Universidad Politécnica de Texoco (UPTex)

Manual de Usuario

Programa: IMC convertido a .exe

Autor:

Villegas de la Cruz Grace Itzel

Texcoco, México, 2024

Contenido

Contenido de imágenes

1. Objetivos

El presente manual tiene como objetivo describir el funcionamiento de un programa en Python con una interfaz gráfica de usuario (GUI) para calcular el Índice de Masa Corporal (IMC). Este documento incluye una descripción general del programa, una descripción detallada de los archivos necesarios para su funcionamiento, y la interpretación del proceso de conversión a un archivo ejecutable.

Objetivos Específicos del Programa:

1. **Desarrollar una Interfaz Gráfica Intuitiva**:
    - Implementar una interfaz gráfica de usuario utilizando Tkinter para permitir a los usuarios ingresar fácilmente sus datos personales, como altura y peso.
2. **Calcular el Índice de Masa Corporal (IMC)**:
    - Crear una función para calcular el IMC usando la fórmula estándar:

$$IMC = \frac{peso\ (kg)}{(altura\ (m))^2}$$

        - Asegurar que la función pueda manejar entradas en metros y centímetros, realizando la conversión según sea necesario.
3. **Validación de Datos y Manejo de Errores**:
    - Implementar validación de datos para asegurar que los usuarios ingresen valores válidos y proporcionar retroalimentación inmediata en caso de errores.
4. **Guardar Resultados en Archivos CSV**:
    - Implementar una funcionalidad para guardar los resultados del IMC junto con los datos personales en un archivo CSV para facilitar el análisis posterior.
5. **Incorporar Elementos Visuales Atractivos**:
    - Mejorar la experiencia del usuario añadiendo elementos visuales como imágenes y gráficos en la interfaz.

6. **Documentar el Proceso de Instalación y Uso**:
    - Crear una guía de instalación y uso detallada para que los usuarios puedan instalar y utilizar la aplicación sin dificultades.
7. **Convertir la Aplicación a un Archivo Ejecutable**:
    - Utilizar herramientas como auto-py-to-exe para convertir el programa Python en un archivo ejecutable (.exe), asegurando que la aplicación pueda ser ejecutada en sistemas Windows sin necesidad de tener Python instalado.

2. Introducción

El programa para calcular el IMC está diseñado en Python y consta de varias fases clave que aseguran su correcto funcionamiento. Estas fases son:

1. Fase de Entrada de Datos
2. Fase de Cálculo del IMC.
3. Fase de Validación y Almacenamiento de Resultados:

1.1 Fase de entrada de datos.

Los usuarios ingresan sus datos personales (altura y peso) a través de una interfaz gráfica de usuario (GUI) desarrollada con Tkinter. (Imagen 1. Ingresar datos del usuario)

Imagen 1. Ingresar datos del usuario

1.2 Fase de cálculo del IMC

El programa calcula el IMC utilizando la fórmula estándar y maneja la conversión de unidades según sea necesario. (Imagen 2. Botón para calcular IMC)

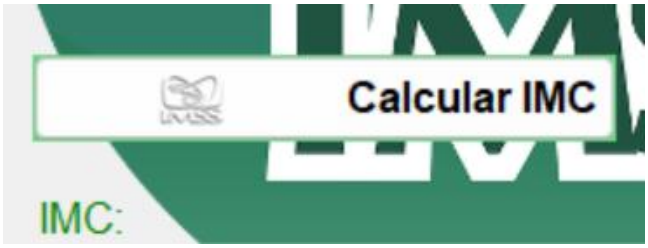

Imagen 2. Botón para calcular IMC

1.3 Fase de validación y almacenamiento de resultados

Se valida la entrada de datos, se maneja cualquier error potencial y se guardan los resultados en un archivo CSV para su análisis posterior. (Imagen 3. Botón para guardar los datos en un CSV)

Imagen 3. Botón para guardar los datos en un CSV

2. Estructura del Programa

2.1 Componentes Principales

El programa está compuesto por varios archivos clave que incluyen la lógica principal, la configuración y el manejo de datos. Los componentes principales son:

1. **main.py**:
    - Contiene la lógica principal de la aplicación, incluyendo la interfaz gráfica y las funciones de cálculo del IMC.

2. **config.py**:
    - Gestiona configuraciones adicionales y la conversión de unidades.
3. **Archivos de Datos**:
    - Archivos CSV utilizados para almacenar los resultados del IMC..

3. Código Fuente

3.1 Descripción del Código

El código fuente está escrito en Python y está organizado en módulos para facilitar su mantenimiento y ampliación.

3.2 Código del Archivo main.py

```
#Se importaron las bibliotecas y módulos necesarios
import tkinter as tk
from tkinter import ttk, messagebox
from PIL import ImageTk, Image
import csv
import os
#Definición de la función calcular_imc
def calcular_imc():
    try:
#Obtener y convertir los valores de entrada
        peso = float(entry_peso.get())
        altura = float(entry_altura.get())
        edad = int(entry_edad.get())
        sexo = var_sexo.get()
        nombre = entry_nombre.get()

        if not nombre:
            messagebox.showerror("Error", "Por favor ingrese el nombre del paciente.")
            return

        ks = 1.0 if sexo == "Hombre" else 1.1
        ka = 1 + 0.01 * (edad - 25)
        imc = (peso / (altura ** 2)) * ks * ka

```

```
28.         label_resultado.config(text=f"IMC: {imc:.2f}")
29.
30.         # Mensajes según el rango de IMC
31.         if imc < 18.5:
32.             messagebox.showinfo("Estado de IMC", "IMC indica
   desnutrición.")
33.         elif imc >= 18.5 and imc < 24.9:
34.             messagebox.showinfo("Estado de IMC", "IMC en rango
   saludable.")
35.         elif imc >= 24.9 and imc < 29.9:
36.             messagebox.showinfo("Estado de IMC", "IMC indica
   sobrepeso.")
37.         else:
38.             messagebox.showinfo("Estado de IMC", "IMC indica
   obesidad.")
39.
40.     except ValueError:
41.         messagebox.showerror("Error", "Por favor ingrese valores
   válidos.")
42.
43. def guardar_datos():
44.     nombre = entry_nombre.get()
45.     if not nombre:
46.         messagebox.showerror("Error", "Por favor ingrese el nombre del
   paciente.")
47.         return
48.
49.     datos = {
50.         "Nombre": entry_nombre.get(),
51.         "Peso (kg)": entry_peso.get(),
52.         "Altura (m)": entry_altura.get(),
53.         "Edad": entry_edad.get(),
54.         "Sexo": var_sexo.get(),
55.         "IMC": label_resultado.cget("text").split(": ")[1]
56.     }
57.
58.     archivo = f"{nombre}.csv"
59.     archivo_existe = os.path.isfile(archivo)
60.
61.     with open(archivo, mode='a', newline='') as file:
62.         writer = csv.DictWriter(file, fieldnames=datos.keys())
63.         if not archivo_existe:
64.             writer.writeheader()
65.         writer.writerow(datos)
66.
```

```
67.     messagebox.showinfo("Guardado", f"Datos guardados en el archivo
   {archivo}")
68.
69.# Función para configurar el cursor como hand2 cuando el ratón entra
   en un botón
70.def configurar_cursor_hand2(event):
71.     event.widget.config(cursor="hand2")
72.
73.# Crear la ventana principal
74.root = tk.Tk()
75.root.title("Calculadora de IMC")
76.
77.# Crear un canvas con la imagen de fondo
78.canvas = tk.Canvas(root, width=600, height=400)
79.canvas.pack()
80.
81.# Título "IMSS" centrado
82.titulo_imss = tk.Label(canvas, text="IMSS", font=("Helvetica", 24,
   "bold"), fg="green")
83.titulo_imss.place(relx=0.5, y=20, anchor=tk.CENTER)
84.
85.imagen_fondo = ImageTk.PhotoImage(Image.open("IMSS-logo.png"))
86.canvas.create_image(0, 0, anchor=tk.NW, image=imagen_fondo)
87.
88.# Variables
89.var_sexo = tk.StringVar(value="Hombre")
90.
91.# Estilo para los botones redondeados
92.estilo = ttk.Style()
93.estilo.configure('BotonRedondo.TButton', borderwidth=5,
   bordercolor="#2CBF4E", background="#6ED185",
94.                 foreground="black", padx=30, pady=30,
   relief=tk.RAISED,
95.                 font=("Helvetica", 12, "bold"))
96.
97.# Widgets sobre el canvas
98.tk.Label(canvas, text="Nombre:", font=("Helvetica", 12),
   justify=tk.CENTER, foreground="green").place(x=50, y=80)
99.entry_nombre = tk.Entry(canvas, width=25, font=("Helvetica", 12))  #
   Ajuste en el ancho del Entry
100.     entry_nombre.place(x=200, y=80)
101.
102.     tk.Label(canvas, text="Peso (kg):", font=("Helvetica", 12),
   justify=tk.CENTER, foreground="green").place(x=50, y=130)
```

```
103.     entry_peso = tk.Entry(canvas, width=15, font=("Helvetica",
   12))  # Ajuste en el ancho del Entry
104.     entry_peso.place(x=200, y=130)
105.
106.     tk.Label(canvas, text="Altura (m):", font=("Helvetica", 12),
   justify=tk.CENTER, foreground="green").place(x=50, y=180)
107.     entry_altura = tk.Entry(canvas, width=15, font=("Helvetica",
   12))  # Ajuste en el ancho del Entry
108.     entry_altura.place(x=200, y=180)
109.
110.     tk.Label(canvas, text="Edad:", font=("Helvetica", 12),
   justify=tk.CENTER, foreground="green").place(x=50, y=230)
111.     entry_edad = tk.Entry(canvas, width=15, font=("Helvetica",
   12))  # Ajuste en el ancho del Entry
112.     entry_edad.place(x=200, y=230)
113.
114.     tk.Label(canvas, text="Sexo:", font=("Helvetica", 12),
   justify=tk.CENTER, foreground="green").place(x=50, y=280)
115.     tk.Radiobutton(canvas, text="Hombre", variable=var_sexo,
   value="Hombre", font=("Helvetica", 10)).place(x=200, y=280)
116.     tk.Radiobutton(canvas, text="Mujer", variable=var_sexo,
   value="Mujer", font=("Helvetica", 10)).place(x=270, y=280)
117.
118.     # Cargar la imagen para el botón de calcular IMC
119.     image_calcular_imc = Image.open("imss_bot.jpeg")
120.     image_calcular_imc = image_calcular_imc.resize((110, 23),
   Image.LANCZOS)
121.     image_calcular_imc = ImageTk.PhotoImage(image_calcular_imc)
122.
123.     # Crear el botón con la imagen y configurar el cursor hand2
124.     btn_calcular_imc = ttk.Button(canvas, text="Calcular IMC",
   image=image_calcular_imc, style='BotonRedondo.TButton',
   command=calcular_imc, compound=tk.LEFT)
125.     btn_calcular_imc.place(x=50, y=330)
126.     btn_calcular_imc.bind("<Enter>", configurar_cursor_hand2)
127.     btn_calcular_imc.bind("<Leave>", lambda e:
   btn_calcular_imc.config(cursor=""))
128.
129.     # Crear el botón "Guardar Datos" y configurar el cursor hand2
130.     btn_guardar_datos = ttk.Button(canvas, text="Guardar Datos",
   style='BotonRedondo.TButton', command=guardar_datos)
131.     btn_guardar_datos.place(x=300, y=330)
132.     btn_guardar_datos.bind("<Enter>", configurar_cursor_hand2)
133.     btn_guardar_datos.bind("<Leave>", lambda e:
   btn_guardar_datos.config(cursor=""))
```

```
134.
135.     label_resultado = tk.Label(canvas, text="IMC: ", font=("Arial",
   12), justify=tk.CENTER, foreground="green")
136.     label_resultado.place(x=50, y=380)
137.
138.     # Ejecutar la aplicación
139.     root.mainloop()
```

4. Detalles Técnicos

4.1. Validación de Datos

- **Entrada de Peso y Altura**:

Para garantizar que los datos ingresados por el usuario sean válidos y puedan ser utilizados en cálculos, el programa realiza las siguientes validaciones:

- **Conversión de Datos:**
    - **Peso:** Se convierte el valor ingresado a un número de punto flotante (float). Esto es necesario para manejar fracciones y asegurar que los cálculos sean precisos.
    - **Altura:** Similarmente, el valor de la altura se convierte a un número de punto flotante.
    - **Manejo de Errores:** Si el usuario ingresa un valor no numérico, la conversión a float o int fallará, y el programa debería capturar esta excepción para mostrar un mensaje de error adecuado.

4.2. Cálculo del IMC

- **Fórmula del IMC**:

La fórmula utilizada para calcular el IMC es:

$$IMC = \frac{peso\ (kg)}{(altura\ (m))^2}$$

Peso: Debe estar en kilogramos (kg).

Altura: Debe estar en metros (m). Si la altura se ingresa en centímetros, debes convertirla a metros dividiendo por 100.

4.3. Manejo de Archivos CSV

- **Almacenamiento de Resultados**:

El programa puede guardar los resultados del cálculo en un archivo CSV para su posterior análisis o registro. Cada registro en el archivo CSV incluirá:

- **Peso:** El valor ingresado por el usuario.
- **Altura:** El valor ingresado por el usuario.
- **IMC:** El valor calculado.

5. Conversión a Archivo Ejecutable

5.1. Uso de auto-py-to-exe

Para convertir el programa en un archivo ejecutable, siga los siguientes pasos:

1. **Instalar auto-py-to-exe**:

   pip install auto-py-to-exe

2. **Ejecutar auto-py-to-exe**:

   auto-py-to-exe

3. **Configurar y Convertir**:

   - . En la interfaz gráfica de auto-py-to-exe, sigue estos pasos para configurar el archivo ejecutable:

Seleccionar el Archivo Python:

- **Script de Entrada**: En la sección "Script de Entrada", selecciona tu archivo Python (main.py). Este es el archivo que deseas convertir en un ejecutable.

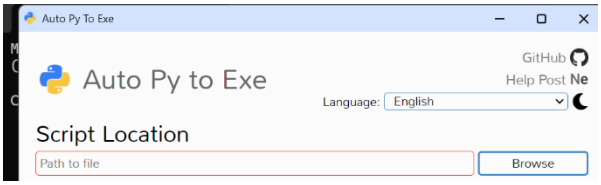

Imagen 4. Seleccionar el archivo a convertir a .exe

Configuración General:

- **Icono del Archivo Ejecutable** (Opcional): Puedes añadir un icono personalizado para el archivo .exe en la sección "Icono". Haz clic en "Add Icon" y selecciona el archivo de icono (en formato .ico).
- **Output Directory**: Define la carpeta en la que se guardará el archivo ejecutable. Por defecto, se guarda en la misma carpeta que el archivo .py.
- **Añadir Archivos**: Si tu script depende de archivos adicionales (como imágenes o archivos de configuración), agrégales en la sección "Additional Files". Esto asegura que todos los recursos necesarios estén incluidos en el ejecutable.

Imagen 5. Ajustes adicionales

Configuración Avanzada:

- **Configuraciones Adicionales**:
  - **--onefile**: Si seleccionas esta opción, auto-py-to-exe empaquetará todo en un solo archivo .exe, lo cual es útil para distribución. Si no la seleccionas, el ejecutable se descompondrá en varios archivos y carpetas.
  - **--noconsole**: Si no quieres que se muestre la consola al ejecutar el archivo .exe (útil para aplicaciones GUI), activa esta opción.

Configuración de Paquetes y Dependencias:

- **Dependencias**: Asegúrate de que todas las bibliotecas y módulos que tu script necesita estén instalados en tu entorno de Python. auto-py-to-exe generalmente detecta estas dependencias automáticamente, pero puedes revisar y ajustar según sea necesario.

Compilar el Ejecutable:

- Una vez configuradas todas las opciones, haz clic en el botón "Convert .py to .exe". El proceso de conversión comenzará y auto-py-to-exe generará el archivo ejecutable basado en tus configuraciones.

Imagen 6. Boton para convertir a .exe

4. Verificación y Pruebas

- **Ejecutar el Archivo .exe**: Navega a la carpeta de salida y ejecuta el archivo .exe para verificar que todo funcione como se espera.
- **Pruebas Adicionales**: Realiza pruebas adicionales para asegurarte de que el archivo ejecutable maneje correctamente las entradas y las funcionalidades de la aplicación.

6. Guía de Instalación y Uso

6.1. Instalación

- **Requisitos del Sistema**:
  - Sistema Operativo: Windows
  - Python 3.x y bibliotecas necesarias.

6..2. Uso

- **Iniciar la Aplicación**:
  - Ejecutar el archivo .exe generado o el script main.py.

- **Ingresar Datos**:
    - Introduzca su peso y altura en los campos correspondientes y haga clic en "Calcular".
- **Guardar Resultados**:
    - Los resultados se guardan automáticamente en un archivo CSV.