

Clase3

October 11, 2025

```
[2]: # Variables / datos atómicos en python
entero = 15
flotante = 15.3
cadena = "texto"
booleano = True
```

```
[98]: type(booleano)
```

```
[98]: bool
```

```
[3]: #Más ejercicios con variables
suma = entero + flotante
print(suma)
print(cadena)
```

30.3

texto

```
[4]: # Lista en python

nombres = ["Noemi", "Jonathan", "Melissa", "Edgar"]
numeros = [1,2,3,4,5,6,7]
combi = ["hola", 12, [1,2]]
print(nombres)
```

['Noemi', 'Jonathan', 'Melissa', 'Edgar']

```
[5]: # Vamos a hablar de gatitos
nombre = "Panchito"
edad = 4
personalidad = "jugueton"
```

```
[6]: def gatite(name, age, style):
    print("Le gatite ", name, "tiene ", age, "años y su personalidad es ", style)
```

```
[7]: gatite(nombre, edad, personalidad)
gatite("polo", 7, "cariñoso")
```

Le gatite Panchito tiene 4 años y su personalidad es jugueton
Le gatite polo tiene 7 años y su personalidad es cariñoso

1 Estructura condicional if, elif, else (Decisiones)

1.1 Lógica:

“Si pasa algo -> haz esto. Si no haz otra cosa”

” > mayor que, < menor que, <= menor que o igual, >= mayor que o igual, == igual, != diferente”

[8]: edad = 18

```
if edad >= 18:  
    print("Eres mayor de edad")  
else:  
    print("Eres menor de edad")
```

Eres mayor de edad

[9]: if edad >= 18:

```
    print("Eres mayor de edad")  
    años_para_cumplir_21 = 21-edad  
    print("Te faltan ", años_para_cumplir_21, "años para cumplir 21 años")  
else:  
    print("Eres menor de edad")  
    años_para_cumplir_21 = 21-edad  
    print("Te faltan ", años_para_cumplir_21, "años para cumplir 21 años")
```

Eres mayor de edad

Te faltan 3 años para cumplir 21 años

[10]: hora_de_comida = 3

```
contacto = 4  
contacto2 = 3
```

Tengo una regla de que si alguien necesita apoyo en mi hora de comida no puedo, todas las demás horas si.

```
if contacto != hora_de_comida:  
    print(" Si te puedo atender")  
else:  
    print("No te puedo atender")
```

Si te puedo atender

[11]: if contacto2 != hora_de_comida:

```
    print(" Si te puedo atender")  
else:  
    print("No te puedo atender")
```

No te puedo atender

1.2 Ejercicio

```
[21]: # input me permite obtener respuestas o interacción, el script se pausa ↵ esperando una respuesta, y siempre esa respuesta será ↵ una cadena de texto, si quiero cambiarle el tipo pongo int para entero, antes. edad = int(input("Cuantos años tienes:"))

if edad >= 18:
    print("Puedes aprender ciencia de datos sin permiso de tus papás")
else:
    print("Que chido que empieces joven")
```

Cuantos años tienes: 36

Puedes aprender ciencia de datos sin permiso de tus papás

2 Estructura de repetición for (Ciclo controlado)

2.1 Lógica:

“Repite esta acción una cantidad determinada de veces”

```
[19]: for i in range(3,7):
        print("Hola ", i)
```

Hola 3
Hola 4
Hola 5
Hola 6

```
[13]: for i in nombres:
        print(i)
```

Noemi
Jonathan
Melissa
Edgar

2.2 Ejercicio

```
[22]: lenguajes = ["Python", "R", "SQL", "Julia", "Scala"]

for lenguaje in lenguajes:
    print("Uno de los lenguajes más usados es:", lenguaje)
```

Uno de los lenguajes más usados es: Python
Uno de los lenguajes más usados es: R
Uno de los lenguajes más usados es: SQL
Uno de los lenguajes más usados es: Julia
Uno de los lenguajes más usados es: Scala

3 Estructura de repetición while (Ciclo condicionado)

3.1 Lógica:

“Repite algo mientras se cumple una condición”

```
[14]: x = 0

while x < 5:
    print(x)
    #x = x+1
    x += 1
```

```
0
1
2
3
4
```

```
[23]: dias = 0

while dias < 3:
    print("Estudia un poco más, Día ", dias + 1)
    dias += 1

print ("Bien hecho has estudiado al menos 3 días esta semana")
```

```
Estudia un poco más, Día  1
Estudia un poco más, Día  2
Estudia un poco más, Día  3
Bien hecho has estudiado al menos 3 días esta semana
```

4 Cadenas de texto

```
[25]: texto = "Hola mundo"
```

```
[28]: type(texto)
```

```
[28]: str
```

```
[32]: texto.upper()
```

```
[32]: 'HOLA MUNDO'
```

```
[33]: texto.lower()
```

```
[33]: 'hola mundo'
```

```
[36]: texto.replace("Hola", "Hellooow")  
[36]: 'Hellooow mundo'  
  
[37]: len(texto)  
[37]: 10  
  
[38]: texto[0]  
[38]: 'H'  
  
[39]: texto[4]  
[39]: ''  
  
[40]: texto[0:4]  
[40]: 'Hola'  
  
[42]: texto[-1]  
[42]: 'o'  
  
[44]: texto[-2]  
[44]: 'd'  
  
[47]: texto[9]  
[47]: 'o'  
  
[48]: texto[5:]  
[48]: 'mundo'  
  
[49]: texto[:5]  
[49]: 'Hola '  
  
[50]: texto[5:] + texto[:5]  
[50]: 'mundoHola '
```

5 Listas

```
[52]: mi_lista = [1,2,3, "Python", True]
```

```
[53]: mi_lista[0]
[53]: 1
[54]: mi_lista[-1]
[54]: True
[55]: mi_lista.append(5) # me sirve para agregar elementos al final de la lista
[57]: mi_lista.append("seis")
[58]: mi_lista
[58]: [1, 2, 3, 'Python', True, 5, 'seis']
[61]: mi_lista.insert(3,4) # insert me agrega el elemento en la posición indicada
    ↪ primero va la posición y luego el elemento a agregar
[62]: mi_lista
[62]: [1, 2, 3, 4, 4, 'Python', True, 5, 'seis']
[63]: mi_lista.remove(4) # quita el primer elemento que haga match con el valor que le
    ↪ indique
[64]: mi_lista
[64]: [1, 2, 3, 4, 'Python', True, 5, 'seis']
[65]: mi_lista.insert(5,5)
[66]: mi_lista
[66]: [1, 2, 3, 4, 'Python', 5, True, 5, 'seis']
[67]: mi_lista.remove(5)
[68]: mi_lista
[68]: [1, 2, 3, 4, 'Python', True, 5, 'seis']
[70]: for i in range(1,5):
        mi_lista.remove(i)
        print(mi_lista)

[2, 3, 4, 'Python', True, 5, 'seis']
[3, 4, 'Python', True, 5, 'seis']
```

```
[4, 'Python', True, 5, 'seis']
['Python', True, 5, 'seis']

[71]: mi_lista.pop() #Elimina el último elemento

[71]: 'seis'

[72]: mi_lista

[72]: ['Python', True, 5]

[73]: mi_lista.sort()

-----
TypeError                                         Traceback (most recent call last)
Cell In[73], line 1
----> 1 mi_lista.sort()

TypeError: '<' not supported between instances of 'bool' and 'str'

[74]: numeros = [7,5,1,4,3]

[75]: numeros.sort() # Ordena listas que tengan valores todos del mismo tipo

[76]: numeros

[76]: [1, 3, 4, 5, 7]

[77]: numeros.reverse() # Invierte el orden

[78]: numeros

[78]: [7, 5, 4, 3, 1]

[79]: mi_lista.append(5)

[80]: mi_lista

[80]: ['Python', True, 5, 5]

[81]: mi_lista.count(5) #Cuenta las veces que se repite un valor en la lista

[81]: 2

[82]: len(mi_lista) #para saber la longitud de mi lista

[82]: 4
```

```
[83]: mi_lista.append([2,3])  
[84]: mi_lista  
[84]: ['Python', True, 5, 5, [2, 3]]  
[85]: mi_lista[4]  
[85]: [2, 3]  
[86]: mi_lista[4][0]  
[86]: 2  
[87]: mi_lista[4][1]  
[87]: 3
```

6 Tuplas

```
[88]: mi_tupla = (1,2,3,"Python")  
[89]: mi_tupla[2]  
[89]: 3  
[91]: mi_tupla.count(3)  
[91]: 1  
[92]: mi_tupla.index("Python")  
[92]: 3  
[93]: list(mi_tupla)  
[93]: [1, 2, 3, 'Python']  
[94]: tuple(mi_lista)  
[94]: ('Python', True, 5, 5, [2, 3])
```

7 Diccionarios

```
[99]: mi_dict = {"nombre":"Python", "version":3.12, "activo":True}  
[100]: mi_dict
```

```
[100]: {'nombre': 'Python', 'version': 3.12, 'activo': True}

[105]: mi_dict["nombre"]

[105]: 'Python'

[102]: mi_dict["version"]

[102]: 3.12

[103]: mi_dict["activo"]

[103]: True

[104]: mi_dict["año"] = 2024

[106]: mi_dict

[106]: {'nombre': 'Python', 'version': 3.12, 'activo': True, 'año': 2024}

[107]: mi_dict.keys()

[107]: dict_keys(['nombre', 'version', 'activo', 'año'])

[108]: for key in mi_dict.keys():
    print(key)

nombre
version
activo
año

[109]: mi_dict.values()

[109]: dict_values(['Python', 3.12, True, 2024])

[110]: for value in mi_dict.values():
    print(value)

Python
3.12
True
2024

[111]: mi_dict.items()

[111]: dict_items([('nombre', 'Python'), ('version', 3.12), ('activo', True), ('año', 2024)])
```

```
[113]: for key,value in mi_dict.items():
         print(key, ":", value)

nombre : Python
version : 3.12
activo : True
año : 2024

[115]: mi_dict.pop("activo")

[115]: True

[116]: mi_dict

[116]: {'nombre': 'Python', 'version': 3.12, 'año': 2024}

[117]: mi_dict.clear()

[118]: mi_dict

[118]: {}

[119]: dicton = {}

[120]: dicton

[120]: {}

[121]: lista = []

[122]: lista

[122]: []

[123]: materias = {}

[125]: materias["Lunes"]= [6103, 7540]
      materias["Martes"]= [6201]
      materias["Miércoles"]= [6103, 7540]
      materias["Jueves"]= []
      materias["Viernes"]= [6201]

[126]: materias

[126]: {'Lunes': [6103, 7540],
      'Martes': [6201],
      'Miércoles': [6103, 7540],
      'Jueves': [],
      'Viernes': [6201]}
```

```
[127]: materias["domingo"]
```

```
-----  
KeyError Traceback (most recent call last)  
Cell In[127], line 1  
----> 1 materias["domingo"]  
  
KeyError: 'domingo'
```

```
[128]: #get me sirve para revisar si tengo una clave dentro de mi dict, sin que me de  
    ↪error,  
# el primer parametro es la clave y el segundo un mensaje si no la encuentra  
    ↪esa clave  
materias.get("domingo", "no existe")
```

```
[128]: 'no existe'
```

```
[130]: materias.get("Lunes")
```

```
[130]: [6103, 7540]
```

```
[133]: import numpy
```

8 Funciones

- “def”: palabra reservada para definir una función
- saludar: nombre de la función
- (): donde puedo indicar los datos de entrada o parametros
- “:” : indican que lo que sigue es el bloque de la función
- El código dentro de mi función está identado

```
[135]: def saludar():  
    print("Hola, bienvenida al mundo de la ciencia de datos")
```

```
[136]: saludar()
```

Hola, bienvenida al mundo de la ciencia de datos

```
[140]: # funciones con parametros  
name = "Edgar"  
  
def saludar_persona(nombre):  
    print("Hola ", nombre, "bienvenida al curso")  
  
saludar_persona("Haydé")  
saludar_persona("Jonathan")  
saludar_persona(name)
```

```
Hola Haydé bienvenida al curso  
Hola Jonathan bienvenida al curso  
Hola Edgar bienvenida al curso
```

```
[141]: # funciones con input
```

```
def saludar_input():  
    nombre = input("Cual es tu nombre?")  
    print("Hola ", nombre)
```

```
[3]: # Return
```

```
def sumar(a,b):  
    resultado = a + b  
    return resultado  
    hola = 123  
    print(hola)
```

```
[4]: sumar(1,2)
```

```
[4]: 3
```

```
[ ]:
```