CONTROL en PYTHON

CRISTIAN ANDRES VILLACIS MENDOZA DESARROLLO DE SOFTWARE

link de GITHUB: https://github.com/villacis-cristian/ejercicios_en_clase_python.git

EJERCICIOS

1. Escribe una función que reciba como entrada una *lista* con números y devuelva como resultado una *lista* con los cuadrados de los números contenidos en la lista de entrada.
2. Escribe una función que reciba números como entrada y devuelva la suma de los mismos. La función debe ser capaz de recibir una cantidad indeterminada de números. La función no debe recibir directamente ningún objeto complejo (lista, conjunto, etc.).
3. Escribe una función que reciba un string como entrada y devuelva el string al revés. Ejemplo: si el string de entrada es 'hola', el resultado será 'aloh'.
4. Escribe una función *lambda* que, al igual que la función desarrollada en el ejercicio anterior, invierta el string recibido como parámetro. Ejemplo: si el string de entrada es 'hola', el resultado será 'aloh'.
5. Escribe una función que compruebe si un número se encuentra dentro de un rango específico.
6. Escribe una función que reciba un número entero positivo como parámetro y devuelva una lista que contenga los 5 primeros múltiplos de dicho número. Por ejemplo, si la función recibe el número 3, devolverá la lista [3, 6, 9, 12, 15]. Si la función recibe un parámetro incorrecto (por ejemplo, un múmero menor o igual a cero), mostrará un mensaje de error por pantalla y devolverá una lista vacía.
7. Escribe una función que reciba una lista como parámetro y compruebe si la lista tiene duplicados. La función devolverá *True* si la lista tiene duplicados y *False* si no los tiene.
8. Escribe una función *lambda* que, al igual que la función desarrollada en el ejercicio anterior, reciba una lista como parámetro y compruebe si la lista tiene duplicados. La función devolverá *True* si la lista tiene duplicados y *False* si no los tiene.
9. Escribe una función que compruebe si un string dado es un palíndromo. Un palíndromo es una secuencia de caracteres que se lee igual de izquierda a derecha que de derecha a izquierda. Por ejemplo, la función devolverá *True* si recibe el string "reconocer" y *False* si recibe el string "python".

EJERCICIO 01

In [5]:
```python
def cuadrados_lista(numeros):  # Define una función llamada 'cuadrados_lista' que recibe una lista 'numeros'
    resultado = []  # Crea una lista vacía llamada 'resultado' para guardar los cuadrados
    for num in numeros:  # Itera sobre cada número en la lista de entrada 'numeros'
        resultado.append(num ** 2)  # Calcula el cuadrado del número y lo agrega a 'resultado'
    return resultado  # Devuelve la lista con los cuadrados

# Ejemplo de uso:
lista_original = [1, 2, 3, 4]  # Define una lista de ejemplo
resultado = cuadrados_lista(lista_original)  # Llama a la función con la lista
print("Ejercicio 1:", resultado)  # Imprime el resultado: [1, 4, 9, 16]
```

```
Ejercicio 1: [1, 4, 9, 16]
```

EJERCICIO 02

In [6]:
```python
def suma_numeros(*args):  # Define una función que acepta cualquier cantidad de argumentos (*args)
    total = 0  # Inicializa una variable 'total' en 0 para acumular la suma
    for num in args:  # Itera sobre cada número en 'args' (los argumentos pasados)
        total += num  # Suma cada número al total acumulado
    return total  # Devuelve el resultado final de la suma

# Ejemplo de uso:
resultado = suma_numeros(1, 2, 3, 4, 5)  # Llama a la función con 5 números
print("Ejercicio 2:", resultado)  # Imprime La suma: 15
```

```
Ejercicio 2: 15
```

EJERCICIO 03

In [7]:
```python
def invertir_string(cadena):  # Define una función que recibe un string 'cadena'
    resultado = ""  # Crea un string vacío para almacenar el resultado invertido
    for i in range(len(cadena)-1, -1, -1):  # Itera desde el último carácter hasta el primero
        resultado += cadena[i]  # Concatena cada carácter en orden inverso
    return resultado  # Devuelve el string invertido

# Ejemplo de uso:
texto = "hola"  # Define un string de ejemplo
resultado = invertir_string(texto)  # Llama a la función para invertirlo
print("Ejercicio 3:", resultado)  # Imprime "aloh"
```

```
Ejercicio 3: aloh
```

EJERCICIO 04

In [8]:
```python
invertir_lambda = lambda s: s[::-1]  # Define una función lambda que toma 's' y devuelve 's' invertido usando slicing

# Ejemplo de uso:
texto = "hola"  # String de ejemplo
resultado = invertir_lambda(texto)  # Llama a la lambda pasando 'texto'
print("Ejercicio 4:", resultado)  # Imprime "aloh"
```

```
Ejercicio 4: aloh
```

EJERCICIO 05

In [9]:
```python
def en_rango(numero, inicio, fin):  # Define una función que recibe un número y un rango (inicio, fin)
    return inicio <= numero <= fin  # Retorna True si 'numero' está entre 'inicio' y 'fin' (incluyendo extremos)

# Ejemplo de uso:
numero = 5  # Número a verificar
inicio_rango = 1  # Límite inferior del rango
fin_rango = 10  # Límite superior del rango
resultado = en_rango(numero, inicio_rango, fin_rango)  # Llama a la función
print("Ejercicio 5:", resultado)  # Imprime True (5 está entre 1 y 10)
```

```
Ejercicio 5: True
```

EJERCICIO 06

In [10]:
```python
def primeros_multiplos(n):  # Define una función que recibe un número 'n'
    if n <= 0:  # Si 'n' es negativo o cero...
        print("Error: El número debe ser positivo")  # Muestra un mensaje de error
        return []  # Retorna una lista vacía
    return [n * i for i in range(1, 6)]  # Si 'n' es positivo, genera una lista con los primeros 5 múltiplos

# Ejemplo de uso (caso correcto):
numero = 3  # Número válido
resultado = primeros_multiplos(numero)  # Llama a la función
print("Ejercicio 6 (caso correcto):", resultado)  # Imprime [3, 6, 9, 12, 15]

# Ejemplo de uso (caso incorrecto):
numero_erroneo = -2  # Número inválido
resultado_error = primeros_multiplos(numero_erroneo)  # Llama a la función (imprime error)
print("Ejercicio 6 (caso incorrecto):", resultado_error)  # Imprime [] (lista vacía)
```

```
Ejercicio 6 (caso correcto): [3, 6, 9, 12, 15]
Error: El número debe ser positivo
Ejercicio 6 (caso incorrecto): []
```

EJERCICIO 07

In [11]:
```python
def tiene_duplicados(lista):  # Define una función que recibe una lista
    return len(lista) != len(set(lista))  # Compara la longitud de la lista con su versión en conjunto (sin duplicados)

# Ejemplo de uso:
lista_con_duplicados = [1, 2, 3, 2]  # Lista con duplicados
resultado = tiene_duplicados(lista_con_duplicados)  # Llama a la función
print("Ejercicio 7:", resultado)  # Imprime True (tiene duplicados)
```

```
Ejercicio 7: True
```

EJERCICIO 08

In [12]:
```python
tiene_duplicados_lambda = lambda l: len(l) != len(set(l))  # Lambda que hace lo mismo que la función anterior

# Ejemplo de uso:
lista = [1, 2, 2, 3]  # Lista con duplicados
resultado = tiene_duplicados_lambda(lista)  # Llama a la lambda
print("Ejercicio 8:", resultado)  # Imprime True
```

```
Ejercicio 8: True
```

EJERCICIO 09

In [13]:
```python
def es_palindromo(cadena):  # Define una función que recibe un string 'cadena'
    cadena = cadena.lower()  # Convierte el string a minúsculas para evitar problemas de mayúsculas/minúsculas
    return cadena == cadena[::-1]  # Compara el string original con su versión invertida

# Ejemplo de uso:
palabra = "reconocer"  # Palíndromo
resultado = es_palindromo(palabra)  # Llama a la función
print("Ejercicio 9 (palíndromo):", resultado)  # Imprime True

palabra_no_palindromo = "python"  # No palíndromo
resultado = es_palindromo(palabra_no_palindromo)  # Llama a la función
print("Ejercicio 9 (no palíndromo):", resultado)  # Imprime False
```

```
Ejercicio 9 (palíndromo): True
Ejercicio 9 (no palíndromo): False
```