

MODULO EN PYTHON INTRODUCCION

POR: Guillermo Andres De Mendoza Corrales



Temario

Inducción al lenguaje

Ambiente de ejecución

MarkDown

Comentarios

Instanciación de variables

Operadores numéricos

Conversión de variables

Interacción de salida con la consola

Interacción de entrada con la consola

Clase String

Librería Delay

Librería Math

Librería Random

Librería Time

Condicionales y operaciones booleanas

Ciclos for-while

Continue-Break

Ciclo mejorado for-each

Objeto None

Listas 1d – nd

Numpy

Sets

Mapas

Funciones

Excepciones

Graficas



Comentarios

```
In [6]: # Este es un comentario
```

```
In [7]: """
Este es un comentario
Este tambien
Y este tambien
"""
```

<https://markdown-it.github.io/>



Instanciación de variables =

```
In [2]: v1 = "Hola"  
        v2 = "Mundo"  
        r = v1 + " " + v2  
        print(r)
```

Hola Mundo

```
In [4]: n1 = 3  
        n2 = 2  
        r = n1 + n2  
        print(r)
```

5

```
In [3]: v1 = True  
        v2 = False  
        print(v1 or v2)
```

True

```
In [5]: n1 = 3.1  
        n2 = 2.5  
        r = n1 + n2  
        print(r)
```

5.6



Operaciones numéricas

```
n1 = 10
n2 = 3

Suma print( n1 + n2 ) → 13
Resta print( n1 - n2 ) → 7
Multiplicación print( n1 * n2 ) → 30
División print( n1 / n2 ) → 3.333333333333335
División entera print( int(n1 / n2) ) → 3
Residuo div entera print( n1 % n2 ) → 1
*Potencia print( n1 ** n2 ) → 1000
```



Conversión de variables

String a Numero

```
numeroString = "3"  
numeroEntero = int(numeroString)  
print(numeroEntero)  
print(type(numeroEntero))
```

```
3  
<class 'int'>
```

Numero a string

```
numeroEntero = 3  
numeroString = str(numeroEntero)  
print(numeroString)  
print(type(numeroString))
```

```
3  
<class 'str'>
```



Conversión de variables

Flotante a Entero

```
numeroFlotante = 3.111111
numeroEntero = int(numeroFlotante)
print(numeroEntero)
print(type(numeroEntero))
```

```
3
<class 'int'>
```

Entero a Flotante

```
numeroEntero = 3
numeroFlotante = float(numeroEntero)
print(numeroFlotante)
print(type(numeroFlotante))
```

```
3.0
<class 'float'>
```



Interacción de salida con la consola

```
v1 = "Hola"  
v2 = "Mundo"  
print("%s %s"%(v1,v2))
```

Hola Mundo

```
v1 = 3.1415  
print("Numero: %.2f"%(v1))
```

Numero: 3.14

%s -> String, Boolean

%d -> Decimal

%f -> Flotante

%.nf -> n: numero decimales



Interacción de entrada con la consola

```
consoleInput = input("Ingrese su nombre")
print("Su nombre es: %s"%(consoleInput))
```

Ingrese su nombre



```
consoleInput = input("Ingrese su nombre")
print("Su nombre es: %s"%(consoleInput))
```

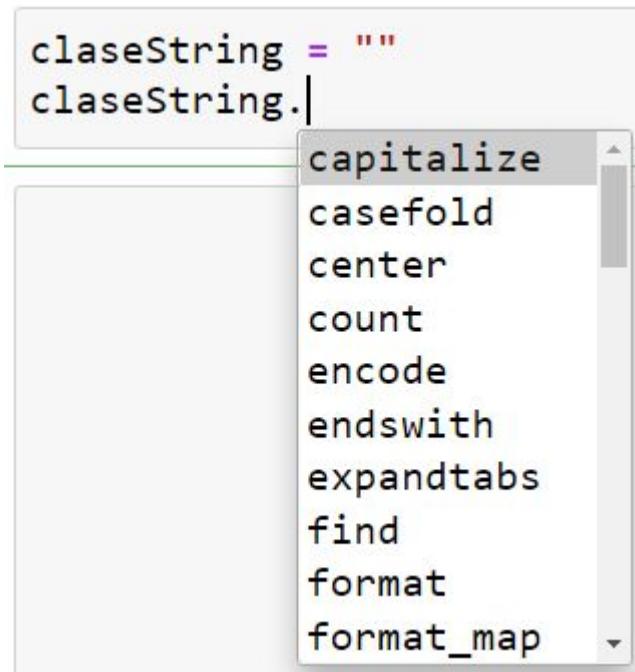
Ingrese su nombreGuillermo
Su nombre es: Guillermo

La entrada siempre
es un string



Clase string

Una variable String pertenece a la clase String, por lo tanto tendrá múltiples métodos, y sus caracteres son obtenidos por medio de posiciones



Clase string

Obtener un carácter

```
claseString = "Hola Mundo"  
print(claseString[3])
```

a

Partir un String

```
claseString = "Hola Mundo"  
print(claseString[3:7])
```

a Mu

Tamaño

```
claseString = "Hola Mundo"  
print(len(claseString))
```

10

Eliminar espacios antes y despues

```
a = " Hello, World! "  
print(a.strip()) # returns "Hello, World!"
```

Hello, World!

Minúsculas

```
a = "Hello, World!"  
print(a.lower())
```

hello, world!

Reemplazar caracteres

```
a = "Hello, World!"  
print(a.replace("H", "J"))
```

Jello, World!

Mayúsculas

```
a = "Hello, World!"  
print(a.upper())
```

HELLO, WORLD!

Separar string por carácter

```
a = "Hello, World!"  
print(a.split(","))
```

['Hello', ' World!']



Delay

```
import time

print("A")
time.sleep(1)
print("B")
time.sleep(1)
print("C")
time.sleep(1)
```

A
B
C

El programa espera
un tiempo en
segundos



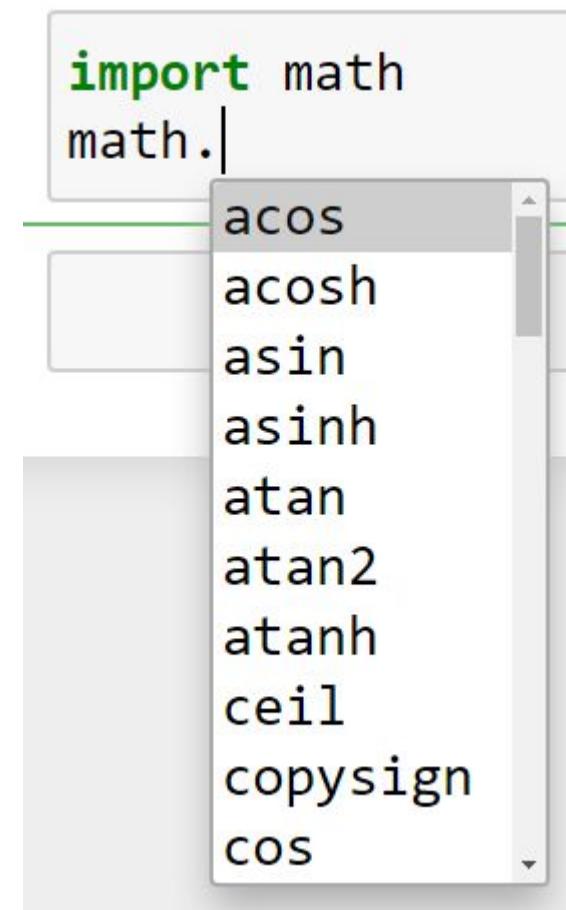
Math

```
import math  
print(math.pi)
```

3.141592653589793

<https://docs.python.org/3/library/math.html>

Para obtener los métodos de una librería teclear TAB después del punto



Random

```
import random

#numero flotante del 0 al 1 -> [0,1)
r1 = random.random()

#numero entero del n1 al n2 -> [n1,n2]
r2 = random.randint(1,6)

print("Numero [0,1): %f"%(r1))
print("Numero [1,6]: %d"%(r2))
```

Numero [0,1): 0.172517

Numero [1,6]: 5

<https://docs.python.org/3/library/random.html>



Time

```
import datetime
d1 = datetime.datetime.now()
d2 = datetime.datetime(2009, 1, 6, 15, 8, 24, 78915)

print(d1)
print(d2)
```

Año – Mes – Dia – Hora – Minuto – Segundo - MicroSegundos

2020-10-02 20:28:08.426115

2009-01-06 15:08:24.078915

<https://docs.python.org/3/library/datetime.html>



Operaciones Booleanas

OR

```
v1 = True  
v2 = False  
  
print(v1 or v2)
```

True

AND

```
v1 = True  
v2 = False  
  
print(v1 and v2)
```

False

NOT

```
v1 = True  
print(not v1)
```

False

A	B	A AND B	A OR B	NOT A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False



Operación de igualdad ==

```
nombre1 = "Carlos"  
nombre2 = "Guillermo"  
  
print(nombre1 == nombre2)
```

False

```
nombre1 = "Guillermo"  
nombre2 = "Guillermo"  
  
print(nombre1 == nombre2)
```

True



Condicionales booleanos

<code>print(9 < 10)</code>	True
<code>print(10 < 10)</code>	False
<code>print(11 < 10)</code>	False

<code>print(9 <= 10)</code>	True
<code>print(10 <= 10)</code>	True
<code>print(11 <= 10)</code>	False

<code>print(9 > 10)</code>	False
<code>print(10 > 10)</code>	False
<code>print(11 > 10)</code>	True

<code>print(9 >= 10)</code>	False
<code>print(10 >= 10)</code>	True
<code>print(11 >= 10)</code>	True



Condicional IF

Solo puede entrar a una condición !

```
if( OPERADOR BOOLEANO ):  
    — ACCIONES CON SANGRIA SI EL BOOL ES TRUE
```

```
if( OPERADOR BOOLEANO ):  
    — ACCIONES CON SANGRIA SI EL BOOL ES TRUE  
else:  
    — ACCIONES CON SANGRIA SI EL BOOL ES FALSE
```



Condicional IF

Solo puede entrar a una condición !

```
if( OPERADOR BOOLEANO 1 ):  
    — ACCIONES CON SANGRIA SI EL BOOL1 ES TRUE  
elif( OPERADOR BOOLEANO 2 ):  
    — ACCIONES CON SANGRIA SI EL BOOL1 ES FALSE y BOOL2 es TRUE  
else:  
    — ACCIONES CON SANGRIA SI EL BOOL1 ES FALSE y BOOL2 es FALSE
```



Condicional ELIF

Solo puede entrar a una condición !

```
edadPersona = 17

if(edadPersona>18):
    print("Es mayor de edad")
elif(edadPersona==18):
    print("Tiene exactamente 18")
else:
    print("Es menor de edad")
```



Condición ternaria

on_true if expression else on_false

```
edad = 22
indicador = "Es mayor de edad" if edad >= 18 else "Es menor de edad"
print(indicador)
```

'Es mayor de edad'



Ciclos

FOR

*Se exactamente cuantas ejecuciones de deben realizar

```
for nombreVariable in range(nInicio, nFinal):
```

— ACCIONES

WHILE

*No se cuantas veces se va a ejecutar algo

```
while condicionBooleana:
```

— ACCIONES



Ciclos

FOR

```
for numero in range(0,10):  
    print(numero)
```

0
1
2
3
4
5
6
7
8
9

*Se exactamente cuantas ejecuciones de deben realizar

*El rango no incluye el ultimo numero

WHILE

```
contador = 0  
while contador < 10:  
    print(contador)  
    contador = contador + 1
```

0
1
2
3
4
5
6
7
8
9

*No se cuantas veces se va a ejecutar algo



Ciclos

break: termina la ejecución del ciclo

Continue: salta a la siguiente iteración del ciclo

```
for numero in range(0,10):
    if 0 == numero%2:
        continue
    print(numero)
```

1
3
5
7
9

```
for numero in range(0,10):
    if numero > 5:
        break
    print(numero)
```

0
1
2
3
4
5



Ciclo Mejorado - ForEach

For normal:

```
lista = [0,1,2,3,4,5,6,7,8,9,10]
```

```
#for normalito de toda la vida
for indice in range(0,10):
    print(lista[indice])
```

```
0
1
2
3
4
5
6
7
8
9
```

For mejorado:

```
lista = [0,1,2,3,4,5,6,7,8,9,10]
```

```
#for mejorado - for each
for numero in lista:
    print(numero)
```

```
0
1
2
3
4
5
6
7
8
9
10
```



None

None significa que la variable no apunta a ninguna espacio en memoria (null)

```
variable = None
```

```
variable = None

if( variable is None ):
    print("variable is null")
else:
    print("variable have a value")
```

variable is null

```
variable = "hola"

if( variable is None ):
    print("variable is null")
else:
    print("variable have a value")
```

variable have a value



Listas – crear - agregar

Estructura que almacena datos de forma dinámica

```
#Creamos una Lista vacia
personas = []

#Agregamos elementos a la Lista
personas.append("Guillermo")
personas.append("Mario")
personas.append("Luigi")
personas.append("Bowser")
personas.append("Joshi")

#print(primera)
print(personas)
```

```
['Guillermo', 'Mario', 'Luigi', 'Bowser', 'Joshi']
```



Listas – estructura - obtener

```
[ 'Guillermo', 'Mario', 'Luigi', 'Bowser', 'Joshi' ]
```

Posición:	0	1	2	3	4
Posición:	-5	-4	-3	-2	-1

Tamaño de la lista: 5

```
#obtenemos La cantidad de elementos de La Lista  
print(len(personas))
```

5

Obtener el elemento 3

```
#obtenemos el elemento 3 de la lista  
print(personas[3])
```

Bowser



Listas – insertar

```
#Creamos una lista vacia
personas = []

#Agregamos elementos a la lista
personas.append("Guillermo")
personas.append("Mario")
personas.append("Luigi")
personas.append("Bowser")
personas.append("Joshi")

#print( "Despues de eliminar: %s" %(personas))

#Insertamos la nueva persona en la posicion deseada
personas.insert(3, "Wario") #Indice , Elemento

#print( "Despues de eliminar: %s" %(personas))
```

```
Despues de eliminar: ['Guillermo', 'Mario', 'Luigi', 'Bowser', 'Joshi']
Despues de eliminar: ['Guillermo', 'Mario', 'Luigi', 'Wario', 'Bowser', 'Joshi']
```



Listas – eliminar – por objeto

```
#Creamos una lista vacia
personas = []

#agregamos elementos a la lista
personas.append("Guillermo")
personas.append("Mario")
personas.append("Luigi")
personas.append("Bowser")
personas.append("Joshi")

#printamos la lista antes de eliminar una persona
print("Despues de eliminar: %s"%(personas))

#eliminar un elemento de la lista indicando su valor
personas.remove("Bowser")

#printamos la lista despues de eliminar una persona
print("Despues de eliminar: %s"%(personas))
```

```
Despues de eliminar: ['Guillermo', 'Mario', 'Luigi', 'Bowser', 'Joshi']
Despues de eliminar: ['Guillermo', 'Mario', 'Luigi', 'Joshi']
```



Listas – eliminar – por posición

```
#Creamos una Lista vacia
personas = []

#Agregamos elementos a la Lista
personas.append("Guillermo")
personas.append("Mario")
personas.append("Luigi")
personas.append("Bowser")
personas.append("Joshi")

#Imprimimos la Lista antes de eliminar una persona
print("Despues de eliminar: %s"%(personas))

#Eliminar un elemento de la Lista indicando su posicion|
personas.pop(3)

#Imprimimos la Lista despues de eliminar una persona
print("Despues de eliminar: %s"%(personas))
```

```
Despues de eliminar: ['Guillermo', 'Mario', 'Luigi', 'Bowser', 'Joshi']
Despues de eliminar: ['Guillermo', 'Mario', 'Luigi', 'Joshi']
```



Listas – buscar un elemento

```
#Creamos una lista vacia
personas = []

#agregamos elementos a la lista
personas.append("Guillermo")
personas.append("Mario")
personas.append("Luigi")
personas.append("Bowser")
personas.append("Joshi")

#buscar elementos
indiceDeBowser = personas.index("Bowser")
print("Indice de bowser: %s"%(indiceDeBowser))

#buscar elementos no existentes
indiceDeBPeach = personas.index("Peach")
print("Indice de Peach: %s"%(indiceDeBPeach))
```

Indice de bowser: 3

ValueError

Traceback (most recent call last)

```
<ipython-input-78-49425966ddc3> in <module>
    14
    15 #buscar elementos no existentes
--> 16 indiceDeBPeach = personas.index("Peach")
    17 print("Indice de Peach: %s"%(indiceDeBPeach))
```

ValueError: 'Peach' is not in list



Listas – preguntar si se encuentra el elemento

```
myList = ["Guillermo", "Mario", "Luigi", "Peach", "Mario"]

estaContenidoMario = "Mario" in myList
print(estaContenidoMario)

estaContenidoMacman = "Pacman" in myList
print(estaContenidoMacman)
```

True

False



Listas – Matrices

Ejemplo matriz 3x3

```
matriz = [[1,2,3],[4,5,6],[7,8,9]]  
print(matriz)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Obtener elemento individual

```
print(matriz[1][1])
```

```
5
```

Imprimir todos los elementos

```
for fila in range(0,len(matriz)):  
    for columna in range(0,len(matriz[fila])):  
        print("[fila][columna] = %d" % (fila,columna,matriz[fila][columna]))
```

```
[0][0] = 1  
[0][1] = 2  
[0][2] = 3  
[1][0] = 4  
[1][1] = 5  
[1][2] = 6  
[2][0] = 7  
[2][1] = 8  
[2][2] = 9
```



Numpy

Crear matriz

```
import numpy as np  
  
matriz = np.array([[7,8,5],[3,5,7]])  
print(matriz)
```

```
[[7 8 5]  
 [3 5 7]]
```

Suma o resta de matrices

```
import numpy as np  
  
matriz1 = np.array([[1,2,3],[4,5,6]])  
matriz2 = np.array([[1,1,1],[1,1,1]])  
matrizSuma = matriz1 + matriz2  
print(matrizSuma)
```

```
[[2 3 4]  
 [5 6 7]]
```

Multiplicación de matrices

```
import numpy as np  
  
matriz1 = np.array([[1,2],[4,5]])  
matriz2 = np.array([[3,3],[1,1]])  
matrizMultiplicacion = matriz1 * matriz2  
print(matrizMultiplicacion)
```

```
[[3 6]  
 [4 5]]
```

Multiplicación por escalar

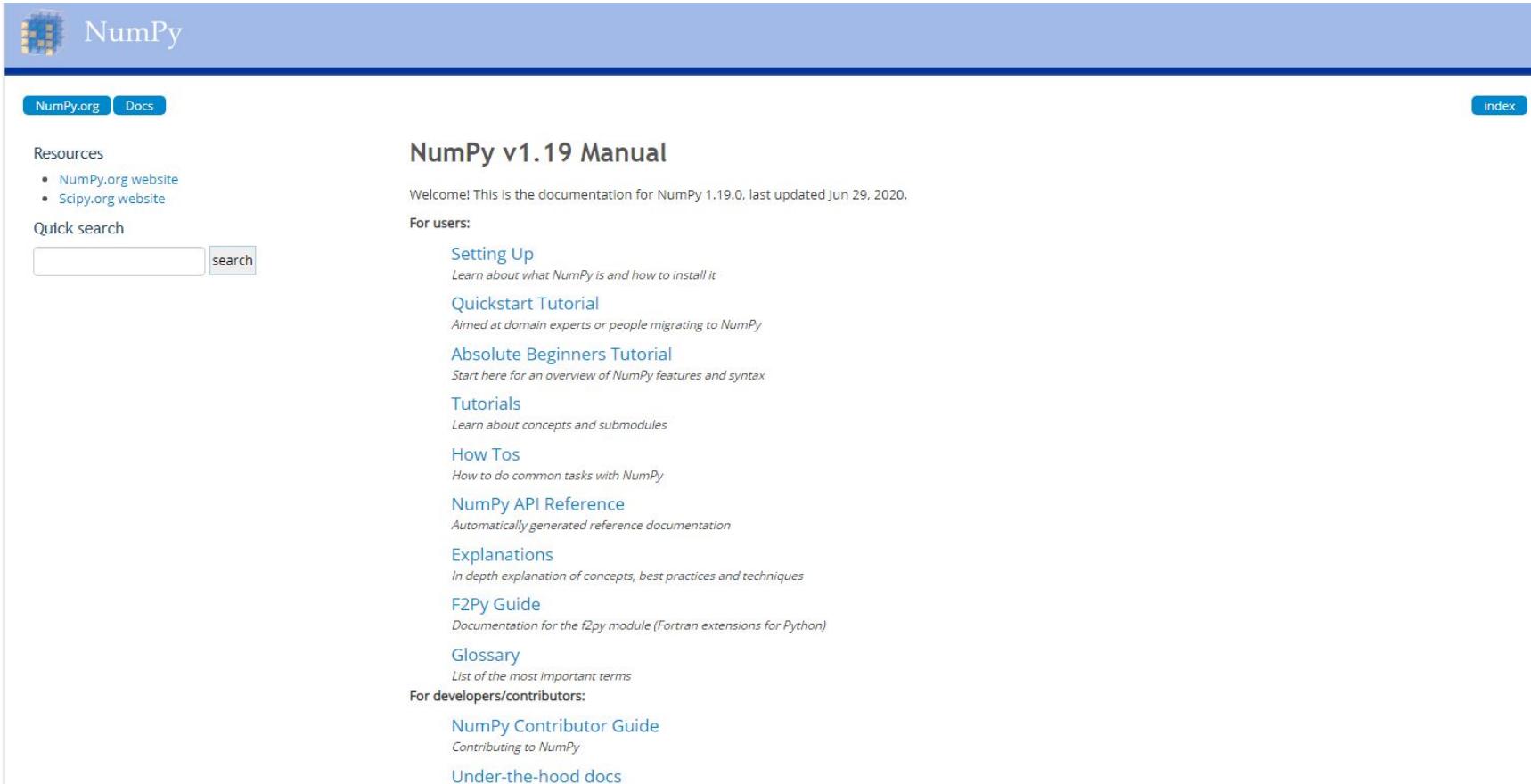
```
import numpy as np  
  
matriz = np.array([[7,8,5],[3,5,7]])  
matrizPorVector = 2*matriz  
print(matriz)
```

```
[[7 8 5]  
 [3 5 7]]
```



NumPy

<https://numpy.org/>



The screenshot shows the NumPy v1.19 Manual website. At the top, there's a blue header bar with the NumPy logo and the text "NumPy". Below the header, a navigation bar includes links for "NumPy.org" and "Docs" on the left, and "index" on the right. The main content area is titled "NumPy v1.19 Manual". It starts with a welcome message: "Welcome! This is the documentation for NumPy 1.19.0, last updated Jun 29, 2020." A section titled "For users:" lists several links: "Setting Up" (with a subtitle "Learn about what NumPy is and how to install it"), "Quickstart Tutorial" (with a subtitle "Aimed at domain experts or people migrating to NumPy"), "Absolute Beginners Tutorial" (with a subtitle "Start here for an overview of NumPy features and syntax"), "Tutorials" (with a subtitle "Learn about concepts and submodules"), "How Tos" (with a subtitle "How to do common tasks with NumPy"), "NumPy API Reference" (with a subtitle "Automatically generated reference documentation"), "Explanations" (with a subtitle "In depth explanation of concepts, best practices and techniques"), "F2Py Guide" (with a subtitle "Documentation for the f2py module (Fortran extensions for Python)"), and "Glossary" (with a subtitle "List of the most important terms"). Below these, a section titled "For developers/contributors:" contains links for "NumPy Contributor Guide" (with a subtitle "Contributing to NumPy") and "Under-the-hood docs". On the far left of the main content area, there's a sidebar with sections for "Resources" (links to NumPy.org and Scipy.org websites) and "Quick search" (with a search input field and a "search" button).



Sets

Estructura que almacena datos los cuales no pueden encontrarse repetidos

Estructura en Python = { e1, e2 , e3 ... eN }

SET

```
mySet = {"Guillermo", "Mario", "Luigi","Peach","Mario"}  
print(mySet)  
print("tamaño del set: %d"%(len(mySet)))  
  
{'Peach', 'Luigi', 'Mario', 'Guillermo'}  
tamaño del set: 4
```

LIST

```
myList = ["Guillermo", "Mario", "Luigi","Peach","Mario"]  
print(myList)  
print("tamaño del set: %d"%(len(myList)))  
  
['Guillermo', 'Mario', 'Luigi', 'Peach', 'Mario']  
tamaño del set: 5
```



Sets

Agregar elemento

```
thisset = {"apple", "banana", "cherry"}  
thisset.add("orange")  
print(thisset)
```

```
{'orange', 'banana', 'cherry', 'apple'}
```

Tamaño

```
thisset = {"apple", "banana", "cherry"}  
print(len(thisset))
```

3

Eliminar elemento

```
thisset = {"apple", "banana", "cherry"}  
thisset.remove("banana")  
print(thisset)
```

```
{'cherry', 'apple'}
```



List vs Sets

	Mutable	Ordered	Indexing / Slicing	Duplicate Elements
List	✓	✓	✓	✓
Tuple	✗	✓	✓	✓
Set	✓	✗	✗	✗



Mapas -> diccionario

```
myDiccionarioCodigoPais = {}
myDiccionarioCodigoPais["Colombia"] = 57
myDiccionarioCodigoPais["Usa"] = 1
myDiccionarioCodigoPais["Mexico"] = 52

#obtener llaves
print(myDiccionarioCodigoPais.keys()) → dict_keys(['Colombia', 'Usa', 'Mexico'])

#obtener un elemento
print(myDiccionarioCodigoPais["Colombia"]) → 57

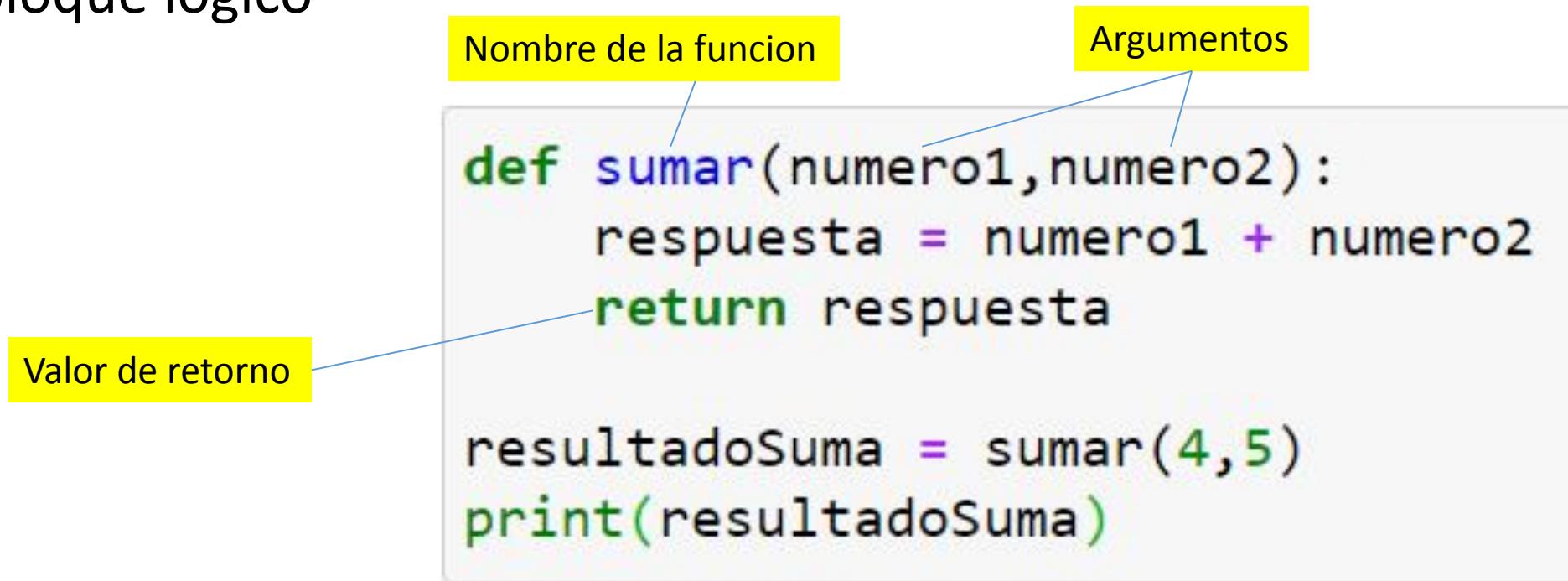
#actualizar un elemento
myDiccionarioCodigoPais["Colombia"] = 0
print(myDiccionarioCodigoPais["Colombia"]) → 0

#borrar elemento
myDiccionarioCodigoPais.pop("Colombia") → {'Usa': 1, 'Mexico': 52}
print(myDiccionarioCodigoPais)
```



Funciones

Una función es una estructura de código que empaqueta código en un bloque lógico



The diagram illustrates the structure of a Python function. It shows a code block with three annotations:

- Nombre de la función** (Name of the function) points to the word **def**.
- Argumentos** (Arguments) points to the parameters **numero1** and **numero2**.
- Valor de retorno** (Return value) points to the **return** statement.

```
def sumar(numero1,numero2):
    respuesta = numero1 + numero2
    return respuesta

resultadoSuma = sumar(4,5)
print(resultadoSuma)
```



Funciones – ejemplo 1

No es obligatorio tener argumentos o retornos

```
def decirHola():
    print("Hola")

decirHola()
```

Hola

Podría tener argumentos pero no retorno

```
def decirHola(persona):
    print("Hola %s"%(persona))

decirHola("Guillermo")
```

Hola Guillermo

O podría tener retorno pero no argumento

```
def obtenerSaludo():
    return "Hola "

saludo = obtenerSaludo() + "Guillermo"
print(saludo)
```

Hola Guillermo



Funciones – ejemplo 2

```
def contarCaracteresTexto(texto,caracter):
    contador = 0
    for indice in range(0,len(texto)):
        if(texto[indice]==caracter):
            contador = contador + 1
    return contador

resultado = contarCaracteresTexto("Este es un texto cualquiera","a")
print(resultado)
```

2

Este es un texto cualquiera



Funciones – ejemplo 3

No hay límite de la cantidad de argumento

```
def sumarCincoNumeros(n1,n2,n3,n4,n5):
    respuesta = n1 + n2 + n3 + n4 + n5
    return respuesta

respuestaSuma = sumarCincoNumeros(1,2,3,4,5)
print(respuestaSuma)
```

15

Llamar una función con más o menos argumentos de los que tiene definidos genera un **error (Exception)**

```
def sumarCincoNumeros(n1,n2,n3,n4,n5):
    respuesta = n1 + n2 + n3 + n4 + n5
    return respuesta

respuestaSuma = sumarCincoNumeros(1,2,3,4)
print(respuestaSuma)

-----
TypeError                                     Traceback (most recent call last)
<ipython-input-54-4568f8c385d9> in <module>
      3     return respuesta
      4
----> 5 respuestaSuma = sumarCincoNumeros(1,2,3,4)
      6 print(respuestaSuma)

TypeError: sumarCincoNumeros() missing 1 required positional argument: 'n5'
```



Funciones – parámetros por defecto

Los argumentos por defecto permiten que al llamar una función, estos sean opcionales, y siempre deben declararse como ultimo parámetro

Argumentos por defecto

```
def potenciaNumero(numero, potencia=2):  
    resultado = numero  
    for iteracion in range(1, potencia):  
        resultado = resultado * numero  
    return resultado
```

```
potenciaNumero(2,5)
```

32

```
def potenciaNumero(numero, potencia=2):  
    resultado = numero  
    for iteracion in range(1, potencia):  
        resultado = resultado * numero  
    return resultado
```

```
potenciaNumero(2)
```

4



Excepciones

Una excepción es cuando el programa genera un error al ejecutar alguna línea de código

División por 0

```
numero1 = 10
numero2 = 0
resultado = numero1/numero2
```

```
-----
ZeroDivisionError                                Traceback (most recent call last)
<ipython-input-60-a0f1fc4fd19c> in <module>
      1 numero1 = 10
      2 numero2 = 0
----> 3 resultado = numero1/numero2

ZeroDivisionError: division by zero
```

Elemento por fuera del rango

```
lista = ["Guillermo", "Andes"]
print(lista[99])
```

```
-----
IndexError                                     Traceback (most recent call last)
<ipython-input-61-a4ec13351e4f> in <module>
      1 lista = ["Guillermo", "Andes"]
----> 2 print(lista[99])

IndexError: list index out of range
```



Excepciones – try catch finally

Permite ejecutar código que puede fallar

try:

#codigo que puede fallar

except:

#codigo que se ejecutar al ejecutar error

finally:

#codigo que se ejecutar con o sin fallo



Excepciones – try catch finally

Una excepción es cuando el programa genera un error al ejecutar alguna línea de código

```
numero1 = 10
numero2 = 0
print("Antes de dividir")
resultado = numero1/numero2
print("Despues de dividir")
print("resultado: %f"%resultado)
```

Antes de dividir

```
-----
ZeroDivisionError                                T
<ipython-input-62-f6b2766942e0> in <module>
      2 numero2 = 0
      3 print("Antes de dividir")
----> 4 resultado = numero1/numero2
      5 print("Despues de dividir")
      6 print("resultado: %f"%resultado)
```

ZeroDivisionError: division by zero

```
numero1 = 10
numero2 = 0

print("Inicio del programa")

try:
    print("Dentro del try")
    print("Antes de dividir")
    resultado = numero1/numero2
    print("Despues de dividir")
except:
    print("ERROR CAPTURADO !")
finally:
    print("Dentro de finally")

print("Fin del programa")
```

Inicio del programa
Dentro del try
Antes de dividir
ERROR CAPTURADO !
Dentro de finally
Fin del programa



Graficas líneas

<https://matplotlib.org/contents.html>

```
import matplotlib.pyplot as plt

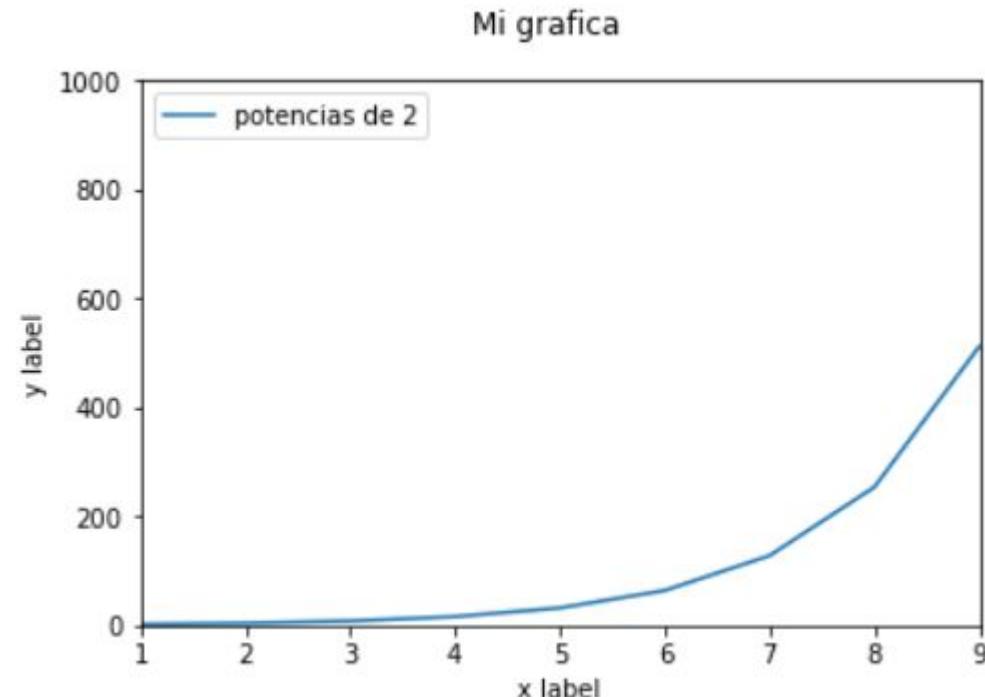
datosX = [1,2,3,4,5,6,7,8,9]
datosY = [2,4,8,16,32,64,128,256,512]

plt.suptitle('Mi grafica')
plt.ylabel('y label')
plt.xlabel('x label')
plt.legend("a")

plt.ylim(0, 1000)
plt.xlim(1, 9)

plt.plot(datosX,datosY,label="potencias de 2")
plt.legend(loc="upper left")

plt.show()
```



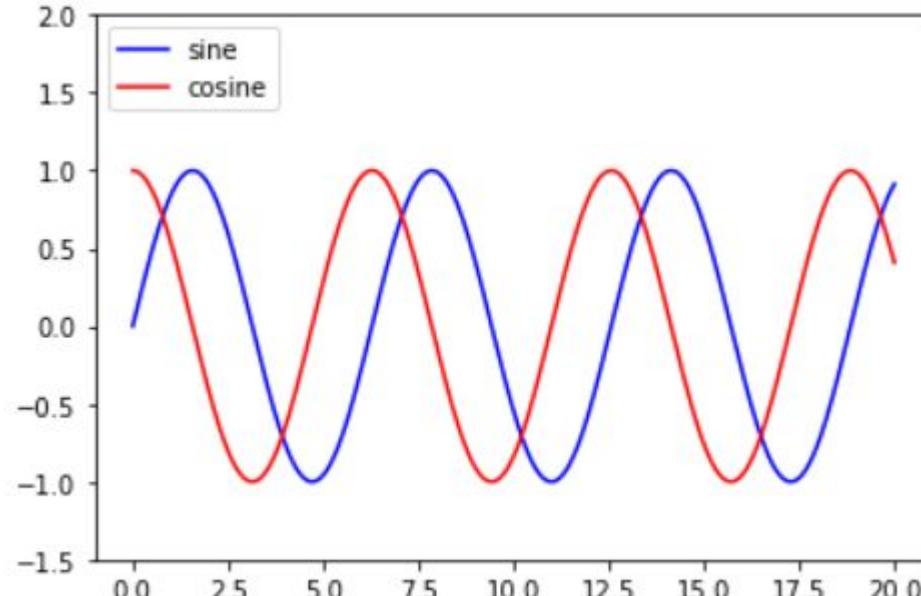
Graficas líneas

<https://matplotlib.org/contents.html>

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

x = np.linspace(0, 20, 1000)
y1 = np.sin(x)
y2 = np.cos(x)

plt.plot(x, y1, "-b", label="sine")
plt.plot(x, y2, "-r", label="cosine")
plt.legend(loc="upper left")
plt.ylim(-1.5, 2.0)
plt.show()
```



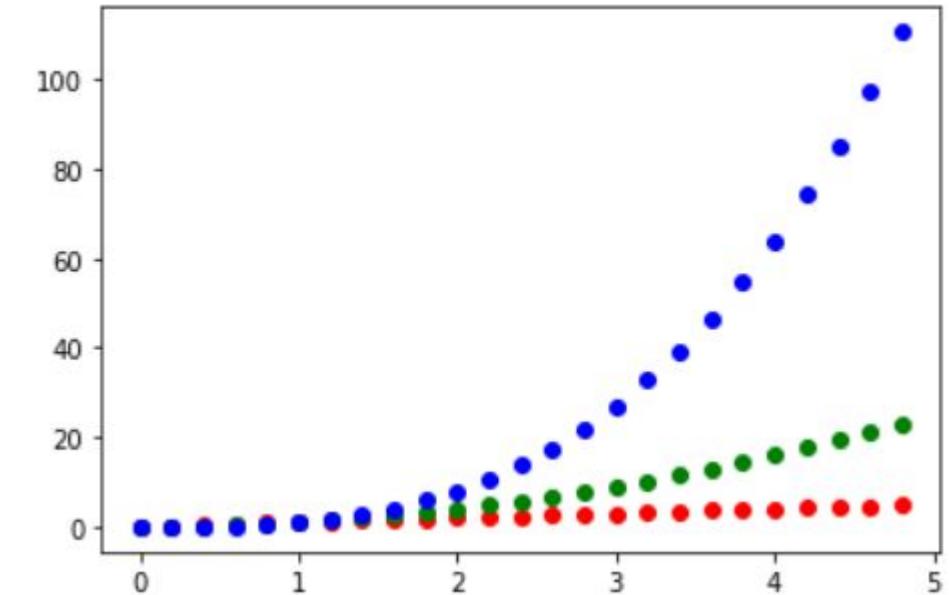
Graficas puntos

<https://matplotlib.org/contents.html>

```
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

plt.scatter(t, t,color='r')
plt.scatter(t, t**2,color='g')
plt.scatter(t, t**3,color='b')
plt.show()
```



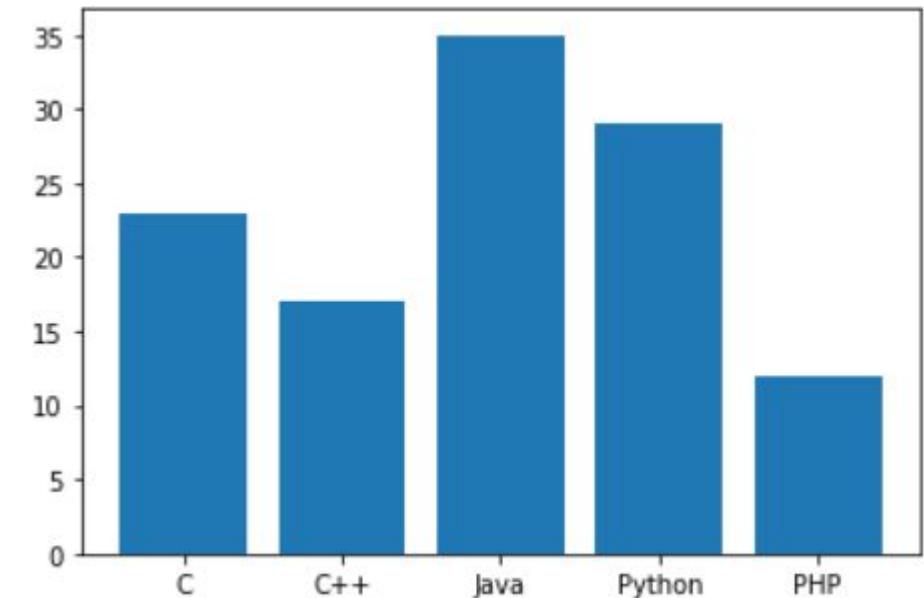
Graficas barras

<https://matplotlib.org/contents.html>

```
import matplotlib.pyplot as plt

dataX = ['C', 'C++', 'Java', 'Python', 'PHP']
dataY = [23, 17, 35, 29, 12]

plt.bar(dataX,dataY)
plt.show()
```



Graficas pastel

<https://matplotlib.org/contents.html>

```
import matplotlib.pyplot as plt

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')

plt.pie(sizes, explode, labels, autopct='%1.1f%%', shadow=True)
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

```
import matplotlib.pyplot as plt

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')

plt.pie(sizes, explode, labels, autopct='%1.1f%%', shadow=True)
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

