Tipos de datos

Tipos

- Hemos definido una variable como un contenedor de información pero, ¿toda la información es igual? Obviamente no. No es lo mismo un número entero que un número real, texto, una fecha, etc.
- Dentro del ordenador la información se guarda toda igual como números binarios (1 y 0). Necesitamos de un discriminador que aporte información de cómo interpretar esa información bruta. Ese discriminador es lo que llamamos tipo.

Tipado

- Python es un lenguaje de programación con un tipado fuerte y dinámico.
  - Fuerte: dado el valor de una variable de un tipo concreto, no se puede usar como si fuera de otro tipo distinto a menos que se haga una conversión.
  - Dinámico: una misma variable puede tomar valores de distinto tipo. Basta con reasignar la variable con otro valor de distinto tipo.

Tipos básicos en Python

- Aunque más adelante veremos como crear nuestros propios tipos en Python, existen tipos predefinidos en Python. Ya nos hemos encontrado con alguno de ellos:
  - Tipo nulo (None): representa la ausencia de tipo.

    ```
    Ej:  a = None # a no tiene tipo
    ```

  - Tipo entero (int): representa un número cualquiera. Precisión arbitraria.

    ```
    Ej:  a = 4 # a es una variable tipo int
         a = 1e50 # no importa el tamaño
    ```

Tipos básicos en Python

- Tipo real (float): representa un número real cualquiera. Precisión arbitraria.

  Ej:
  ```
  a = 1.5 # a es una variable tipo float
  a = 5/2 # es resultado de divisiones
  ```

  Hay que indicar que trunca los números que no puede procesar:

  Ej:
  ```
  a = 2/3
  print(a)
  >> 0,6666666666666666
  ```

Tipos básicos en Python

- Tipo complejo (complex): Python maneja también números complejos.

```
Ej:  a = 1.34 + 6j # j para √-1
     b = 2j
     print(a*b)
     >> (-12+2.68j)
```

- Tipo booleano (bool): solo acepta 2 valores, True y False.

```
Ej: a = True
```

Tipos básicos en Python

- Tipo string (str): guarda texto.

```
Ej:  a = “Hola Mundo!” # Comillas dobles
     b = ‘Hola Mundo’ # Comillas simples
     # Ambas expresiones son equivalentes
     c = ‘‘‘Hola esto es texto
           varias líneas’’’
     # Texto multilínea
```

Nota: Python permite el uso de caracteres especiales como ñ o ç.

Tabla de operadores

Primero Segundo	int	float	complex	bool	str
int	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= And, or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and, or	*
float	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	
complex	+,-,*,**,/,% == and,or	+,-,*,**,/,% == and,or	+,-,*,**,/,% == and,or	+,-,*,**,/,% == and,or	
bool	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	+,-,*,**,/,//,% ==,!=,<,<=,>,>= and,or	*
str	*			*	+

Conversión de tipo

- Podemos convertir y definir tipos de variables usando el nombre de los tipos:

```
Ej:  a = int() # a es un entero que vale 0
     b = 1.4
     b = int(b) # b vale 1, se ha truncado
     c = int("58") # c es un entero
     d = int("101", base=2) # d es 5
     e = str(10.4) # e es texto
```

- str() acepta cualquier tipo de argumento.

type

- Podemos saber el tipo de una variable o valor usando type.

  Ej: type(3) → int

  type(“Hola Mundo!”) → str

  type(1.4) → float

- También podemos saber si una variable o valor pertenece a un tipo concreto con el operador is.

  Ej: type(“Python!”) is float → False