Proyecto Final – Pensamiento Computacional – Docente: Nombre docente

|  |  |
| --- | --- |
| **Equipo de Trabajo** |  |
|  |
|  |

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.

**Nota:** Para el desarrollo de este proyecto solo puede utilizar las estructuras de programación y conceptos explicados en clase. En caso de utilizar estructuras diferentes, el estudiante debe demostrar dominio de estas y estar preparado para explicarlas.

**Sugerencia:** Recuerde apoyarse en monitorías si tiene dudas, o preguntarle al docente.

Versión final del Proyecto

En la entrega 2 se realizó el código inicial en Python, el cual podía incluir el desarrollo parcial o total con las funcionalidades del sistema. Ese código contenía 4 variables principales y un menú que implementaba todas las funcionalidades.

En la versión final, las variables principales deben estar almacenadas en una lista, y cada opción del sistema debe estar implementada en una función independiente. Considere si requiere usar parámetros y retorno de datos.

A continuación, un ejemplo del proyecto IMC, que muestra cómo implementar listas y funciones de acuerdo con lo solicitado para la versión final.

|  |
| --- |
| # Proyecto Monitoreo del Índice de Masa Corporal (IMC)  # Elaborado por:  """  Nombre 1  Nombre 2  Nombre 3  """  # Librería para generar números aleatorios  import random  # Librería para graficar  import matplotlib.pyplot as plt  """  Lista de 2 posiciones.  La posición 0 de la lista representa el imcJovenes  La posición 0 de la lista representa el imcAdultos  """  listaIMC = [0, 0]  # Funciones para cada una de las opciones del proyecto.  # Opción 1: Inicializar valores de IMC → Permite ingresar el IMC promedio inicial de cada grupo.  def ingresarValores():  """  Versión inicial para ingresar los valores  imcJovenes = int(input("Ingrese el IMC promedio de los jóvenes: "))  imcAdultos = int(input("Ingrese el IMC promedio de los adultos: "))  listaIMC[0] = imcJovenes  listaIMC[1] = imcAdultos  """  # versión actualizada para ingresar los valores con control de valores negativos  imcJovenes = int(input("Ingrese el IMC promedio de los jóvenes: "))  imcAdultos = int(input("Ingrese el IMC promedio de los adultos: "))  if imcJovenes < 0 or imcAdultos < 0:  print("Error: No se permiten valores negativos. No se actualizaron los datos.")  else:  listaIMC[0] = imcJovenes  listaIMC[1] = imcAdultos  # Opción 2: Consultar el estado del IMC en todos los grupos → Muestra los valores actuales registrados.  def consultarValores():  """  Versión inicial para mostrar los valores  print("El IMC promedio de los jóvenes es: "+ str(listaIMC[0]))  print("El IMC promedio de los adultos es: " + str(listaIMC[1]))  """  # Versión actualizada para mostrar los datos  print("----------------------")  print("JO: " + str(listaIMC[0]) + " | AD: " + str(listaIMC[1]))  print("----------------------")  # Opción 3: Actualizar el IMC de un grupo específico → Permite modificar el IMC promedio de un grupo seleccionado.  def actualizarValoresGrupoEspecifico():  print("Menú de grupos:")  print("1) Jóvenes")  print("2) Adultos")  grupo = int(input("Seleccione el grupo: "))  if grupo == 1:  imcJovenes = int(input("Ingrese el nuevo IMC promedio de los jóvenes: "))  listaIMC[0] = imcJovenes  elif grupo == 2:  imcAdultos = int(input("Ingrese el nuevo IMC promedio de los adultos: "))  listaIMC[1] = imcAdultos  else:  print("Grupo incorrecto")  # Opción 4: Ajuste global del IMC → Permite aumentar en una unidad los valores de IMC promedio de cada grupo.  def actualizarValores():  imcJovenes = listaIMC[0] + 1  imcAdultos = listaIMC[1] + 1  listaIMC[0] = imcJovenes  listaIMC[1] = imcAdultos  # Opción 5: Cambio aleatorio del IMC en un grupo → Se disminuye en una unidad el IMC promedio en un grupo seleccionado al azar.  def actualizarValorGrupoAleatorio():  # Calcula un número aleatorio entre 1 y 2, porque hay 2 grupos poblacionales  aleatorio = random.randint(1, 2)  if aleatorio == 1:  imcJovenes = listaIMC[0] - 1  listaIMC[0] = imcJovenes  elif aleatorio == 2:  imcAdultos = listaIMC[1] - 1  listaIMC[1] = imcAdultos  """  Opción 6: Clasificación del estado nutricional → El sistema categoriza y muestra el IMC de un grupo poblacional en:  • Bajo peso: IMC menor a 18.5  • Normal: IMC entre 18.5 y 24.9  • Sobrepeso: IMC mayor a 25  """  def clasificar():  print(" ")  print("Menú de grupos:")  print("1) Jóvenes")  print("2) Adultos")  grupo = int(input("Seleccione el grupo: "))  if grupo == 1:  if listaIMC[0] < 18.5:  print("Jóvenes con IMC en Bajo peso")  elif listaIMC[0] >= 18.5 and listaIMC[0] <= 24.9:  print("Jóvenes con IMC Normal")  else:  print("Jóvenes con IMC en Sobrepeso")  elif grupo == 2:  if listaIMC[1] < 18.5:  print("Adultos con IMC en Bajo peso")  elif listaIMC[1] >= 18.5 and listaIMC[1] <= 24.9:  print("Adultos con IMC Normal")  else:  print("Adultos con IMC en Sobrepeso")  else:  print("Grupo incorrecto")  # (NUEVA FUNCIÓN) Actualización masiva → Permite duplicar los valores de IMC promedio de cada grupo.  def actualizarMasivamente():  for i in range(len(listaIMC)):  duplicado = listaIMC[i] \* 2  listaIMC[i] = duplicado  # (NUEVA FUNCIÓN) Actualizar desde archivo → Actualiza los valores de la lista desde un archivo.  def actualizarDesdeArchivo():  archivo = open("datos.txt")  contenido = archivo.read()  lista = contenido.split(",")  listaIMC[0] = float(lista[0])  listaIMC[1] = float(lista[1])  archivo.close()  # (NUEVA FUNCIÓN) Graficar → Muestra los valores de la lista en forma de gráfico.  def graficar():  x = ["Jóvenes", "Adultos"]  y = listaIMC  plt.bar(x, y)  plt.xlabel("Grupos")  plt.ylabel("IMC")  plt.title("Gráfico de IMC")  plt.show()  # Ejecución del programa e invocación de funciones.  while True:  print(" ")  print("Menú de opciones:")  print("1) Inicializar valores de IMC")  print("2) Consultar el estado del IMC en todos los grupos")  print("3) Actualizar el IMC de un grupo específico")  print("4) Ajuste global del IMC")  print("5) Cambio aleatorio del IMC en un grupo")  print("6) Clasificación del estado nutricional")  print("7) Actualizar masivamente los valores de IMC")  print("8) Actualizar IMC desde archivo")  print("9) Graficar")  print("10) Salir")  opcion = int(input("Seleccione una opción: "))  if opcion == 1:  ingresarValores()  elif opcion == 2:  consultarValores()  elif opcion == 3:  actualizarValoresGrupoEspecifico()  elif opcion == 4:  actualizarValores()  elif opcion == 5:  actualizarValorGrupoAleatorio()  elif opcion == 6:  clasificar()  elif opcion == 7:  actualizarMasivamente()  elif opcion == 8:  actualizarDesdeArchivo()  elif opcion == 9:  graficar()  elif opcion == 10:  print("Chao!")  break  else:  print("Opción incorrecta") |

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:

|  |
| --- |
| Copie en este espacio el vínculo a la carpeta de proyecto en Colab, en el cual se cuencuentra el código y el archivo de datos |

Código:

|  |
| --- |
| Copie en este espacio el código Python del proyecto |