

MODULO EN PYTHON

POR: Guillermo Andres De Mendoza Corrales



Temario

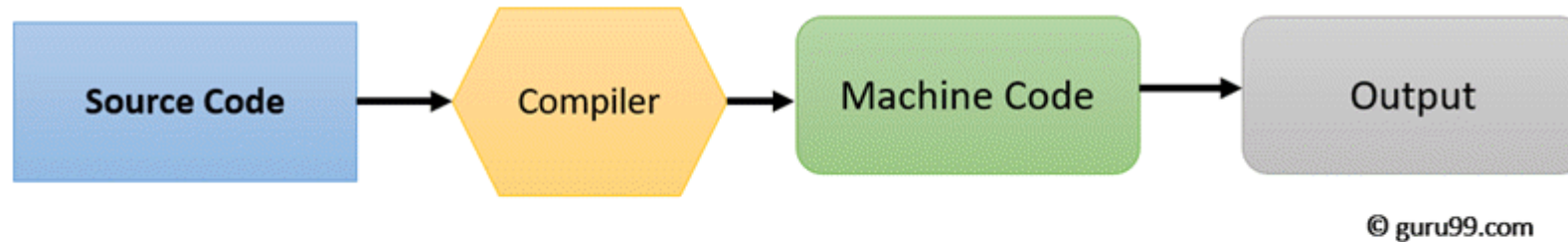
Inducción al lenguaje
Ambiente de ejecución
MarkDown
Comentarios
Instanciación de variables
Operadores numericos
Conversión de variables
Interacción de salida con la consola
Interacción de entrada con la consola
Clase String
Librería Math
Librería Random
Librería DateTime
Condicionales y operaciones booleanas

Ciclos for-while
Continue-Break
Ciclo mejorado for-each
Objeto None y garbage colector
Listas 1d – nd
Numpy basico
Sets
Mapas
Tuplas
Funciones
Excepciones



Python es interpretado, no compilado

How Compiler Works



How Interpreter Works



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

True

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

5

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

5.6



Operaciones numéricas

	<pre>n1 = 10 n2 = 3</pre>	
Suma	<pre>print(n1 + n2)</pre>	13
Resta	<pre>print(n1 - n2)</pre>	7
Multiplicación	<pre>print(n1 * n2)</pre>	30
División	<pre>print(n1 / n2)</pre>	3.3333333333333335
División entera	<pre>print(int (n1 / n2))</pre>	3
Residuo div entera	<pre>print(n1 % n2)</pre>	1
*Potencia	<pre>print(n1 ** n2)</pre>	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 - 1

```
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 salida con la consola - 2

```
v1 = "Hola"  
v2 = "Mundo"  
print( f"{v1} {v2}" )
```

Hola Mundo

Si queremos controlar los puntos decimales -> {:.2f}



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))
```

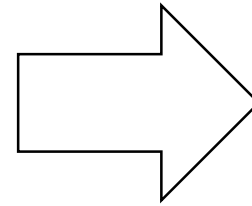
Ingrese su nombreGuillermo
Su nombre es: Guillermo

La entrada siempre
es un string



Python es un archivo .py

```
radio = float(input("Introduce el radio del círculo: "))  
area = math.pi * radio ** 2  
perimetro = 2 * math.pi * radio  
print(f"El área del círculo es: {area:.2f}")  
print(f"El perímetro del círculo es: {perimetro:.2f}")
```



Reto1.py

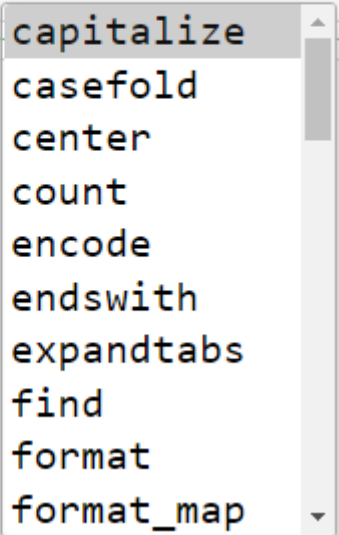
CMD:
python Reto1.py



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

```
classString = ""  
classString.|
```



A screenshot of a code editor showing a list of string methods. The code in the editor is `classString = ""` followed by `classString.|` on the next line. A dropdown menu is open, displaying a list of methods: `capitalize`, `casefold`, `center`, `count`, `encode`, `endswith`, `expandtabs`, `find`, `format`, and `format_map`. The `capitalize` method is currently selected and highlighted.



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!

Mayúsculas

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

HELLO, WORLD!

Reemplazar caracteres

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

Jello, World!

Separar string por carácter

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

['Hello', ' World!']



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

```
import math  
math.|
```

- acos
- acosh
- asin
- asinh
- atan
- atan2
- atanh
- ceil
- copysign
- cos



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>



DateTime

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

print(d1)
print(d2)
```

Año – Mes – Día – 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**

→
Una sangría es igual a 4 espacios



Condicional IF

Solo puede entrar a una condición !

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")

#imprimimos la lista
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")

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

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

#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', '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")

#imprimimos 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")

#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 – 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("[%d][%d] = %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/>

 NumPy

[NumPy.org](#) [Docs](#) [index](#)

Resources

- [NumPy.org website](#)
- [Scipy.org website](#)

Quick search

NumPy v1.19 Manual

Welcome! This is the documentation for NumPy 1.19.0, last updated Jun 29, 2020.

For users:

- [Setting Up](#)
Learn about what NumPy is and how to install it
- [Quickstart Tutorial](#)
Aimed at domain experts or people migrating to NumPy
- [Absolute Beginners Tutorial](#)
Start here for an overview of NumPy features and syntax
- [Tutorials](#)
Learn about concepts and submodules
- [How Tos](#)
How to do common tasks with NumPy
- [NumPy API Reference](#)
Automatically generated reference documentation
- [Explanations](#)
In depth explanation of concepts, best practices and techniques
- [F2Py Guide](#)
Documentation for the f2py module (Fortran extensions for Python)
- [Glossary](#)
List of the most important terms

For developers/contributors:

- [NumPy Contributor Guide](#)
Contributing to NumPy
- [Under-the-hood docs](#)



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'}



Tuple

```
dias_semana = ("Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo")
```

```
dias_semana[2]
```

```
'Miercoles'
```



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")  
print(myDiccionarioCodigoPais)
```

→ {'Usa': 1, 'Mexico': 52}



Funciones

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

Nombre de la función

Argumentos

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

Valor de retorno

```
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 cu**a**lquiera**a**



Funciones – ejemplo 3

No hay limite 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 mas 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



Excepciones – diferentes except

```
def manejar_excepciones():
    try:
        # Solicitar al usuario un número
        num1 = int(input("Introduce el primer número: "))

        # Solicitar al usuario otro número
        num2 = int(input("Introduce el segundo número: "))

        # Intentar dividir los números
        resultado = num1 / num2

    except ValueError:
        print("Error: Debes ingresar un número entero.")
    except ZeroDivisionError:
        print("Error: No se puede dividir entre cero.")
    except IndexError:
        print("Error: El índice que intentaste acceder no existe en la lista.")
    except Exception as e:
        # Esta es una excepción general para capturar cualquier error no esperado
        print(f"Se produjo un error inesperado: {e}")
    else:
        print(f"El resultado de la división es: {resultado}")
    finally:
        print("Fin del programa.")

# Ejecutar la función
manejar_excepciones()
```



Excepciones – lanzar excepción manual

python

```
raise Exception("Mensaje de error")
```

```
def pedir_numero_positivo():  
    numero = int(input("Introduce un número positivo: "))  
  
    if numero < 0:  
        raise ValueError("El número no puede ser negativo.")  
    else:  
        print(f"El número ingresado es: {numero}")
```

