Proyecto: Pensamiento Computacional	**Docente:** Guillermo Carmona González
Equipo de Trabajo	**Integrantes:**
	Juan José Chaparro Gutiérrez
	Tomás Gutiérrez Velásquez
	Amerie Sugimoto Rivera

Objetivo general

Fortalecer y evaluar las habilidades básicas del pensamiento computacional a través del desarrollo un sistema en Python, asegurando que cada miembro del equipo pueda modificar y dar respuesta sobre el código en tiempo real durante la presentación.

Objetivos específicos

- Aplicar los pilares del pensamiento computacional en la resolución de problemas prácticos.
- Implementar y dominar las estructuras de control estudiadas en clase, incluyendo: entradas y salidas de datos, condicionales simples y múltiples, ciclos while y for, manejo de listas, creación y uso de funciones, manejo de archivos y creación de gráficas.
- Desarrollar habilidades en la creación, depuración y modificación de código, garantizando la capacidad de responder preguntas y hacer ajustes inmediatos durante la presentación del proyecto.

Presentación del Proyecto

Los estudiantes deben estar en la capacidad de demostrar su comprensión y dominio del proyecto mediante actividades prácticas como:

- **Modificación de código existente:** Modificar el código del sistema actual en tiempo real durante la presentación.
- **Desarrollo de nuevas funcionalidades del sistema:** Implementar nuevas opciones, utilizando funciones y estructuras algorítmicas adecuadas, con capacidad de explicación en tiempo real.
- **Control de errores y mejoras en el sistema:** Asegurar que las acciones del sistema no generen errores lógicos, como evitar valores negativos o superar topes predefinidos. Mejorar la visualización de los datos para una mejor experiencia de usuario.

- **Fomento de la creatividad mediante la programación:** Incorporar acciones creativas y originales que añadan elementos novedosos al sistema, promoviendo la resolución de problemas de manera innovadora.

Versión FINAL del Proyecto

Siga las siguientes actividades para finalizar el proyecto:

1. Modificar las variables de categoría a una variable tipo lista: La versión inicial usa 4 variables, por ejemplo: `clientesZona1=10`, `clientesZona2=30`, `clientesZona3=20`, `clientesZona4=45`. En la versión final, las variables deben representarse mediante una lista, ejemplo: `clientesZonas=[10,30,20,45]`, donde la posición 0, 1, 2 y 3 representa los clientes de la zona 1, 2, 3 y 4, respectivamente. Los valores se deben solicitar al usuario y asignarse a las posiciones correspondientes en la lista.

2. Convertir las opciones en funciones: Organice el código de cada opción en una función independiente.

3. Completar las funcionalidades del sistema: Si en la versión anterior no se implementaron todas las funcionalidades, agregue las opciones restantes, asegurándose de que cada una esté en una función y sea invocada desde el menú de opciones.

4. Mejorar la presentación de la consulta de datos: Modifique la opción que permite consultar el estado de las variables para mostrar los datos en forma de cuadrícula, como se muestra a continuación:

```
-----------------
Z1: 10 | Z2: 30
-----------------
Z3: 20 | Z4: 45
-----------------
```

Z1: 10 representaría los clientes de la zona 1, el *Z2: 30*, los clientes de la zona 2, y así sucesivamente.

5. Control de actualización de datos con topes mínimos o máximos: Tome alguna de las funciones que permite actualizar valores. Implemente validaciones para asegurar que los valores no sean negativos o superen un valor máximo específico. Por ejemplo, la suma de porcentajes no puede ser mayor a 100.

6. Actualización masiva: Agregue una función que duplique el valor de todas las categorías. Es decir, que los valores de cada posición de la lista se dupliquen. Para esta acción debe utilizar un ciclo `for`.

7. Carga de datos desde un archivo: Agregue una función para actualizar los datos de la lista desde un archivo de texto. Ejemplo: el archivo `datos.txt` contiene `15,25,35,45`, se debe eer el archivo y actualizar la lista con estos valores.

8. Gráfica: Agregue una función para mostrar los datos en forma de gráfica (barras, dispersión, etc.) utilizando la librería Matplotlib.

Entrega

Envíe el documento con el enlace del proyecto y el código al Buzón de Interactiva Virtual o donde el docente le indique. El docente también le indicará la fecha de entrega y presentación del proyecto.

Enlace del proyecto:

https://colab.research.google.com/drive/1OyVYmkyiQ4cv_nsKAepOdORZYykmuHxH?usp=sharing

Código:

```
# Proyecto de Gestión de Presupuesto de Marketing

# Elaborado por:
"""
Juan José Chaparro Gutiérrez
Tomás Gutiérrez Velásquez
Amerie Sugimoto Rivera
"""

from tabulate import tabulate
import matplotlib.pyplot as plt


# Lista de presupuestos y nombres de regiones
presupuestos = [0, 0, 0, 0]
regiones = ["Norteamérica", "Europa", "Latinoamérica", "Asia"]

# 1. Ingresar presupuesto inicial
def ingresar_presupuesto():
    for r in range(4):
        print("Ingrese el presupuesto para", regiones[r])
        valor = float(input())
```

```
        if valor < 0 or valor > 1000000:
            print("Presupuesto inválido. Debe estar entre 0 y
1,000,000.")
            return
        presupuestos[r] = valor

# 2. Mostrar presupuestos actuales (con tabulate)
def mostrar_presupuesto_inicial():
    tabla = []
    for r in range(4):
        tabla.append([regiones[r], presupuestos[r]])
    print(tabulate(tabla, headers=["Región", "Presupuesto"],
tablefmt="grid"))

# 3. Modificar presupuesto de una región (con control de tope)
def mod_presupuesto_region():
    print("Seleccione la región a modificar:")
    print("1. Norteamérica")
    print("2. Europa")
    print("3. Latinoamérica")
    print("4. Asia")
    opcion = int(input("Ingrese la opción: "))

    if 1 <= opcion <= 4:
        nuevo = float(input(f"Ingrese el nuevo presupuesto para
{regiones[opcion - 1]}: "))
        if nuevo < 0 or nuevo > 1000000:
            print("Presupuesto inválido. Debe estar entre 0 y
1,000,000.")
        else:
            presupuestos[opcion - 1] = nuevo
    else:
        print("Opción inválida.")

# 4. Redistribuir presupuesto equitativamente
def redistribuir_presupuesto_equi():
    suma = sum(presupuestos)
    nuevo = suma / 4
    for r in range(4):
        presupuestos[r] = nuevo
    print("Presupuesto redistribuido en partes iguales.")

# 5. Simular campaña con tope máximo
def simulacion_campana():
    print("Seleccione la región para la campaña:")
```

```
    print("1. Norteamérica")
    print("2. Europa")
    print("3. Latinoamérica")
    print("4. Asia")
    region = int(input("Ingrese la opción: "))
    aumento = 10000

    if region >= 1 and region <= 4:
        nueva = presupuestos[region - 1] + aumento
        suma_total = 0
        for r in range(4):
            if r == region - 1:
                suma_total += nueva
            else:
                suma_total += presupuestos[r]

        if suma_total > 100000:
            print("No se puede aumentar. Se pasa del tope máximo.")
        else:
            presupuestos[region - 1] = nueva
            print("Se aumentó el presupuesto de", regiones[region -
1])
    else:
        print("Opción inválida.")

# 6. Clasificar presupuesto total
def clasificar_presupuesto():
    suma = sum(presupuestos)

    if suma < 100000:
        print("Presupuesto Bajo")
    elif suma <= 300000:
        print("Presupuesto Equilibrado")
    else:
        print("Presupuesto Alto")

# 7. Cargar presupuestos desde archivo
def cargar_desde_archivo():
    try:
        archivo = open("datos.txt", "r")
        datos = archivo.read().split(",")
        for i in range(4):
            presupuestos[i] = float(datos[i])
        archivo.close()
```

```
        print("Presupuestos cargados desde archivo.")
    except:
        print("Error al leer el archivo. Asegúrese de que
'datos.txt' exista y tenga 4 valores separados por coma.")

# 8. Graficar presupuestos con Matplotlib
def graficar_presupuestos():
    plt.bar(regiones, presupuestos, color='lightblue')
    plt.title("Presupuesto por Región")
    plt.xlabel("Regiones")
    plt.ylabel("Presupuesto")
    for i in range(len(presupuestos)):
        plt.text(i, presupuestos[i], str(presupuestos[i]))
    plt.tight_layout()
    plt.show()


# Menú principal
while True:
    print("\n---- MENÚ DE OPCIONES ----")
    print("1. Ingresar presupuesto inicial")
    print("2. Mostrar presupuesto actual")
    print("3. Modificar presupuesto de una región")
    print("4. Redistribuir presupuesto equitativamente")
    print("5. Simular campaña de marketing")
    print("6. Clasificar presupuesto")
    print("7. Cargar presupuestos desde archivo")
    print("8. Graficar presupuestos")
    print("9. Salir")
    print("--------------------------")

    opcion = int(input("Ingrese una opción: "))

    if opcion == 1:
        ingresar_presupuesto()
    elif opcion == 2:
        mostrar_presupuesto_inicial()
    elif opcion == 3:
        mod_presupuesto_region()
    elif opcion == 4:
        redistribuir_presupuesto_equi()
    elif opcion == 5:
        simulacion_campana()
    elif opcion == 6:
        clasificar_presupuesto()
```

```
    elif opcion == 7:
        cargar_desde_archivo()
    elif opcion == 8:
        graficar_presupuestos()
    elif opcion == 9:
        print("Saliendo del sistema...")
        break
    else:
        print("Opción no válida.")
```