Actividad 5: Lista, Range y Random en Python

Nombre: Alejandro Ramirez Cruz

Matricula: 379551

Fecha: Lunes 15 septiembre de 2025

Ejercicio 1: Nombre de mascotas o artistas favoritos

Descripción:

Crear una función que utilice una lista con los nombres de tus mascotas o artistas favoritos (mínimo 5, máximo 10). La función debe imprimir cada nombre junto con la cantidad de caracteres que contiene.

[Explicación del problema]:

1. Crear una lista con nombres de mascotas o artistas favoritos.
2. Recorrer la lista utilizando un ciclo for.
3. Imprimir cada nombre junto con la longitud de la cadena (número de caracteres).

Solución:

Se define una función llamada artistas_favoritos, dentro de la cual se crea una lista con los nombres de varios artistas. Luego, mediante un ciclo for, se recorre cada elemento de la lista y con la función len() se obtiene y muestra la cantidad de caracteres de cada nombre. Finalmente se llama a la función para ejecutarse.

```
def artistas_favoritos():  # Funcion creada para mostrar nombre y total de caracteres
    # Lista de artistas favoritos
    art_favs = ["José José", "Ariel Camacho", "Camilo Sesto", "Virlan Garcia","Los Plebes del Rancho de Ariel Camacho"]

    # Encabezado de salida
    print("=== Nombres de artistas favoritos y el total de caracteres ===")

    # Iterar sobre cada artista en la lista
    for artista in art_favs:
        # Mostrar el nombre del artista y la cantidad de caracteres de su nombre
        print(f"{artista} - {len(artista)} caracteres")
```

```
artistas_favoritos()    # Llamar a la funcion
```

```
=== Nombres de artistas favoritos y el total de caracteres ===
José José - 9 caracteres
Ariel Camacho - 13 caracteres
Camilo Sesto - 12 caracteres
Virlan Garcia - 13 caracteres
Los Plebes del Rancho de Ariel Camacho - 38 caracteres
```

Ejercicio 2: Generación de números aleatorios

Descripción:

Crear dos funciones:

1. Una función que genere y regrese una lista con 10 números aleatorios entre 30 y 50 (sin repetidos).
2. Una función que reciba una lista e imprima cada elemento junto con su índice.

[Explicación del problema]:

1. Utilizar la biblioteca random para generar números aleatorios.
2. Asegurarse de que no haya números repetidos en la lista.
3. Recorrer la lista utilizando un ciclo for.

Solución:

Se importa la biblioteca random para generar números aleatorios. Luego, se define una función llamada num_aleatorios, en la cual se usa random.sample, que devuelve una lista de 10 números únicos dentro del rango de 30 a 50. Posteriormente, se crea la función recibir_lista, que recibe la lista como parámetro y la recorre con un ciclo for utilizando enumerate(). De esta manera se obtiene el par (índice, valor) y se imprime cada elemento junto con su índice correspondiente. Finalmente, se llama a ambas funciones para ejecutar el programa.

```
import random     # Importar la biblioteca 'random' para generar números aleatorios
```

```
def num_aleatorios():    # Funcion creada para generar numeros aleatorios
    # random.sample genera una lista de 10 números sin repetir en el rango de 30 a 50
    numeros = random.sample(range(30, 51), 10)  # El range(30, 51)' incluye del 30 al 50
    return numeros  # Devuelve la lista de números generados

# Funcion que imprime los elementos de una lista mostrando su índice y valor
def recibir_lista(lista_num):
    # enumerate devuelve pares (índice, valor) para iterar sobre la lista
    for i, valor in enumerate(lista_num):
        print(f"[{i}] : {valor}")  # Imprime índice y número correspondiente
```

```
lista = num_aleatorios()   # Genera la lista de números aleatorios

print("Índice y números:")  # Imprime un mensaje

recibir_lista(lista)  # Llama a la funcion recibiendo a la lista
```

```
Índice y números:
[0] : 37
[1] : 34
[2] : 48
[3] : 36
[4] : 45
[5] : 49
[6] : 32
[7] : 33
[8] : 43
[9] : 47
```

Ejercicio 3: Suma de elementos correspondientes

Descripción:

Crear una función que reciba dos listas de números del mismo tamaño y calcule la suma de los elementos correspondientes de cada lista. Si las listas no son del mismo tamaño, usar el tamaño de la lista más pequeña.

[Explicación del problema]:

1. Recibir dos listas como parámetros.
2. Verificar si tienen el mismo tamaño y, si no, trabajar con la más corta.
3. Recorrer ambas listas simultáneamente y sumar elemento a elemento.
4. Guardar los resultados en una nueva lista.
5. Devolver dicha lista con los valores sumados.

Solución:

Se define la función suma_elementos, que recibe dos listas como parámetros. Dentro de un bloque try, se crea una lista local para almacenar los resultados. Se verifica si ambas listas tienen el mismo tamaño y se muestra un mensaje al usuario.

Se determina el tamaño mínimo de las listas y se recorre cada índice hasta ese tamaño, sumando los elementos correspondientes y guardando la suma en lista_suma. Si ocurre algún error durante la ejecución, el bloque except captura la excepción y devuelve una lista vacía. En el bloque principal, se solicita al usuario la cantidad de elementos para cada lista y luego cada elemento individualmente, validando que sean números enteros.

Si el usuario ingresa un valor incorrecto, se captura la excepción ValueError y se inicializan las listas vacías. Finalmente, se llama a la función suma_elementos con las listas ingresadas, se guarda el resultado y se imprimen las listas originales junto con la lista resultante de la suma.

```
def suma_elementos(lista1, lista2):    # Función que recibe y suma dos listas

    try:
        # Lista local para resultados
        lista_suma = []

        # Verifica si las listas tienen el mismo tamaño
        if len(lista1) == len(lista2):
            print("Las listas tienen el mismo tamaño")
        else:
            print("Las listas no tienen el mismo tamaño, se usará la más corta")
```

```
            # Determina el tamaño mínimo entre listas
            tamaño = min(len(lista1), len(lista2))
            print(f"Tamaño a usar: {tamaño}")

            # Itera ambas listas simultáneamente
            for i in range(tamaño):
                # Suma los elementos
                suma = lista1[i] + lista2[i]
                # Guarda la suma en la lista de resultados
                lista_suma.append(suma)

            # Devuelve la lista con resultados
            return lista_suma

        except Exception as e:
            # Captura errores inesperados en la función
            print("Ocurrió un error:", e)
            return []
```

```
# Bloque para pedir datos al usuario
try:
    # Se inicializan las listas vacias
    lista1 = []
    lista2 = []

    # Se pide la cantidad de elementos para cada lista
    elementos_lista1 = int(input("Ingrese el total de elementos de su lista 1: "))
    elementos_lista2 = int(input("Ingrese el total de elementos de su lista 2: "))

    # Se pide los elementos de la primera lista
    for i in range(elementos_lista1):
        elem1 = int(input(f"Ingrese elemento {i+1} de la lista 1: "))
        lista1.append(elem1)

    # Se pide los elementos de la segunda lista
    for i in range(elementos_lista2):
        elem2 = int(input(f"Ingrese elemento {i+1} de la lista 2: "))
        lista2.append(elem2)

except ValueError:
    # Validación de entradas: si no son números enteros
    print("Error: solo se permiten números enteros")
    lista1, lista2 = [], []  # Reinicializa las listas a vacias en caso de error

# Se llama a la función y se guarda el resultado
resultado = suma_elementos(lista1, lista2)

# Imprime las listas originales y el resultado final
print("Lista 1:", lista1)
print("Lista 2:", lista2)
print("Resultado:", resultado)
```

```
Ingrese el total de elementos de su lista 1: 3
Ingrese el total de elementos de su lista 2: 3
Ingrese elemento 1 de la lista 1: 10
Ingrese elemento 2 de la lista 1: 12
Ingrese elemento 3 de la lista 1: 2
Ingrese elemento 1 de la lista 2: 1
Ingrese elemento 2 de la lista 2: 2
Ingrese elemento 3 de la lista 2: 3
Las listas tienen el mismo tamaño
Tamaño a usar: 3
Lista 1: [10, 12, 2]
Lista 2: [1, 2, 3]
Resultado: [11, 14, 5]
```

Ejercicio 4: Eliminar duplicados

Descripción:

Crear una función llamada eliminar_duplicados que reciba una lista como parámetro y elimine los elementos duplicados. El resultado debe ser una nueva lista sin duplicados.

[Explicación del problema]:

1. Solicitar al usario la cantidad de elementos de la lista y los elementos
2. Validar que todos los elementos sean números enteros

3. Eliminar los duplicados de la lista original
4. Manejar errores usando try-except
5. Devolver la lista sin duplicados junto a la lista original

Solución:

Se define la función eliminar_duplicados, que recibe la lista original como parámetro. Dentro de un bloque try, se verifica que todos los elementos sean enteros.

La lista se convierte a un conjunto (set), lo que elimina automáticamente los elementos duplicados. Luego, se convierte el conjunto nuevamente a lista para devolverla. Si ocurre algún error durante la ejecución, el bloque except captura la excepción y devuelve una lista vacía. En el bloque principal, se solicita al usuario el número de elementos y cada elemento individualmente, validando que sean números enteros.

Si el usuario ingresa un valor no entero, se captura la excepción ValueError y se muestra un mensaje de error. Finalmente, se llama a la función eliminar_duplicados con la lista ingresada y se imprime tanto la lista original como la lista resultante sin duplicados.

```
def eliminar_duplicados(lista_original):     # Función que elimina duplicados de una lista de enteros
    try:
        # Validar que todos los elementos sean enteros
        if not all(isinstance(x, int) for x in lista_original):
            raise ValueError("Todos los elementos deben ser números enteros")

        # Convertir la lista a un conjunto para eliminar duplicados
        conjunto_sin_duplicados = set(lista_original)

        # Convertir el conjunto nuevamente a lista
        lista_sin_duplicados = list(conjunto_sin_duplicados)

        # Regresa la lista sin duplicados
        return lista_sin_duplicados

    except Exception as e:
        # Captura errores inesperados en la función
        print("Ocurrió un error:", e)
        return []
```

```
# Bloque para capturar la lista desde el usuario
lista_original = []
try:
    elementos = int(input("Ingrese el total de elementos de la lista: "))
    for i in range(elementos):
        numeros = int(input(f"Ingrese elemento {i+1}: "))
        lista_original.append(numeros) # Agrega los elementos a la lista
except ValueError:
    # Validación de entradas: si no son números enteros
    print("Error: Solo se permiten números enteros")

# Se llama a la función y se guarda el resultado
resultado = eliminar_duplicados(lista_original)

# Imprime las listas originales y el resultado final
print("Lista original:", lista_original)
print("Lista sin duplicados:", resultado)
```

```
Ingrese el total de elementos de la lista: 3
Ingrese elemento 1: 1
Ingrese elemento 2: 1
Ingrese elemento 3: 2
Lista original: [1, 1, 2]
Lista sin duplicados: [1, 2]
```

Ejercicio 5: Media y mediana

Descripción:

Crear una función que calcule la media y la mediana de una lista de números enteros.

[Explicación del problema]:

1. Crear una función que reciba una lista.
2. Calcular la media dividiendo la suma de los elementos entre su longitud.
3. Ordenar la lista y calcular la mediana según su tamaño.
4. Retornar ambos valores.

Solución:

Se creó la función media_y_mediana, que recibe una lista de números ingresados por el usuario. Primero se calcula la media utilizando sum() y len(). Luego, la lista se ordena con sort() para poder calcular la mediana, diferenciando si la cantidad de elementos es par o impar. Finalmente, la función imprime tanto la media como la mediana de los números de la lista.

```
def media_y_mediana(lista_num):     # Función que calcula la media y la mediana de una lista de números

    # Calcular la media usando sum() y len()
    media = sum(lista_num) / len(lista_num)

    # Ordenar la lista para calcular la mediana
    lista_num.sort()

    # Calcular la mediana según si el número de elementos es par o impar
    n = len(lista_num)

    if n % 2 == 0:    # Si es par
        mediana = (lista_num[n // 2 - 1] + lista_num[n // 2]) / 2
    else:       # Si es impar
        mediana = lista_num[n // 2]

    # Imprime los resultados
    print(f"Media de números: {media}")
    print(f"Mediana de números: {mediana}")
```

```
# Bloque para capturar 5 números ingresados por el usuario
lista = []
for i in range(5):
    numeros = int(input(f"Ingrese número {i+1} para la lista: "))
    lista.append(numeros)

# Llamar a la función con la lista
media_y_mediana(lista)
```

```
Ingrese número 1 para la lista: 5
Ingrese número 2 para la lista: 3
Ingrese número 3 para la lista: 2
Ingrese número 4 para la lista: 6
Ingrese número 5 para la lista: 1
Media de números: 3.4
Mediana de números: 3
```