|  |  |  |
| --- | --- | --- |
| | GUÍA 3.1.5:Guía Ejercicios Resueltos Tuplas, Conjuntos | | |
| 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 |

|  |
| --- |
| **Código QR con relleno sólido**  **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.

|  |
| --- |
| **Lista con relleno sólido Requerimientos para esta actividad** |

## Para el desarrollo de esta actividad deberás disponer de:

## Computador

## Visual Studio Code

|  |  |
| --- | --- |
| **Inteligencia artificial con relleno sólidoActividad** |  |

**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.**

Desarrollar un programa que indique si un número es primo o no dentro de un rango de números . Utilicen un diccionario para almacenar los números primos y una función para verificar si un número es primo. (recuerde, algoritmo no es comprobado para todos los números primos, solo para los primeros números sirve el siguiente algoritmo)

# Ejercicio 2: Números Primos con Diccionarios

def es\_primo(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

rango\_inferior = int(input("Ingrese el rango inferior: "))

rango\_superior = int(input("Ingrese el rango superior: "))

# Crear diccionario para almacenar números primos

numeros\_primos = {num: "Es un número primo" for num in range(rango\_inferior, rango\_superior + 1) if es\_primo(num)}

print("Números primos encontrados:")

for numero, descripcion in numeros\_primos.items():

print(f"{numero}: {descripcion}")

Ejercicio Adicional 3: Tuplas y Conjuntos en Combinación

Creen un programa que solicite a los usuarios ingresar nombres y edades. Almacenen estos datos en un diccionario. Luego, imprima solo las edades.

## Ejercicio 3: Datos Personales con Diccionarios

datos\_personales = {}

while True:

nombre = input("Ingrese un nombre (o 'fin' para terminar): ")

if nombre.lower() == "fin":

break

edad = int(input("Ingrese la edad: "))

datos\_personales[nombre] = edad # Agregar al diccionario

# Identificar edades únicas

edades\_unicas = {edad for edad in datos\_personales.values()}

print("Edades únicas presentes:", edades\_unicas)