



Clase Teórica - Más utilidades básicas en Python

 17 **Sesión 3 - 07/03/2025**


 **Objetivo:** Explicar cómo utilizar más operaciones básicas con variables en Python sin usar bucles ni métodos. Saber dónde vas a trabajar y utilizar funcionalidades que son muy útiles a la hora de diseñar código

1 Longitud de una cadena (len)

- ◆ Podemos medir cuántos caracteres tiene una cadena usando `len()`.

```
nombre = "Mario Flores"
```

```
print("Longitud del nombre:", len(nombre))
```

 Útil para validar el tamaño de una contraseña o contar caracteres.

2 Convertir texto a mayúsculas y minúsculas

- ◆ Métodos `upper()` y `lower()`.

```
print("En mayúsculas:", nombre.upper())
```

```
print("En minúsculas:", nombre.lower())
```

 Útil para normalizar datos.

3 Extraer parte de una cadena (Slicing)

- ◆ Podemos seleccionar solo una parte del texto.

```
print("Primeros 3 caracteres:", nombre[:3]) # "Mar"
```

```
print("Últimos 4 caracteres:", nombre[-4:]) # "ores"
```

 Se usa en procesamiento de texto.

4 Reemplazar palabras en una cadena

- ♦ Método `replace()`, cambia palabras dentro de un texto.

```
frase = "Me gusta Java"
```

```
print("Cambio de palabra:", frase.replace("Java", "Python"))
```

 Útil en limpieza de datos.


5 Verificar si una cadena contiene otra (in)

- ♦ Podemos comprobar si una palabra está dentro de otra.

```
print("Python" in frase) # False
```

```
nueva_frase = "Me gusta Python"
```

```
print("Python" in nueva_frase) # True
```

 Se usa en búsquedas.

6 Unir varias palabras en una sola cadena

- ♦ Método `join()` para unir listas en una sola cadena.

```
palabras = ["Hola", "mundo", "Python"]
```

```
print("Frase completa:", " ".join(palabras))
```

 Se usa en generación de texto dinámico.

7 Dividir una cadena en partes

- ♦ Método `split()` para separar una cadena en una lista.

```
oracion = "Python es divertido"

palabras = oracion.split() # ["Python", "es", "divertido"]

print("Lista de palabras:", palabras)
```

 Útil en procesamiento de archivos de texto.

8 Redondear un número decimal

- ♦ Método `round()`, redondea números flotantes.

```
numero = 3.14159

print("Número redondeado:", round(numero, 2)) # 3.14
```

 Usado en cálculos financieros.

9 Formatear números con decimales

- ♦ Método `format()` para mostrar un número con decimales fijos.

```
precio = 19.99


print("Precio con 2 decimales: {:.2f}".format(precio)) # "19.99"
```

 Se usa en reportes y facturas.

10 Obtener el valor ASCII de un carácter

- ♦ Función `ord()` devuelve el valor ASCII.

```
print("Código ASCII de 'A':", ord('A')) # 65
```


 Se usa en criptografía.

11 Elevar un número al cuadrado

- ♦ Operador ****** para calcular potencias.

```
numero = 5
```


```
print("5 elevado al cuadrado:", numero ** 2) # 25
```

 Se usa en matemáticas.

12 Obtener la raíz cuadrada

- ♦ Usamos **** (1/2)** para calcular la raíz cuadrada.

```
print("Raíz cuadrada de 25:", 25 ** 0.5) # 5.0
```

 Alternativa a la función **sqrt()**.


13 División entera y resto

- ♦ División normal **/**, entera **//** y módulo **%**.

```
print("División normal:", 10 / 3) # 3.3333
```

```
print("División entera:", 10 // 3) # 3
```

```
print("Resto:", 10 % 3) # 1
```

 Útil para cálculos matemáticos.

14 Generar un número aleatorio

- ♦ `random.randint()` genera números aleatorios.

```
import random

print("Número aleatorio entre 1 y 10:", random.randint(1, 10))
```

📌 Se usa en juegos y simulaciones.

15 Convertir números a cadenas y viceversa

- ♦ `str()` convierte un número a texto, `int()` convierte texto a número.

```
numero = 100

texto = str(numero) # "100"

print("Convertido a texto:", texto)

cadena = "200"

numero = int(cadena) # 200

print("Convertido a número:", numero)
```

📌 Útil para manipulación de datos.

16 Redondear siempre hacia arriba

- ♦ Usamos `math.ceil()`.

```
import math

print("Redondeo hacia arriba de 3.2:", math.ceil(3.2)) # 4
```

 Se usa en cálculos financieros.


17 Convertir una lista en un conjunto (eliminar duplicados)

♦ `set()` convierte listas en conjuntos.

```
numeros = [1, 2, 2, 3, 4, 4, 5]
```

```
sin_duplicados = set(numeros)
```


```
print("Lista sin duplicados:", sin_duplicados)
```

 Se usa en análisis de datos.

18 Repetir una cadena varias veces

♦ `*` multiplica cadenas.

```
print("Python! " * 3)
```


 Se usa en generación de texto dinámico.

19 Obtener el tipo de una variable

♦ `type()` nos dice el tipo de dato.

```
dato = 3.14
```

```
print("Tipo de dato:", type(dato)) # <class 'float'>
```

 Útil para depuración.

20 Combinar cadenas y variables en un print

- ♦ Usamos `f""` para incluir variables dentro de un texto.

```
nombre = "Mario"
```

```
edad = 30
```

```
print(f"Hola, soy {nombre} y tengo {edad} años.")
```

📌 Se usa en generación de mensajes dinámicos.

IMPORTANTE

La **f** dentro del `print()` indica que estamos utilizando una **f-string** o **formatted string literals**, una forma moderna y eficiente de formatear cadenas en Python.

📌 ¿Qué es una f-string?

Una **f-string** (`f""`) permite insertar variables y expresiones directamente dentro de una cadena de texto, sin necesidad de concatenaciones (+) o métodos como `.format()`.

Ejemplo sin f-string (concatenación tradicional):

```
nombre = "Mario"
```

```
edad = 30
```

```
print("Hola, soy " + nombre + " y tengo " + str(edad) + " años.")
```

❌ Inconvenientes:

- Se deben concatenar cadenas con `+`.
 - Se necesita convertir `edad` a `str()`, ya que no se pueden concatenar strings y números directamente.
-

Ejemplo con f-string (forma más moderna y legible):

```
nombre = "Mario"
```

```
edad = 30
```

```
print(f"Hola, soy {nombre} y tengo {edad} años.")
```

Ventajas:

- Más legible y fácil de escribir.
 - No requiere convertir números a cadenas (`str()`).
 - Se pueden incluir expresiones dentro de `{}`.
-

Aplicación en tu código

```
numero = float(input("Escribe un número decimal: "))
```

```
print(f"Número redondeado: {round(numero, 2)}") # Inserta el número redondeado
```

```
print(f"Cuadrado: {numero ** 2}") # Calcula y muestra el cuadrado
```

```
print(f"Raíz cuadrada: {numero ** 0.5}") # Calcula y muestra la raíz cuadrada
```

♦ ¿Por qué se usa `f""` en este caso?

Porque permite insertar directamente los resultados de `round(numero, 2)`, `numero ** 2` y `numero ** 0.5` dentro del texto sin concatenaciones.

¿Puedo hacer lo mismo sin f-strings?

Sí, pero sería menos intuitivo:


```
print("Número redondeado: " + str(round(numero, 2)))
```

```
print("Cuadrado: " + str(numero ** 2))
```

```
print("Raíz cuadrada: " + str(numero ** 0.5))
```


Problemas:

- Necesitas usar `str()` para convertir valores numéricos en texto.
- Más código y más propenso a errores.

 **Conclusión:** Usa **f-strings** siempre que necesites incluir variables dentro de cadenas. Son más eficientes, legibles y recomendadas en Python moderno.

Ejercicios con estas utilidades combinadas

Enunciados de Ejercicios Prácticos

1 Generador de nombres de usuario

- Pide al usuario su nombre y apellido.
- Genera un nombre de usuario en minúsculas, sin espacios.
- Añade un número aleatorio al final.
- Muestra el nombre de usuario generado.

2 Analizador de frases

- Pide al usuario que ingrese una frase.
- Muestra la cantidad de caracteres de la frase.
- Indica si la frase contiene la palabra "Python".
- Convierte la frase a mayúsculas.
- Muestra la frase invertida.

3 Cálculo de descuentos

- Pide al usuario el precio de un producto.
- Aplica un 15% de descuento.
- Muestra el precio final con dos decimales.
- Muestra el precio redondeado hacia arriba.

4 Generador de etiquetas de productos

- Pide el nombre de un producto y su precio.
- Convierte el nombre del producto a mayúsculas.
- Muestra el precio con dos decimales.
- Genera un código basado en el valor ASCII de la primera letra del producto.

5 Conversión de tipos y manipulación de listas

- Pide al usuario una lista de números separados por comas.
- Convierte cada número a entero.
- Elimina los números repetidos.
- Muestra la lista ordenada sin duplicados.

6 Creación de mensajes personalizados

- Pide al usuario su nombre, edad y ciudad.
- Muestra un mensaje con toda la información.
- Si la edad es menor de 18, redondea hacia arriba hasta la mayoría de edad.

7 Generador de contraseñas aleatorias

- Pide al usuario su nombre.
- Genera una contraseña segura con la primera letra en mayúscula, un número aleatorio y un símbolo especial.
- Muestra la contraseña generada.

8 Verificación de nombres en listas

- Pide al usuario su nombre.
- Verifica si su nombre está en una lista de invitados predefinida.
- Si está en la lista, muestra su posición.

9 Manipulación de nombres



- Pide al usuario su nombre y apellido.
- Convierte el nombre a minúsculas y el apellido a mayúsculas.
- Genera un alias combinando las primeras 3 letras del nombre y del apellido.
- Muestra el alias generado.

10 Formatear y mostrar datos matemáticos

- Pide al usuario un número decimal.
- Muestra el número redondeado a dos decimales.
- Calcula y muestra su cuadrado.
- Calcula y muestra su raíz cuadrada.

RESULTADOS: Ejercicios con estas utilidades combinadas

1 Generador de nombres de usuario

- #  Pide al usuario su nombre y apellido, luego genera un nombre de usuario.
- #  El nombre de usuario será en minúsculas, sin espacios y terminará con un número aleatorio.

```
import random

nombre = input("Escribe tu nombre: ")
apellido = input("Escribe tu apellido: ")

# Convertimos a minúsculas y unimos nombre y apellido sin espacios
nombre_usuario = (nombre + apellido).lower()


# Generamos un número aleatorio para añadirlo al nombre de usuario
numero = random.randint(10, 99)

# Concatenamos todo
usuario_final = f"{nombre_usuario}{numero}"

print(f"Tu nombre de usuario es: {usuario_final}")
```

✓ Usa: `lower()`, `join()`, `random.randint()`, `f""`

2 Analizador de frases




- #  Pide una frase al usuario y muestra:
- # - Cuántos caracteres tiene
- # - Si contiene la palabra "Python"
- # - La frase en mayúsculas
- # - La frase invertida

```
frase = input("Escribe una frase: ")

print("Longitud de la frase:", len(frase))
print("¿Contiene 'Python'?:", "Python" in frase)
print("Frase en mayúsculas:", frase.upper())
print("Frase invertida:", frase[::-1]) # Slicing para invertir
```

✓ Usa: `len()`, `in`, `upper()`, `slicing`

3 Cálculo de descuentos

- #  Pide al usuario un precio y aplica un descuento del 15%.
- #  Luego muestra el precio final formateado con dos decimales.
- #  Además, muestra el valor redondeado hacia arriba.

```
import math




precio = float(input("Escribe el precio del producto: "))

# Calculamos el descuento y el nuevo precio
descuento = precio * 0.15
precio_final = precio - descuento

# Formateamos a dos decimales y redondeamos hacia arriba
print(f"Precio con descuento: ${precio_final:.2f}")
print(f"Redondeado hacia arriba: ${math.ceil(precio_final)}")
```

✓ Usa: `format()`, `math.ceil()`, `*`, `-`

4 Generador de etiquetas de productos

- #  Pide al usuario un nombre de producto y su precio.
- #  Genera una etiqueta con el nombre en mayúsculas, su precio con dos decimales
- #  y un código ASCII basado en la primera letra del producto.



```
producto = input("Nombre del producto: ")
precio = float(input("Precio del producto: "))

etiqueta = f"PRODUCTO: {producto.upper()} - PRECIO: ${precio:.2f} - CÓDIGO: {ord(producto[0])}"

print(etiqueta)
```

✓ Usa: `upper()`, `format()`, `ord()`, `f" "`

5 Conversión de tipos y manipulación de listas

- #  Pide una lista de números separados por comas y convierte cada uno en entero.
- #  Luego elimina los números repetidos y los muestra ordenados.

```
numeros = input("Escribe números separados por comas: ")
```

```
# Convertimos en lista, eliminamos duplicados y ordenamos
lista_numeros = list(set(map(int, numeros.split(","))))

print("Lista sin duplicados y ordenada:", sorted(lista_numeros))
```

✓ Usa: `split()`, `set()`, `map()`, `sorted()`

6 Creación de mensajes personalizados

```
# 📌 Pide al usuario su nombre, edad y ciudad.
# 📌 Muestra un mensaje con toda la información usando f-strings.
# 📌 Si la edad es menor de 18, redondea hacia arriba para calcular la mayoría de edad.
```

```
import math

nombre = input("Escribe tu nombre: ")
edad = int(input("Escribe tu edad: "))
ciudad = input("Escribe tu ciudad: ")

# Calculamos la edad mínima si no es mayor de edad
edad_redondeada = math.ceil(edad / 18) * 18

mensaje = f"Hola {nombre}, tienes {edad} años y vives en {ciudad}. Edad mínima adulta: {edad_redondeada}."

print(mensaje)
```

✓ Usa: `f""`, `math.ceil()`, `int()`, `input()`

7 Generador de contraseñas aleatorias

```
# 📌 Crea una contraseña segura con la primera letra de tu nombre,
# 📌 un número aleatorio, y un símbolo especial.
```

```
import random



nombre = input("Escribe tu nombre: ")

# Generamos una contraseña aleatoria
contraseña = f"{nombre[0].upper()}-{random.randint(100, 999)}-*"
```

```
print(f"Tu nueva contraseña es: {contraseña}")
```

✓ Usa: `upper()`, `random.randint()`, `f"`, `slicing`

8 Verificación de nombres en listas

- #  Pide al usuario su nombre y verifica si está en una lista de invitados.
- #  Muestra su posición en la lista.

```
invitados = ["Mario", "Ana", "Carlos", "Elena", "Pablo"]
```

```
nombre = input("Escribe tu nombre: ")
```

```
if nombre in invitados:
```




```
    print(f"Bienvenido, {nombre}! Estás en la posición {invitados.index(nombre) + 1}.")
```

```
else:
```

```
    print("Lo siento, no estás en la lista.")
```

✓ Usa: `in`, `index()`, `+`

9 Manipulación de nombres

- #  Pide al usuario su nombre y apellido.
- #  Convierte el nombre a minúsculas, el apellido a mayúsculas
- #  y genera un alias combinando las primeras 3 letras de cada uno.

```
nombre = input("Escribe tu nombre: ")
```



```
apellido = input("Escribe tu apellido: ")
```

```
alias = nombre[:3].lower() + apellido[:3].upper()
```

```
print(f"Tu alias es: {alias}")
```

✓ Usa: `lower()`, `upper()`, `slicing`, `+`

10 Formatear y mostrar datos matemáticos

- #  Pide al usuario un número flotante.
- #  Muestra el número redondeado, su cuadrado y su raíz cuadrada.

```
numero = float(input("Escribe un número decimal: "))
```

```
print(f"Número redondeado: {round(numero, 2)}")
```

```
print(f"Cuadrado: {numero ** 2}")
```

```
print(f"Raíz cuadrada: {numero ** 0.5}")
```

✓ Usa: `round()`, `** 2`, `** 0.5`, `float()`