

# Tipos de datos básicos

## Datos enteros

Un dato entero en Python, representado por el tipo *int*, es un tipo de dato numérico que almacena valores sin decimales, tanto positivos como negativos, y puede ser de cualquier tamaño sin límite de precisión.

### Ejemplo de uso de variables de tipo int

```
a = 4
b = 7
c = a + b
print(f"El valor de a es: {a}")
print(f"El valor de b es: {b}")
print(f"El valor de c es: {c}")
print(f"El tipo de a es: {type(a)}")
print(f"El tipo de b es: {type(b)}")
print(f"El tipo de c es: {type(c)}")
```

```
El valor de a es: 4
El valor de b es: 7
El valor de c es: 11
El tipo de a es: <class 'int'>
El tipo de b es: <class 'int'>
El tipo de c es: <class 'int'>
```

### Ejemplo: Cálculos con números enteros

```
# Definir 2 números enteros
num1 = 15
num2 = 4
```

```
# Realizar operaciones aritméticas
suma = num1 + num2
resta = num1 - num2
multiplicacion = num1 * num2
division_entera = num1 // num2
modulo = num1 % num2

# Imprimir los resultados
print(f"El tipo de suma es: {type(suma)} y su valor es {suma}")
print(f"El tipo de resta es: {type(resta)} y su valor es {resta}")
print(f"El tipo de multiplicacion es: {type(multiplicacion)} \
y su valor es {multiplicacion}")
print(f"El tipo de division_entera es: {type(division_entera)} \
y su valor es {division_entera}")
print(f"El tipo de modulo es: {type(modulo)} y su valor es {modulo}")
```

```
El tipo de suma es: <class 'int'> y su valor es 19
El tipo de resta es: <class 'int'> y su valor es 11
El tipo de multiplicacion es: <class 'int'> y su valor es 60
El tipo de division_entera es: <class 'int'> y su valor es 3
El tipo de modulo es: <class 'int'> y su valor es 3
```

## Flotantes

Un flotante en Python, representado por el tipo *float*, es un tipo de dato numérico que almacena valores con decimales, permitiendo representar números fraccionarios y realizar cálculos con precisión decimal.

### Ejemplo de uso de variables de tipo float

```
a = 4
b = 7.2
c = a + b
print(f"El valor de a es: {a}")
print(f"El valor de b es: {b}")
print(f"El valor de c es: {c}")
print(f"El tipo de a es {type(a)}")
print(f"El tipo de b es {type(b)}")
print(f"El tipo de c es {type(c)}")
```

```
El valor de a es: 4
```

El valor de b es: 7.2  
El valor de c es: 11.2  
El tipo de a es <class 'int'>  
El tipo de b es <class 'float'>  
El tipo de c es <class 'float'>

### Ejemplo: Vamos a convertir grados celsius a Fahrenheit

Utilice la fórmula

$$F = \frac{9}{5}C + 32$$

```
celsius = 24

fahrenheit = celsius * 9 / 5 + 32

print(f"La temperatura en grados Fahrenheit es: {fahrenheit}")
print(f"El tipo de dato de la variable fahrenheit es: {type(fahrenheit)}")
```

La temperatura en grados Fahrenheit es: 75.2  
El tipo de dato de la variable fahrenheit es: <class 'float'>

### Complejos

Un número complejo en Python, representado por el tipo *complex*, es un tipo de dato numérico que almacena valores con una parte real y una parte imaginaria, permitiendo realizar cálculos en el plano complejo. Los números complejos se escriben en la forma  $a + bj$ , donde  $a$  es la parte real y  $b$  es la parte imaginaria.

### Ejemplo de uso de variables de tipo complex

```
z1 = 4 + 2j
z2 = 1 + 3j
z3 = z1 + z2
print(f"El valor de z1 es: {z1}")
print(f"El valor de z2 es: {z2}")
print(f"El valor de z3 es: {z3}")
print(f"El tipo de z1 es: {type(z1)}")
print(f"El tipo de z2 es: {type(z2)}")
print(f"El tipo de z3 es: {type(z3)}")
```

```
El valor de z1 es: (4+2j)
El valor de z2 es: (1+3j)
El valor de z3 es: (5+5j)
El tipo de z1 es: <class 'complex'>
El tipo de z2 es: <class 'complex'>
El tipo de z3 es: <class 'complex'>
```

```
z1 = 2 + 3j
z2 = 4 - 2j

suma = z1 + z2
resta = z1 - z2
multiplicacion = z1 * z2
division = z1/z2

print(f"Suma: {suma}")
print(f"Resta: {resta}")
print(f"Multiplicacion: {multiplicacion}")
print(f"Division: {division}")
```

```
Suma: (6+1j)
Resta: (-2+5j)
Multiplicacion: (14+8j)
Division: (0.1+0.8j)
```

## Booleanos

Los booleanos en Python, representados por el tipo *bool*, son un tipo de dato que puede tener uno de dos valores: True o False. Se utilizan principalmente para representar condiciones y controlar el flujo de ejecución en estructuras de control como condicionales y bucles.

### Ejemplo de uso de variables de tipo bool

```
a = 4
b = 4
c = a == b
print(f"El valor de a es: {a}")
print(f"El valor de b es: {b}")
print(f"El valor de c es: {c}")
print(f"El tipo de a es: {type(a)}")
print(f"El tipo de b es: {type(b)}")
print(f"El tipo de c es: {type(c)}")
```

```
El valor de a es: 4
El valor de b es: 4
El valor de c es: True
El tipo de a es: <class 'int'>
El tipo de b es: <class 'int'>
El tipo de c es: <class 'bool'>
```

### Ejemplo: Verificar si una contraseña es correcta

```
contraseña_correcta = "H2G4"
contraseña_prueba = "H2G5"
verificacion = contraseña_correcta == contraseña_prueba
print(f"El valor de verificacion es: {verificacion}")
print(f"El tipo de verificacion es: {type(verificacion)}")
```

```
El valor de verificacion es: False
El tipo de verificacion es: <class 'bool'>
```

### Cadenas de caracteres (Strings)

Los strings en Python, representados por el tipo *str*, son secuencias de caracteres utilizadas para almacenar y manipular texto. Se pueden definir usando comillas simples ‘Texto’ o dobles “Texto”.

#### Ejemplo utilizando variables de tipo str

```
a = "Mi nombre es: "
b = "José Ortega"
c = a + b
print(f"El valor de a es: {a}")
print(f"El valor de b es: {b}")
print(f"El valor de c es: {c}")
print(f"El tipo de a es: {type(a)}")
print(f"El tipo de b es: {type(b)}")
print(f"El tipo de c es: {type(c)}")
```

```
El valor de a es: Mi nombre es:
El valor de b es: José Ortega
El valor de c es: Mi nombre es: José Ortega
El tipo de a es: <class 'str'>
El tipo de b es: <class 'str'>
El tipo de c es: <class 'str'>
```

### Otro ejemplo utilizando strings

```
nombre = "José"  
curso = "R"  
print(f"Este curso de {curso} fue creado por {nombre}")
```

Este curso de R fue creado por José