importe * descuento / 100)  
datos = {"descuento": 10, "importe": 1500}  
print (calcular(**datos))
```

Uso de librerías

- Son archivos que contienen conjunto de funciones.



- Para usarlas hay que importarlas



- Numerical Python, proporciona la vectorización de operaciones matemáticas en el tipo de matrices, mejorando el rendimiento y, en consecuencia, acelera la ejecución.
- Esta orientada en administrar y tratar los datos como matrices.
- `ndarray`, es un *n-dimensional array* que contiene valores del mismo tipo.
- Son indexados por enteros no negativos.

```
import numpy as np
a=np.array([1,2,3])
type(a)
```

`numpy.ndarray`

```
a.shape
a[0],a[2]
a[1]=5
```

- Array con más de una dimensión:

```
b=np.array([[2,7,9],[5,1,3]])
b.shape
b.size
```

- Crear un numpy array

```
np.arange(7)
np.random.random((3,3))
np.ones((2,3))
np.zeros((1,2))
np.eye(3)
np.full((3,2),7)
np.linspace(1,2,4)
```

```
aleatorios = np.random.random(20)
aleatorios[7:10]
```

Funciones

```
a=np.array([[1,2,3],[4,5,6]])
```

```
b=np.array([[7,8,9],[10,11,12]])
```

```
np.add(a,b) #a+b hace lo mismo
```

```
np.subtract(a,b) # lo mismo que a-b
```

```
np.multiply(a,b) #a*b funciona igual
```

```
np.divide(a,b) #lo mismo que a/b
```

```
np.sqrt(a)
```

```
a=np.array([[8, .2],[13,4.2]])
```

```
np.sum(a)
```

```
np.sum(a,axis=0) #suma de cada columna
```

```
#Convertir de float a entero con astype:
```

```
a = np.array([5.7, 4.2, 6.0])
```

```
b = a.astype(np.int32)
```

```
print(b)
```

```
#Tratar de convertir array de string a un tipo de array de enteros:
```

```
a = np.array(["23", "12", "nan", "32", "10"])
```

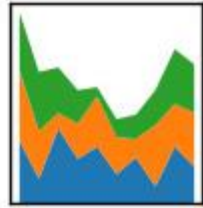
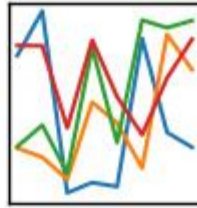
```
b = a.astype(np.int32) # ¿se puede?
```

```
# b=a.astype(np.float)
```

```
np.nansum(b)
```

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- Pandas provee data estructurada como series, dataframes y paneles que ayudan en la manipulación de data sets y series temporales.

Diagram illustrating a DataFrame structure with columns and rows.

Columns: Name, Team, Number, Position, Age

Rows: 0, 1, 2, 3, 4, 5, 6

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Data: A pink box highlights the data for Jonas Jerebko (Row 2), including his Name, Team, Number, Position, and Age.

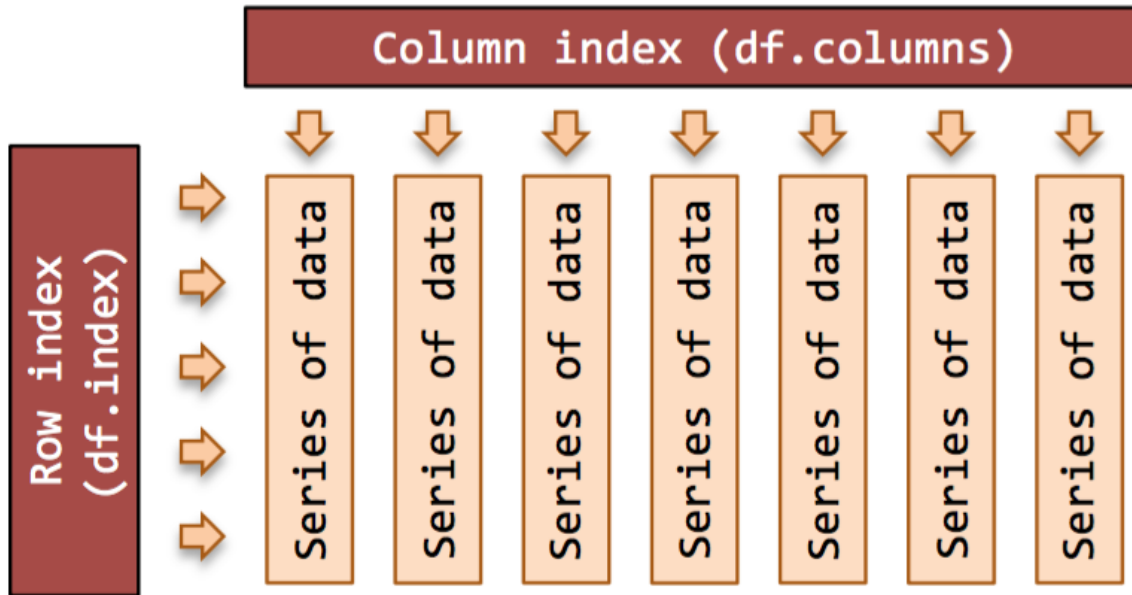
Diagram illustrating a DataFrame structure with columns and rows.

Columns: Open, High, Low, Close, Adj Close, Volume

Rows: 2013-12-10, 2013-12-11, 2013-12-12, 2013-12-13, 2013-12-16

Date	Open	High	Low	Close	Adj Close	Volume
2013-12-10	80.511429	81.125717	80.171425	80.792854	68.761612	69567400
2013-12-11	81.000000	81.567146	79.955711	80.194283	68.252151	89929700
2013-12-12	80.305717	80.762856	80.004288	80.077141	68.152473	65572500
2013-12-13	80.407143	80.411430	79.095711	79.204285	67.409607	83205500
2013-12-16	79.288574	80.377144	79.287140	79.642860	67.782852	70648200

Pandas Dataframe



```
[1]: import pandas as pd
      srs_a = pd.Series([10,30,60,80,90])
      srs_b = pd.Series(['red', 'green', 'blue', 'white', 'black'])
      df = pd.DataFrame({'a': srs_a, 'b': srs_b})
      df
```

	a	b
0	10	red
1	30	green
2	60	blue
3	80	white
4	90	black

Usando Dataframe Pandas

```
# importar la librería pandas como pd
```

```
import pandas as pd
```

```
# lista de strings
```

```
lst = ['Math', 'Geo', 'Stat', 'Lan', 'En', 'Science']
```

```
# Llamando al constructor DataFrame
```

```
df = pd.DataFrame(lst)
```

```
print(df)
```

```
# Creando dataframe desde un diccionario
```

```
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],  
        'Age':[27, 24, 22, 32],  
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],  
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
```

```
df = pd.DataFrame(data)
```

```
# Seleccionar 2 columnas
```

```
print(df[['Name', 'Qualification']])
```

```
# Revisando el dataframe
```

```
df.tail(2)
```

```
df['Age']+=1
```

```
df.head(3)
```

```
df.describe()
```

```
#Adicionando una columna
```

```
df['complains'] = np.zeros(df.index.size)
```

```
df.head(3)
```

```
# Indexing
```

```
df[1:3][['Name', 'Age']]
```

```
df.loc[:, 'Address']
```

```
df['Address']
```

```
df.iloc[:,2]
```

```
df.loc[0:2, ['Name', 'Age']]
```

Importando dataset

- Importar Pandas

```
import pandas as pd
```

- Importando un dataset

```
furniture=pd.read_csv('furniture.csv')
```

```
furniture
```

```
furniture.columns # Muestra el nombre de las columnas
```

```
furniture.columns[0:2] #slice
```

```
furniture.dtypes
```

```
furniture['Brand'].dtypes
```

```
furniture.shape
```



Es una popular librería para el trazado 2D científico en Python. Permite guardar gráficos en diferentes formatos de imagen de trama y vector, incluidos PDF, JPG, SVG, PNG, BMP y GIF.

#Usando una distribución normal

```
otros = np.random.randn(100)
```

#Importar la librería

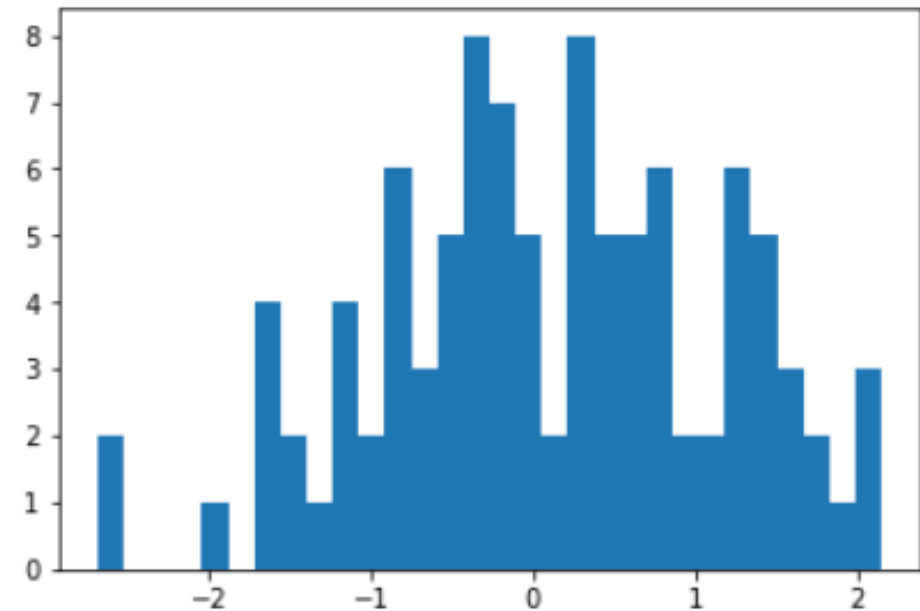
```
import matplotlib.pyplot as plt
```

Indicar que use the GUI interna

```
%matplotlib inline
```

```
plt.hist(otros, bins=30)
```

```
(array([2., 0., 0., 0., 1., 0., 4., 2., 1., 4., 2., 6., 3., 5., 8., 7., 5.,  
       2., 8., 5., 5., 6., 2., 2., 6., 5., 3., 2., 1., 3.]),  
 array([-2.68260955, -2.52154094, -2.36047233, -2.19940372, -2.03833511,  
        -1.8772665 , -1.71619789, -1.55512928, -1.39406067, -1.23299206,  
        -1.07192345, -0.91085484, -0.74978623, -0.58871762, -0.42764901,  
        -0.2665804 , -0.10551179,  0.05555682,  0.21662543,  0.37769404,  
         0.53876265,  0.69983126,  0.86089987,  1.02196848,  1.18303709,  
         1.3441057 ,  1.50517431,  1.66624292,  1.82731153,  1.98838014,  
         2.14944875])),  
<a list of 30 Patch objects>)
```



Trazado y atributos

```
import matplotlib.pyplot as plt
import numpy as np

# Get some random points from a gamma dist.
L = np.random.gamma(0.2, 0.2, 100)

%matplotlib inline

# Plot and change the line using a format string:
#line, = plt.plot(L, "--") # Note the comma

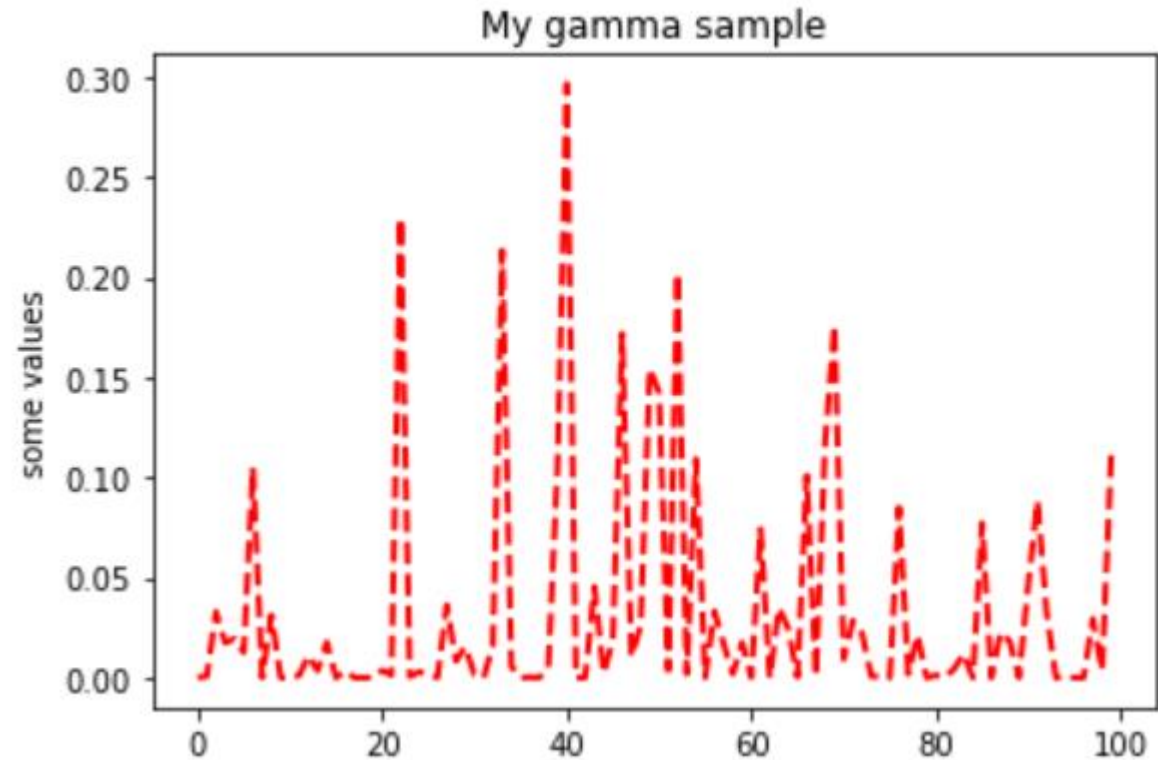
# This is the same as:
line, = plt.plot(L, linestyle="--")

# Method 2: Use setp and pass the object:
plt.setp(line, color="r")

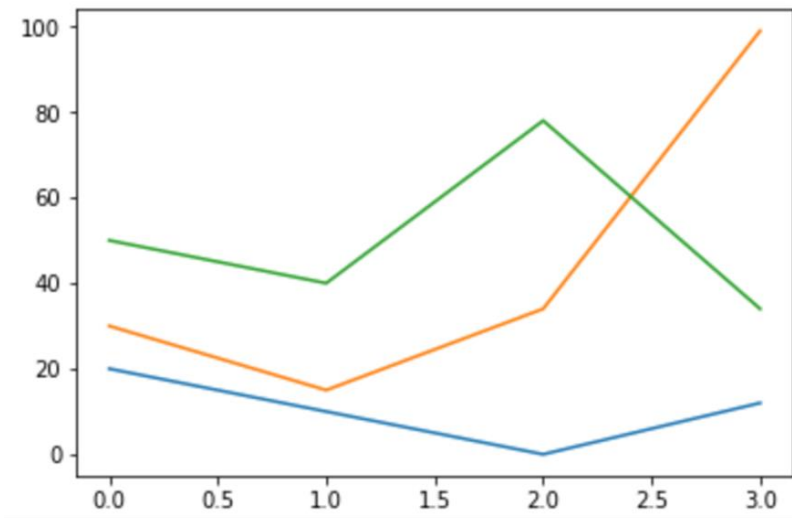
# Method 3: Call directly the set_XX on the object:
line.set_linewidth(2.0)

# Change figure-related properties:
plt.title("My gamma sample")
plt.ylabel("some values")
```

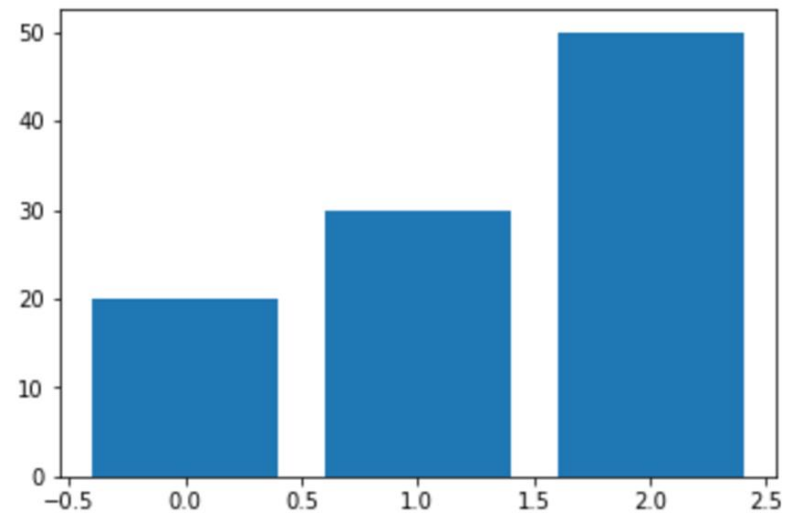
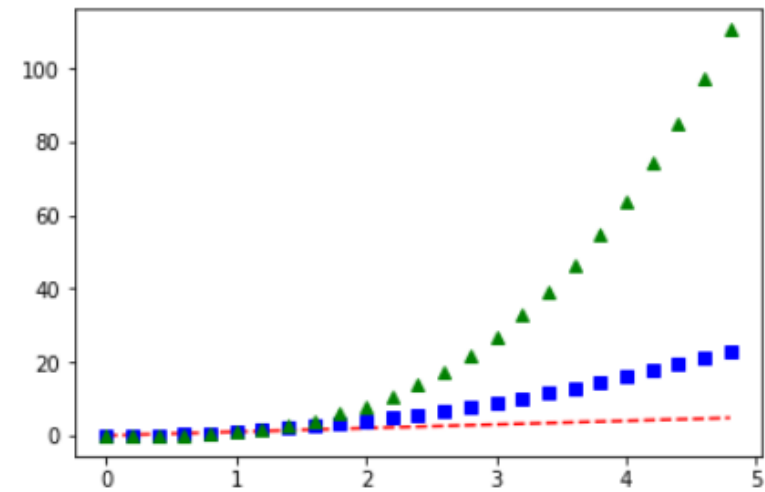
```
Text(0, 0.5, 'some values')
```



Más gráficas



3



#Guardando gráfico

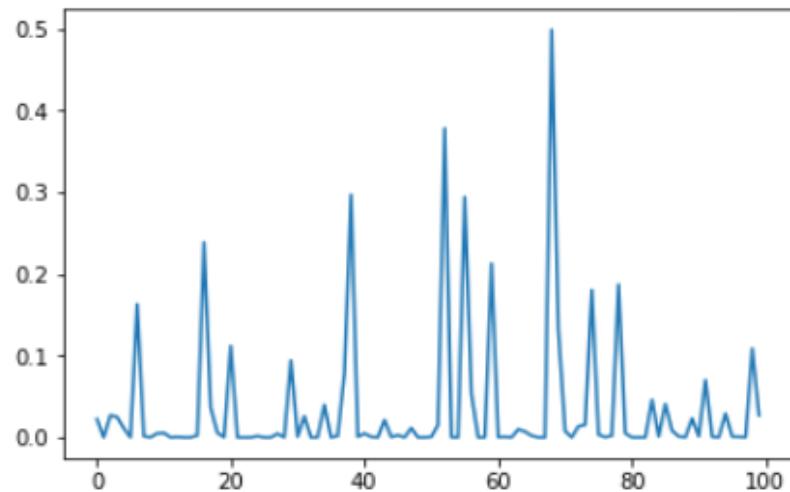
```
L = np.random.gamma(0.2, 0.2, 100)
```

```
plt.plot(L)
```

```
plt.savefig("gamma.pdf", format="pdf")
```

```
%ls -la
```

```
-rw-r--r-- 1 jupyter jupyter 2631 Jan 27 23:06 Arbol.ipynb
-rw-r--r-- 1 jupyter jupyter 7476 Jan 28 17:44 gamma.pdf
drwxr-xr-x 2 jupyter jupyter 4096 Jan 27 23:05 .ipynb_checkpoints/
-rw-r--r-- 1 jupyter jupyter 37733 Jan 28 17:43 Untitled.ipynb
```



#Adicionar texto

```
plt.plot(L)
```

```
t = plt.text(15, 0.5, "random!") # Se da la coordenada
```

```
t.set_fontsize(20)
```

```
t2 = plt.text(60, 0.35, r"$\mu=0.2$")
```

