# ACTIVIDAD 5 LISTAS, RANGE Y RANDOM EN PYTHON

COLIN REYES BRIAN GABRIEL
MATRICULA: 379494
FECHA: 15/09/25

## EJERCICIO 1 Nombres de Mascotas o Artistas Favoritos

### DESCRIPCION

El programa solicita al usuario cuántos nombres va a ingresar (validando que sea un entero entre 5 y 10 y volviendo a pedir en caso de error), crea una lista vacía y, en un bucle, lee cada nombre (eliminando espacios sobrantes con strip) y lo añade con append; luego recorre la lista con enumerate() para obtener índice y nombre, convierte el nombre a mayúsculas con upper() y calcula su longitud con len() para imprimir cada línea en el formato [índice] NOMBRE --> N CARACTERES (por defecto el índice empieza en 0, pero se puede cambiar a 1 sumando 1 al índice al imprimir).

### SOLUCIÓN

```
def main():
    while True:
        try:
            i = int(input("Ingresa el número de mascotas o artistas que añadirás a la lista (5 a 10): "))
        except ValueError:
            print("Por favor ingresa un número entero.")
            continue

        if 5 <= i <= 10:
            break
        else:
            print("Rango incorrecto. Intenta de nuevo.")

    lista = []
    for n in range(i):
        nombre = input(f"Ingresa el nombre número {n}: ").strip()
        lista.append(nombre)

    for idx, nombre in enumerate(lista):
        longitud = len(nombre)
        print(f"[{idx}] {nombre.upper()} --> {longitud} CARACTERES")

if __name__ == "__main__":
    main()
```

```
Ingresa el número de mascotas o artistas que añadirás a la lista (5 a 10): perro
Por favor ingresa un número entero.
Ingresa el número de mascotas o artistas que añadirás a la lista (5 a 10): 6
Ingresa el nombre número 0: perro
Ingresa el nombre número 1: gato
Ingresa el nombre número 2: avion
Ingresa el nombre número 3: movil
Ingresa el nombre número 4: 44
Ingresa el nombre número 5: luis
[0] PERRO --> 5 CARACTERES
[1] GATO --> 4 CARACTERES
[2] AVION --> 5 CARACTERES
[3] MOVIL --> 5 CARACTERES
[4] 44 --> 2 CARACTERES
[5] LUIS --> 4 CARACTERES
```

## EJERCICIO 2 Generación de Números Aleatorios

### DESCRIPCION

La función generar_numeros_aleatorios usa random.sample para obtener una lista de números únicos en el rango especificado (por defecto 10 números entre 30 y 50 inclusive) y valida que el rango sea suficiente para la cantidad pedida; la función imprimir_con_indices recorre la lista con enumerate() e imprime cada elemento junto con su índice en el formato solicitado.

## SOLUCIÓN

```
import random

def generar_numeros_aleatorios(cantidad=10, inicio=30, fin=50):

    if cantidad < 1:
        raise ValueError("La cantidad debe ser al menos 1.")
    if cantidad > (fin - inicio + 1):
        raise ValueError("Rango insuficiente para generar números únicos.")
    return random.sample(range(inicio, fin + 1), cantidad)

def imprimir_con_indices(lista):
    for idx, val in enumerate(lista):
        print(f"[{idx}] {val}")

if __name__ == "__main__":
    numeros = generar_numeros_aleatorios()
    imprimir_con_indices(numeros)
```

```
[0] 39
[1] 36
[2] 34
[3] 43
[4] 50
[5] 46
[6] 38
[7] 45
[8] 40
[9] 30
```

## EJERCICIO 3 Suma de Elementos Correspondientes

## DESCRIPCION

El programa solicita al usuario que introduzca dos listas de números (cada lista en una línea, números separados por espacios o comas), convierte y valida esas entradas (intentando primero convertir a int y si no a float, volviendo a pedir si hay valores no numéricos), luego llama a la función que suma elemento a elemento usando el tamaño de la lista más pequeña (mostrando una advertencia si las longitudes difieren) y finalmente imprime las dos listas y la lista resultado; el proceso incluye manejo de errores con try-except para capturar entradas inválidas o problemas durante la suma.

## SOLUCIÓN

```
def parsear_lista_desde_input(prompt):
    while True:
        linea = input(prompt).strip()
        if not linea:
            print("No ingresaste nada. Intenta de nuevo.")
            continue

        tokens = [t for t in linea.replace(",", " ").split() if t.strip() != ""]
        numeros = []
        try:
            for tok in tokens:
                try:
                    num = int(tok)
                except ValueError:
                    num = float(tok)
                numeros.append(num)
            return numeros
        except ValueError:
            print("La entrada contiene valores no numéricos. Ingresa sólo números separados por espacios o comas.")
```

```
def sumar_elementos_correspondientes(list1, list2):
    try:
        len1 = len(list1)
        len2 = len(list2)
    except TypeError:
        raise TypeError("Ambos argumentos deben ser secuencias con longitud.")
    n = min(len1, len2)
    if len1 != len2:
        print("Advertencia: las listas tienen distinto tamaño. Se usará el tamaño de la lista más pequeña.")
    resultado = []
    for i in range(n):
        try:
            suma = list1[i] + list2[i]
        except Exception as e:
            try:
                suma = float(list1[i]) + float(list2[i])
            except Exception:
                raise TypeError(f"No se pudo sumar los elementos en la posición {i}: {list1[i]!r} + {list2[i]!r}. Error: {e}")
        resultado.append(suma)
    return resultado

if __name__ == "__main__":
    print("Ingresar la primera lista (ejemplo: 1 2 3 4 o 1,2,3,4 o 1 2.5 3):")
    lista1 = parsear_lista_desde_input("Lista 1: ")
    print("Ingresar la segunda lista (mismo formato):")
    lista2 = parsear_lista_desde_input("Lista 2: ")

    try:
        resultado = sumar_elementos_correspondientes(lista1, lista2)
        print("Lista 1:", lista1)
        print("Lista 2:", lista2)
        print("Resultado:", resultado)
    except Exception as err:
        print("Ocurrió un error al sumar las listas:", err)
```

```
Ingresar la primera lista (ejemplo: 1 2 3 4 o 1,2,3,4 o 1 2.5 3):
Lista 1: 4 5 32 5
Ingresar la segunda lista (mismo formato):
Lista 2: 2 42 2 4 4
Advertencia: las listas tienen distinto tamaño. Se usará el tamaño de la lista más pequeña.
Lista 1: [4, 5, 32, 5]
Lista 2: [2, 42, 2, 4, 4]
Resultado: [6, 47, 34, 9]
```

## EJERCICIO 4 Eliminar Duplicados

### DESCRIPCION

El programa solicita al usuario una lista de números enteros (separados por espacios o comas), valida que todos los elementos sean enteros, y la función eliminar_duplicados usa un set como estructura auxiliar para detectar elementos ya vistos y construir una nueva lista que conserva el primer orden de aparición (por ejemplo, [1,2,2,3] → [1,2,3]); si se detecta un elemento no entero se lanza un TypeError y el input helper reintenta hasta obtener una entrada válida.

### SOLUCIÓN

```
def eliminar_duplicados(lista):
    try:
        for el in lista:
            if not isinstance(el, int):
                raise TypeError(f"Elemento no entero detectado: {el!r}")
    except TypeError:
        raise

    resultado = []
    vistos = set()
    for el in lista:
        if el not in vistos:
            vistos.add(el)
            resultado.append(el)
    return resultado
```

```
def parsear_lista_enteros(prompt="Ingresa números enteros separados por espacios o comas: "):
    while True:
        linea = input(prompt).strip()
        if not linea:
            print("No ingresaste nada. Intenta de nuevo.")
            continue
        tokens = [t for t in linea.replace(",", " ").split() if t != ""]
        try:
            numeros = [int(tok) for tok in tokens]
            return numeros
        except ValueError:
            print("Entrada inválida: asegúrate de ingresar sólo números enteros separados por espacios o comas.")


if __name__ == "__main__":
    print("Eliminar duplicados — se conservará el primer orden de aparición.")
    lista = parsear_lista_enteros("Lista original: ")
    try:
        sin_duplicados = eliminar_duplicados(lista)
        print("Lista original:", lista)
        print("Lista sin duplicados:", sin_duplicados)
    except TypeError as err:
        print("Error:", err)
```

```
Eliminar duplicados — se conservará el primer orden de aparición.
Lista original: 2 4 2 43
Lista original: [2, 4, 2, 43]
Lista sin duplicados: [2, 4, 43]
```

## EJERCICIO 5 Media y Mediana

### DESCRIPCION

El programa pide al usuario una lista de números enteros en una sola línea (acepta separadores por espacios o comas), valida que la lista no esté vacía y que todos los elementos sean enteros, luego la función calcular_media_mediana calcula la media como la suma dividida entre la cantidad de elementos y la mediana ordenando la lista y tomando el valor central (o el promedio de los dos centrales si tiene longitud par), ajustando la mediana para que se muestre como entero si resulta exactamente entero; ambos valores se retornan y se imprimen.

### SOLUCIÓN

```
def calcular_media_mediana(lista):
    try:
        n = len(lista)
    except TypeError:
        raise TypeError("Se espera una lista o secuencia con longitud.")
    if n == 0:
        raise ValueError("La lista no puede estar vacía.")
    for el in lista:
        if not isinstance(el, int):
            raise TypeError(f"Elemento no entero detectado: {el!r}")

    media = sum(lista) / n

    ordenada = sorted(lista)
    mitad = n // 2
    if n % 2 == 1:
        mediana = ordenada[mitad]
    else:
        mediana = (ordenada[mitad - 1] + ordenada[mitad]) / 2
        if isinstance(mediana, float) and mediana.is_integer():
            mediana = int(mediana)

    return media, mediana


def parsear_lista_enteros(prompt="Ingresa números enteros separados por espacios o comas: "):
    while True:
        linea = input(prompt).strip()
        if not linea:
```

```
            if not linea:
                print("No ingresaste nada. Intenta de nuevo.")
                continue
            tokens = [t for t in linea.replace(",", " ").split() if t != ""]
            try:
                numeros = [int(tok) for tok in tokens]
                return numeros
            except ValueError:
                print("Entrada inválida: asegúrate de ingresar sólo números enteros (ej. 1 2 3 o 1,2,3).")


    if __name__ == "__main__":
        print("Cálculo de media y mediana (lista de enteros).")
        lista = parsear_lista_enteros("Lista: ")
        try:
            media, mediana = calcular_media_mediana(lista)
            print("Lista:", lista)
            print("Media:", media)
            print("Mediana:", mediana)
        except Exception as e:
            print("Ocurrió un error:", e)
```

```
Cálculo de media y mediana (lista de enteros).
Lista: 1  4 4 3 5 3
Lista: [1, 4, 4, 3, 5, 3]
Media: 3.3333333333333335
Mediana: 3.5
```

## CONCLUSIONES

Estos programas forman un conjunto práctico y coherente para afianzar conceptos básicos de programación en Python: manejo de listas, uso de ciclos for, definición y uso de funciones, validación de entradas y manejo de excepciones; además introducen librerías y estructuras útiles como random y set, y operaciones comunes como cálculo de media y mediana. Cada ejercicio refuerza buenas prácticas (validar entradas, evitar duplicados, preservar orden cuando se necesita, y separar la lógica en funciones reutilizables) y muestra cómo interactuar con el usuario en la consola. Como próximos pasos sugeridos: añadir pruebas automáticas (unit tests), permitir entrada desde archivos, y manejar más tipos numéricos o datos faltantes para hacer los programas más robustos y aplicables a casos reales.

## GITHUB

https://github.com/resu0044/Phyton