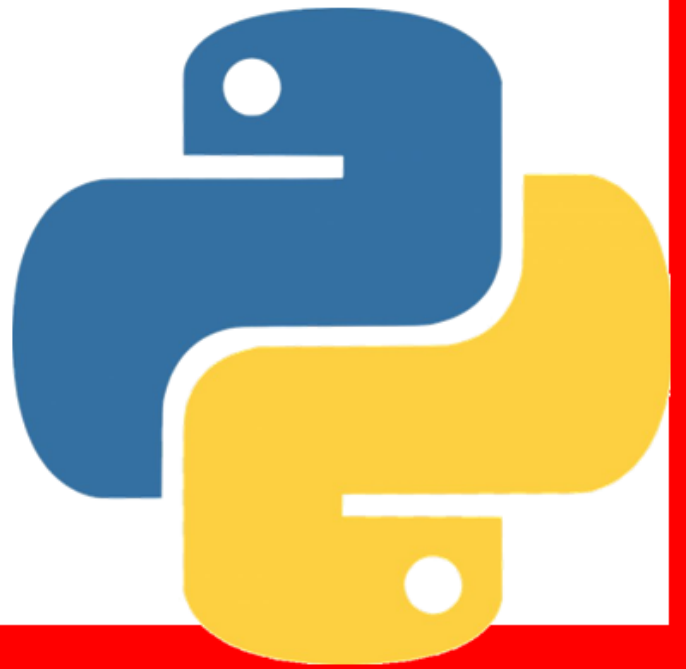
Tipos de datos, expresiones y operaciones

04

Curso Básico python



Comentarios y salida en pantalla

- Los comentarios en Python comienzan con el carácter numeral, #, y se extienden hasta el final físico de la línea.

  #Este es un comentario

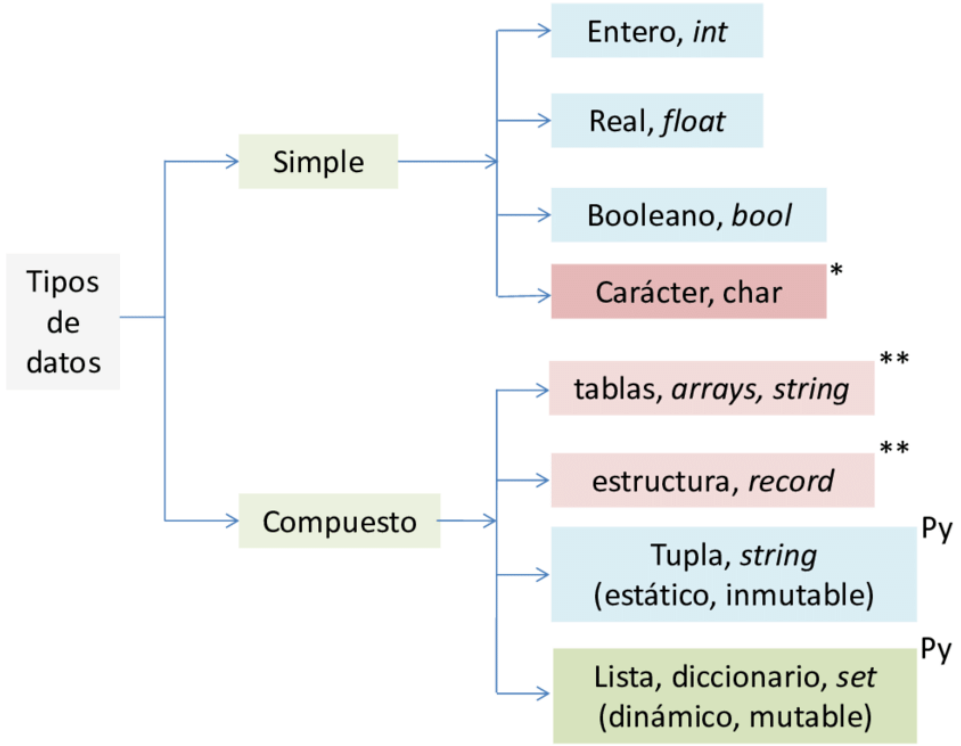
- Para imprimir en pantalla utilizamos:

  print(“mensaje”)

```
>>> print("Este es un mensaje") #Este es un comentario
Este es un mensaje
>>>
```

Tipos de datos

- Los tipos básicos son:
  - Integer <int>
  - Float <float>
  - String <str>
  - Boolean <bool>



Tipos de datos:

*El tipo de dato carácter no existe en Python, un carácter simple se representa como cadena de caracteres (**string).**

** **Estructuras compuestas de lenguajes como** C, FORTRAN, Pascal, Matlab, etc.

Py: Estructuras compuestas en Python.

type(4) #Devuelve el tipo de dato

Variables

- Publicas:
  - Mensaje = “Hola Mundo”
- Privadas
  - _msg = “Mensaje privado”
- Constantes (si pueden cambiar)
  - MSG = “mi constante”
- Var. Con __ (doble guion bajo) no deberian cambiarse
  - __no cambiar= “No le muevas”
- Reasignacion de variables
- Tipado dinámico

- X=67
- X= 50
- type(X)
- X=“Hola”
- type(X)

Reglas de los identificadores

- Variables
  - Pueden contener números y letras
  - No deben comenzar con número
  - Múltiples palabras se unen con _
    - multiple_words
  - No se pueden utilizar palabras reservadas

```
>>> help("keywords")
False               def                 if                  raise
None                del                 import              return
True                elif                in                  try
and                 else                is                  while
as                  except              lambda              with
assert              finally             nonlocal            yield
break               for                 not
class               from                or
continue            global              pass
```

Asignación

- a=7
- b = 89.45
- c= ‘hola’
- cad = “hola”
- var = True

```
>>> a = 7
```

Python permite asignaciones múltiples del tipo

```
>>> x, y, z = 7, 8.2, 9
```

```
>>> var = False
>>> type(var)
<class 'bool'>
```

Operadores aritméticos

Operación	Operador	Expresión	Resultado tipo
Suma	+	a + b	Entero si a y b enteros; real si alguno es real
Resta	-	a - b	Entero si a y b enteros; real si alguno es real
Multiplicación	*	a * b	Entero si a y b enteros; real si alguno es real
División, a÷b (real)	/	a / b	Siempre es real
División (entera)	//	a // b	Devuelve la parte entera del cociente a÷b
Módulo (resto)	%	a % b	Devuelve el resto de la división a÷b
Exponenciación, a^b	**	a**b	Entero si a y b enteros; real si alguno es real

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
```

```
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>>
>>> 17 // 3  # floor division discards the fractional part
5
>>> 17 % 3  # the % operator returns the remainder of the division
2
>>> 5 * 3 + 2  # result * divisor + remainder
17
```

```
>>> 5 ** 2  # 5 squared
25
>>> 2 ** 7  # 2 to the power of 7
128
```

Operadores aritméticos con asignaciones

Operación	Operador	Expresión	Equivalente a
Suma y asigna	+=	a += b	a = a+b
Resta y asigna	-=	a -= b	a = a-b
Multiplica y asigna	*=	a *= b	a = a*b
Divide y asigna	/=	a /= b	a = a/b
Divide y asigna la parte entera	//=	a //= b	a = a//b
Módulo y asigna	%=	a %= b	a = a%b
Potencia y asigna	**=	a **= b	a = a**b

```
>>> c = c + 1     # la variable c se incrementa en 1
>>> d = d - 1     # la variable d se decrementa en 1
```

```
>>> c += 1        # equivale a: c = c + 1
>>> x += 0.01     # equivale a: x = x + 0.01
>>> d -= 2        # equivale a: d = d - 2
```

Operadores lógicos

A	B	A or B	A and B	not A
False	False	False	False	True
False	True	True	False	True
True	False	True	False	False
True	True	True	True	False

equivale

A	B	A or B	A and B	not A
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

```
>>> A = True
>>> type(A)
<class 'bool'>
>>> B = False
>>> A or B
True
```

Operadores relacionales (comparison)

Operadores Relacionales

Matemáticas	En Python	Significado	Ejemplo	Resultado
=	==	Igual a	'a' == 'b'	False
≠	!=	Distinto a	'b' != 'B'	True
<	<	Menor que	7 < 3	False
>	>	Mayor que	7 > 3	True
≤	<=	Menor o igual que	7 <= 7	True
≥	>=	Mayor o igual que	7 >= 3	True

```
>>> temp = 38     # temperatura medida
>>> (temp >= 37) and (temp <= 42)
True
```

```
>>> car = 'q'
>>> (car >= 'a') and (car <= 'z') or (car >= 'A') and (car <= 'Z')
True
>>> # Equivale a
>>> 'a' <= car <= 'z' or 'A' <= car <= 'Z'
True
>>> car = '&'
>>> 'a' <= car <= 'z' or 'A' <= car <= 'Z'
False
```

```
>>> es_letra = 'a' <= car <= 'z' or 'A' <= car <= 'Z'
>>> es_letra
False
```

Orden de los operadores

- PEMDAS
  - Paréntesis
  - Exponentes
  - Multiplicación / División
  - Adición / Sustracción

Operador	Precedencia
()	Mayor
**	|
+x, -x (identidad, cambio de signo)	|
*, /, //, %	|
+, -	|
==, !=, <, >, <=, >=	|
not	|
and	v
or	Menor

Operaciones básicas con texto (strings)

```
>>> '40' + '8'
'408'
>>> Name = 'Luis'
>>> Apellido = 'Garcia'
>>> Name + ' ' + Apellido
'Luis Garcia'
>>> 'Ven'*4        # equivale a 'Ven'+'Ven'+'Ven'+'Ven'
'VenVenVenVen'
>>> 10*'-'
'----------'
```

Lectura de datos

```
>>> Nombre = input('Cómo te llamas? ')
Cómo te llamas? José
>>> type(Nombre)
<class 'str'>
>>> Edad = int(input('Introduce tu edad: '))
Introduce tu edad: 21
>>> type(Edad)
<class 'int'>
>>> Altura = float(input('Cuánto mides? '))
Cuánto mides? 1.78
>>> type(Altura)
<class 'float'>
```

- r = input(“Mensaje”)
- Utiliza un mensaje de texto.
- Lo que se teclea y se introduce al programa, que se suele asignar a una variable, es un valor de tipo string.
- Hay que convertirlo a (int, (float) o (bool)

Conversión entre tipos de datos

- float(), int(),str()

```
>>> int('123')
123
>>> int(27.8)
27
>>> int(-24.9)
-24
```

```
>>> float('123')
123.0
>>> float(Edad)  # Variable Edad de tipo int
21.0
>>> float('abc')
Traceback (most recent call last):
  File "<pyshell#84>", line 1, in <module>
    float('abc')
ValueError: could not convert string to float: 'abc'
```

```
>>> bin(255)
'0b11111111'
>>> bin(256)
'0b100000000'
>>> str(254)
'254'
>>> str(1/3)
'0.3333333333333333'
```

Escritura de datos

- La función print() permite mostrar valores de variables (de cualquier tipo) o expresiones, texto en formato string o combinaciones de todas ellas.

```
>>> a = 3.5
>>> b = 15
>>> print('El producto de', a ,  'por', b ,  'es', a*b)
El producto de 3.5 por 15 es 52.5
>>> print('El producto de '+str(a)+' por '+str(b)+' es '+str(a*b))
El producto de 3.5 por 15 es 52.5
```

Existen parámetros de la función `print()` que indican cómo terminar la escritura (`end`) y cómo separar (`sep`) los elementos del argumento de la función. Cuando se quiera terminar la función `print` sin que salte a una nueva línea, y un nuevo `print` continúe en la misma línea, se agrega al final el argumento `end=''` (comillas simples sin espacio entre ellas). Cuando no se usa el `end`, se utiliza el valor por omisión de salto a nueva línea, que sería `end='\n'`. En lugar de separar las secuencias por espacios en blanco, se puede omitir la separación (`sep=''`) o colocar el string que se quiera como símbolo separador.

File Edit Format Run Options Window Help

```
print('Q', end='')
print('u', end='')
print('é!', end='\n') # equivale a print('é!')
print('A', 'B', 'C')
print('A', 'B', 'C', sep=',')
print('A', 'B', 'C', sep='')
print('10','27','59', sep=':')
print('10','27','59', sep='----')
```

```
Qué!
A B C
A,B,C
ABC
10:27:59
10----27----59
>>>
```

Tipos de errores

Sintácticos:

```
>>> 4/*8
SyntaxError: invalid syntax
>>> pirnt('Hola mundo!')
Traceback (most recent call last):
  File "<pyshell#85>", line 1, in <module>
    pirnt('Hola mundo!')
NameError: name 'pirnt' is not defined
```

De ejecución:

```
>>> alumnos = 56
>>> grupos = 0
>>> alum_grupo = alumnos/grupos
Traceback (most recent call last):
  File "<pyshell#90>", line 1, in <module>
    alum_grupo = alumnos/grupos
ZeroDivisionError: division by zero
```