GUÍA 3.2.3:
Guía Ejercicios Resueltos Listas

Sigla	Asignatura	Experiencia de Aprendizaje
FPY1101	Fundamentos de Programación	EA2: Optimizando el código en Python
Tiempo	**Modalidad de Trabajo**	**Indicadores de logro**
2 h	Individual	IL 3.1 al IL 3.3

Antecedentes generales

- Esta guía tiene como objetivo conocer los aspectos generales en Python
- Ser una guía paso a paso para la construcción de programas en Python
- Todos los ejercicios tienen la solución incluida, pero antes de ver la solución, debes resolver por tu cuenta el ejercicio, de esa manera podrás reforzar y aprender. Las soluciones utilízalas para comparar con tus resultados, tomar nota o simplemente como revisión. debes ser consiente y responsable en tu autoaprendizaje.
- Debate con tu docente las respuestas obtenidas, si tienes dudas, recuerda dar aviso y argumentar, los programas no tienen soluciones únicas, teniendo derivaciones o mecanismos distintos de funcionamiento.
- Esta guía puede ser desarrollada en casa, o guiada por el docente, con el fin de seguir un paso a paso y comprender las explicaciones de la o el docente.

Requerimientos para esta actividad

Para el desarrollo de esta actividad deberás disponer de:

- Computador
- Visual Studio Code

Actividad

EJERCICIO 1 Frecuencia de palabras de un texto

Creen un programa que solicite a los usuarios ingresar un texto. Luego, el programa debe analizar el texto y mostrar la frecuencia de cada palabra. Utilicen un diccionario para almacenar las palabras como claves y la frecuencia como valores.

```
# Ejercicio 1: Frecuencia de Palabras en un Texto
texto = input("Ingrese un texto: ")
palabras = texto.split()
```

```
frecuencia_palabras = {}
for palabra in palabras:
    palabra = palabra.lower()
    frecuencia_palabras[palabra] = frecuencia_palabras.get(palabra, 0) + 1

print("Frecuencia de palabras:")
for palabra, frecuencia in frecuencia_palabras.items():
    print(f"{palabra}: {frecuencia}")
```

EJERCICIO 2 Conjunto números primos

ACLARACIÓN: todavía no existe un mecanismo eficiente para obtener todos los números primos, este ejemplo solo es aplicado a un nivel sencillo y básico, y sirve para efectos de explicar el concepto de colecciones.

Desarrollen un programa que genere un conjunto de números primos dentro de un rango específico. Utilicen un conjunto para almacenar los números primos y una función para verificar si un número es primo.

```
# Ejercicio 2: Conjunto de Números Primos
rango_inferior = int(input("Ingrese el rango inferior: "))
rango_superior = int(input("Ingrese el rango superior: "))

numeros_primos = {num for num in range(rango_inferior, rango_superior + 1) if num > 1 and all(num % i != 0
for i in range(2, int(num**0.5) + 1))}

print("Conjunto de números primos:", numeros_primos)
```

Ejercicio Adicional 3: Tuplas y Conjuntos en Combinación

Creen un programa que solicite a los usuarios ingresar nombres y edades. Almacenen estos datos en una lista de tuplas. Luego, utilicen un conjunto para identificar las edades únicas presentes en la lista.

```
# Ejercicio 3: Tuplas y Conjuntos en Combinación
datos_personales = []

while True:
    nombre = input("Ingrese un nombre: ")
    if nombre.lower() == "fin":
        break
    edad = int(input("Ingrese la edad: "))
    datos_personales.append((nombre, edad))

edades_unicas = {edad for _, edad in datos_personales}

print("Edades únicas presentes:", edades_unicas)
```