|  |  |  |
| --- | --- | --- |
| | GUÍA 3.6.1:Guía Ejercicios Resueltos | | |
| 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 3.5 |

|  |
| --- |
| **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 desarrollarse en casa, o guiada por el docente, para 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:Incorporemos todo en uno en dos matrices escalares**

Crear un programa que permita unir la mayoría de los conocimientos adquiridos durante el semestre. Este algoritmo se llama matrices escalares, y permita multiplicar elementos de una matriz que hemos definido con filas y columnas ingresadas por el usuario. Se debe utilizar sentencia While, For, Matrices, Try Except, validaciones y funciones. Se debe generar un menú de opciones para poder sumar matrices, multiplicar matrices, y un botón salir.

def suma\_matrices(matriz1, matriz2):

if len(matriz1) != len(matriz2) or len(matriz1[0]) != len(matriz2[0]):

raise ValueError("Las matrices deben tener la misma dimensión para sumarse.")

resultado = []

for i in range(len(matriz1)):

fila\_resultado = []

for j in range(len(matriz1[0])):

fila\_resultado.append(matriz1[i][j] + matriz2[i][j])

resultado.append(fila\_resultado)

return resultado

def multiplicacion\_matriz\_escalar(matriz, escalar):

resultado = []

for i in range(len(matriz)):

fila\_resultado = []

for j in range(len(matriz[0])):

fila\_resultado.append(matriz[i][j] \* escalar)

resultado.append(fila\_resultado)

return resultado

def imprimir\_matriz(matriz):

for fila in matriz:

print(fila)

def main():

while True:

print("\nMenú:")

print("1. Sumar matrices")

print("2. Multiplicar matriz por escalar")

print("3. Salir")

try:

opcion = int(input("Ingrese la opción deseada: "))

if opcion == 1:

fila = int(input("Ingrese el número de filas de las matrices: "))

columna = int(input("Ingrese el número de columnas de las matrices: "))

matriz1 = [[int(input(f"Ingrese el elemento ({i+1},{j+1}) de la matriz 1: ")) for j in range(columna)] for i in range(fila)]

matriz2 = [[int(input(f"Ingrese el elemento ({i+1},{j+1}) de la matriz 2: ")) for j in range(columna)] for i in range(fila)]

resultado = suma\_matrices(matriz1, matriz2)

print("\nResultado de la suma:")

imprimir\_matriz(resultado)

elif opcion == 2:

fila = int(input("Ingrese el número de filas de la matriz: "))

columna = int(input("Ingrese el número de columnas de la matriz: "))

matriz = [[int(input(f"Ingrese el elemento ({i+1},{j+1}) de la matriz: ")) for j in range(columna)] for i in range(fila)]

escalar = int(input("Ingrese el valor del escalar: "))

resultado = multiplicacion\_matriz\_escalar(matriz, escalar)

print("\nResultado de la multiplicación por escalar:")

imprimir\_matriz(resultado)

elif opcion == 3:

print("¡Hasta luego!")

break

else:

print("Opción no válida. Por favor, ingrese una opción válida.")

except ValueError as e:

print(f"Error: {e}. Por favor, ingrese valores numéricos válidos.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**EJERCICIO 2: Incorporemos todo en un registro histórico de acciones de un auto**

Crear un programa que permita unir la mayoría de los conocimientos adquiridos durante el semestre. Este programa permitirá guardar los eventos o acciones que se realizan en auto, será una bitácora de todas las acciones que el auto realiza, ejemplo: Se registra como un texto las acciones encender auto, colocar alarma, auto encendido, auto estacionado, auto con nivel de aceite bajo, etc . Se debe utilizar sentencia While, For, listas, Try Except, validaciones, archivo CSV y funciones. Se debe generar un menú de opciones para poder agregar acciones a la bitácora, debe ser un texto, la segunda opción es para ver la bitácora de las acciones hechas en el auto, una tercera opción que guarda la bitácora en un archivo CSV y una cuarta opción para salir.

Reglas de negocio:

* Las acciones se registran con la fecha y hora actual en el formato "dd-mm-yyyy HH:MM:SS", utilizar la librería “from datetime import datetime”
* El usuario debe guardar el archivo CSV con el nombre y extensión
* Debe tener un menú de opciones
* Si el archivo CSV ya existe, se debe sobrescribir

import csv

from datetime import datetime

bitacora = []

def agregar\_evento(evento):

fecha\_hora\_actual = datetime.now().strftime("%d-%m-%Y %H:%M:%S")

fecha\_evento = f"{fecha\_hora\_actual} - {evento}"

bitacora.append(fecha\_evento)

print(f"Evento '{evento}' registrado en la bitácora del auto.")

def historico():

if not bitacora:

print("La bitácora está vacía.")

else:

print("Bitácora del auto:")

for evento in bitacora:

print(evento)

def guardar\_registro\_csv(nombre\_archivo):

try:

with open(nombre\_archivo, mode='w', newline='') as archivo\_csv:

writer = csv.writer(archivo\_csv)

writer.writerow(['Bitácora del Auto'])

for evento in bitacora:

writer.writerow([evento])

print(f"Registro guardado en '{nombre\_archivo}' con éxito.")

except Exception as e:

print(f"Error al guardar el archivo: {e}, coloque un nombre y extensión al archivo")

def menu():

while True:

print("\nMenú de registro histórico de acciones en el Auto:")

print("1. Agregar acción")

print("2. Verificar histórico")

print("3. Guardar en archivo CSV")

print("4. Salir")

try:

opcion = int(input("Ingrese la opción deseada: "))

if opcion == 1:

print("Escriba la acción (EJ: encender auto, viaje de Puerto Montt a Santiago")

print("apagado auto, colocar alarma)")

evento = input("Escriba la acción: ")

agregar\_evento(evento)

elif opcion == 2:

historico()

elif opcion == 3:

nombre\_archivo = input("Ingrese el nombre del archivo CSV para guardar el registro ejemplo: 'bitacoraAuto.csv': ")

guardar\_registro\_csv(nombre\_archivo)

elif opcion == 4:

print("saliendo de la bitácora")

break

else:

print("Opción no válida. Por favor, ingrese una opción válida.")

except ValueError as e:

print(f"Error: {e}. Por favor, ingrese un valor numérico válido.")

if \_\_name\_\_ == "\_\_main\_\_":

menu()