Tipos de Datos - Funciones/Metodos Tipicos

Int

Representa números enteros. Algunas de sus funciones y métodos típicos son:

- `abs(x)`: devuelve el valor absoluto de `x`. Ejemplo: `abs(-5)` devuelve `5`.

```
result = abs(-5)
print(result)

>>> 5
```

- `divmod(x, y)`: devuelve el cociente y el resto de dividir `x` entre `y`. Ejemplo: `divmod(10, 3)` devuelve `(3, 1)`.

```
result = divmod(10, 3)
print(result)

>>> (3, 1)
```

- `pow(x, y)`: devuelve `x` elevado a la potencia `y`. Ejemplo: `pow(2, 3)` devuelve `8`.

```
result = pow(2, 3)
print(result)

>>> 8
```

- `bin(x)`: convierte un número entero en una cadena binaria. Ejemplo: `bin(10)` devuelve `'0b1010'`.

```
result = bin(10)
print(result)

>>> '0b1010'
```

Float

Representa números con punto flotante (números decimales). Algunas de sus funciones y métodos típicos son:

- Los métodos disponibles para los números flotantes son similares a los disponibles para los números enteros.
- **`round(x[, n])`**: redondea el número **`x`** a la cantidad de decimales especificada por el segundo argumento opcional **`n`**. Si no se proporciona este segundo argumento, se redondea al entero más cercano. Ejemplo: **`round(3.14159)`** devuelve **`3`**, mientras que **`round(3.14159, 2)`** devuelve **`3.14`**.

```
result1 = round(3.14159)
result2 = round(3.14159, 2)
print(result1)
print(result2)

>>> 3
>>> 3.14
```

- También puedes usar funciones del módulo math como:
  - **`math.floor(x)`**: redondea hacia abajo al entero más cercano
  - **`math.ceil(x)`**: redondea hacia arriba al entero más cercano

Bool

Representa valores booleanos (**`True`** o **`False`**). Algunas de sus funciones y métodos típicos son los operadores lógicos como:

- **`and`**: devuelve **`True`** si ambos operandos son verdaderos. Ejemplo: **`True and False`** devuelve **`False`**.

```
result = True and False
print(result)

>>> False
```

- **`or`**: devuelve **`True`** si al menos uno de los operandos es verdadero. Ejemplo: **`True or False`** devuelve **`True`**.

```
result = True or False
print(result)
```

```
>>> True
```

- **`not`**: invierte el valor booleano del operando. Ejemplo: **`not True`** devuelve **`False`**.

```
result = not True
print(result)

>>> False
```

String

Representa cadenas de texto. Algunas de sus funciones y métodos típicos son:

- **`.capitalize()`**: devuelve una copia de la cadena con el primer carácter en mayúsculas y el resto en minúsculas. Ejemplo: **`'hello world'.capitalize()`** devuelve **`'Hello world'`**.

```
result = 'hello world'.capitalize()
print(result)

>>> 'Hello world'
```

- **`.lower()`**: devuelve una copia de la cadena con todos los caracteres en minúsculas. Ejemplo: **`'HELLO WORLD'.lower()`** devuelve **`'hello world'`**.

```
result = 'HELLO WORLD'.lower()
print(result)

>>> 'hello world'
```

- **`.upper()`**: devuelve una copia de la cadena con todos los caracteres en mayúsculas. Ejemplo: **`'hello world'.upper()`** devuelve **`'HELLO WORLD'`**.

```
result = 'hello world'.upper()
print(result)

>>> 'HELLO WORLD'
```

- **`.title()`**: devuelve una copia de la cadena con el primer carácter de cada palabra en mayúsculas y el resto en minúsculas. Ejemplo: **`'hello world'.title()`** devuelve **`'Hello World'`**.

```
result =  'hello world'.title()
print(result)

>>> 'Hello World'
```

- **`.replace(old, new[, count])`**: devuelve una copia de la cadena con todas las ocurrencias del substring **`old`** reemplazadas por **`new`**. Si se proporciona el tercer argumento opcional **`count`**, solo se reemplazan las primeras **`count`** ocurrencias. Ejemplo: **`'hello world'.replace('l', 'L')`** devuelve **`'heLLo worLd'`**, mientras que **`'hello world'.replace('l', 'L', 2)`** devuelve **`'heLLo world'`**.

```
result_1 = 'hello world'.replace('l', 'L')
print(result_1)
result_2 = 'hello world'.replace('l', 'L', 2)
print(result_2)

>>> 'heLLo worLd'
>>> 'heLLo world'
```

- **`len(s)`**: devuelve la longitud de la cadena **`s`**. Ejemplo: **`len('hello')`** devuelve **`5`**.

```
result = len('hello')
print(result)

>>> 5
```