Python

Definición: **interpretes** + **entornos**
Hola mundo
Enteros, reales y operadores
Booleanos, operadores lógicos y cadenas
Listas y Tuplas
Bucles y funciones
Clases

Definición: interpretes + entornos

Compilado: Se fabrica el programa antes de ser ejecutado. C.

Interpretado: Se puede ejecutar el programa instrucción a instrucción sin necesidad de estar completo.

Python: **Lenguaje interpretado**

Nota sobre versiones...

Python 2.x: Presentes por motivos de compatibilidad con software y librerías ya desarrolladas que no han sido actualizadas.

Python 3.x: Versiones actuales del lenguaje de programación.

NO SON COMPATIBLES

Instalación...

Linux: apt install

Windows: Descarga de python.org

Online: Entornos Jupyter. (cocalc)

```
print("Hola Mundo")
```

- No hay caractér de fin de sentencia
- Tiene un tipado debil

```
a = 5
b = "pepe"
Print(a)
Print(b)
b=a+2
print(b)
print("La variable a es",a," chupate esa")
a
b
```

- **bool** : Todo cero o vacío es falso. El resto es verdadero.

```
bool(0)
bool(25)
bool([])
bool("232")
bool("")
```

-**Operadores lógicos**: and, or y not

```
True and false
Not false
True or false

3 and 4 → 4
5 or 6 → 5
????

+ detalles
```

-**Entrada por teclado**: input()

```
Variable = input()
int(Variable)
float(Variable)

Int   ← Convierten el texto a números.
float
```

-Condiciones if .. else

```
if edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")
print("¡Hasta la próxima!")
```

-Bucles For

```
For i in [3,4,5]:
    print("*")

For i in range(2,10):
     print(“*”)
```

-Bucles While

```
i = 1
while i <= 3:
    print(i)
    i += 1
print("Programa terminado")
```

-Tuplas

```
(1,”b”,true)
“8”,3,1.24.”hola”

len((1,”b”,true))
3

()
Len(())
0
```

Son inmutables!!!

-Tuplas

Son inmutables!!!

```
>>> tupla = ("uno", "dos", "tres")
>>> tupla[0] = "cuatro"
Traceback (most recent call last):
  File "", line 1, in
TypeError: 'tuple' object does not support item assignment
```

-Tuplas – Operaciones con

```
Concatenación
>>>tupla1 = (1,2,3,4,5)
>>>tupla2 = (6,7,8,9,10)
>>>tup = tupla1 + tupla2
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

Repetición
>>>tupla1 * 3
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

-Tuplas – Operaciones con (II)

```
Comprobación
>>>7 in tupla1
False


Contar número de ocurrencias
>>>tupla3 =(2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
>>>tupla3.count(1)
2
```

-Tuplas – Operaciones con (III)

```
¿En qué posición de la tupla está el valor 3?
>>>tupla1.index(3)
2

¿qué hay en la posición 4 de la tupla?
>>>tupla1[4]
5

Extracción
>>>tupla1[1:3]
(2,3)
```

-Listas

```
>>>primos = [2, 3, 5, 7, 11, 13]
>>>diasLaborables = ["Lunes", "Martes", "Miércoles",
"Jueves", "Viernes"]
>>>fecha = ["Lunes", 27, "Octubre", 1997]
>>> peliculas = [ ["Senderos de Gloria", 1957], ["Hannah y
sus hermanas", 1986]]
```

Son mutables!!!

-Listas

Son mutables!!!

```
>>> lista = ["uno", "dos", "tres"]
>>> lista[0] = "cuatro"
>>> lista
['cuatro', 'dos', 'tres']
```

-Listas

Pueden tener muchos niveles de anidamiento:

```
directores = [ ["Stanley Kubrick", ["Senderos de Gloria",
1957]], ["Woody Allen", ["Hannah y sus hermanas", 1986]] ]
```

-Listas

Acceso a elementos por indices:

```
>>>>>> lista = [10, 20, 30, 40]
>>> lista[2]
30
>>> lista[0]
10
```

-Listas

Concatenar listas:

```
>>> vocales = ["E", "I", "O"]
>>> vocales
['E', 'I', 'O']
>>> vocales = vocales + ["U"]
>>> vocales
['E', 'I', 'O', 'U']
```

-Listas

Se pueden definir desde variables y son inmutables:

```
>>> nombre = "Pepe"
>>> edad = 25
>>> lista = [nombre, edad]
>>> lista
['Pepe', 25]
>>> nombre = "Juan"
>>> lista
['Pepe', 25]
```

-Rangos:

Forma de crear listas

```
>>>x = range(10);
>>>x
range(0,10)
>>>list(x)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>>list(range(7))
[0, 1, 2, 3, 4, 5, 6]
```

No existen los rangos negativos

```
>>>list(range(-5))
[]
```

Se pueden expandir entre dos números

```
>>>list(range(-5,1))
[-5, -4, -3, -2, -1, 0, 1]
```

Cuidado!!!

```
>>> list(range(5, 1))
[]
>>> list(range(3, 3))
[]
```

Se puede utilizar con 3 parámetros

```
>>>list(range(5, 21, 3))
[5, 8, 11, 14, 17, 20]

>>> list(range(10, 0, -2))
[10, 8, 6, 4, 2]
```

Se pueden concatenar listas generadas con varios rangos

```
>>> list(range(3)) + list(range(5))
[0, 1, 2, 0, 1, 2, 3, 4]
```