Analizaré el código línea por línea siguiendo tus reglas estrictas:

Línea 1: import tkinter as tk

- Importa la biblioteca tkinter (Tkinter) y le asigna el alias tk. Tkinter es la biblioteca estándar de Python para crear interfaces gráficas de usuario (GUI).

Línea 2: from tkinter import messagebox

- Importa específicamente el submódulo messagebox desde la biblioteca tkinter. Este submódulo proporciona funciones para mostrar cuadros de diálogo de mensajes estándar, como alertas o información.

Línea 3: (En blanco)

- Línea en blanco: Separa la sección de importaciones de la definición de las funciones de lógica.

Línea 4: def mostrar_historia():

- Define una función llamada mostrar_historia. Esta función no acepta ningún parámetro. Su propósito es mostrar un cuadro de mensaje con información sobre la historia de la animación por computadora.

Línea 5: mensaje = (

- Dentro de la función mostrar_historia: Inicia la definición de una variable mensaje a la que se le asignará una cadena de texto multilinea. El uso de paréntesis () permite que la cadena se divida en varias líneas en el código para mejorar la legibilidad, y Python las concatenará automáticamente en una sola cadena.

Línea 6: "* Década de 1960: Inicios con Ivan Sutherland y el programa Sketchpad.\\n"

- Parte de la cadena multilinea asignada a mensaje: Describe los inicios de la animación por computadora en los años 60. \n inserta un salto de línea en el texto mostrado.

Línea 7: "* 1970s: Ford de Disney introduce secuencias CGI.\\n"

- Parte de la cadena multilinea asignada a mensaje: Menciona la década de 1970 y el uso de CGI en Disney. \n inserta un salto de línea. Nota: El texto "Ford de Disney" parece ser un error tipográfico y debería ser "Futureworld" (la primera película en usar gráficos 3D renderizados).

Línea 8: "* 1995: 'Toy Story' de Pixar es el primer largometraje animado completamente por computadora."

- Parte final de la cadena multilinea asignada a mensaje: Destaca el lanzamiento de 'Toy Story' en 1995. No hay \n al final, lo que significa que no habrá una línea en blanco después de este punto en el cuadro de mensaje.

Línea 9:)

- Cierra la definición de la cadena multilinea asignada a mensaje.

Línea 10: messagebox.showinfo("Historia", mensaje)

- Dentro de la función mostrar_historia: Llama a la función showinfo() del submódulo messagebox. Esta función muestra un cuadro de diálogo informativo.
- "Historia": Es el título que aparecerá en la barra de título del cuadro de diálogo.
- mensaje: Es el contenido del mensaje que se mostrará dentro del cuadro de diálogo.

Línea 11: (En blanco)

- Línea en blanco: Separa la definición de la función mostrar_historia de la siguiente función mostrar_evolucion.

Línea 12: def mostrar_evolucion():

- Define una función llamada mostrar_evolucion. Esta función no acepta ningún parámetro. Su propósito es mostrar un cuadro de mensaje con información sobre la evolución tecnológica en la animación por computadora.

Línea 13: mensaje = (

- Dentro de la función mostrar_evolucion: Inicia la definición de una cadena de texto multilinea para la variable mensaje.

Línea 14: "* 1980/1990: Nace el CGI, mejora el modelado 3D y el renderizado.\\n"

- Parte de la cadena multilinea asignada a mensaje: Describe el período de los 80 y 90 y el nacimiento del CGI.

Línea 15: "* 2000-2010: Realismo con captura de movimiento, sombras y simulaciones físicas.\\n"

- Parte de la cadena multilinea asignada a mensaje: Destaca las mejoras de realismo en la década de 2000.

Línea 16: "* 2010 en adelante: Uso de inteligencia artificial y realidad virtual."

- Parte final de la cadena multilinea asignada a mensaje: Describe las tendencias a partir de 2010.

Línea 17:)

- Cierra la definición de la cadena multilinea asignada a mensaje.

Línea 18: messagebox.showinfo("Evolución", mensaje)

- Dentro de la función mostrar_evolucion: Llama a la función showinfo() para mostrar el cuadro de diálogo informativo.
- "Evolución": Es el título del cuadro de diálogo.
- mensaje: Es el contenido del mensaje.

Línea 19: (En blanco)

- Línea en blanco: Separa la definición de la función mostrar_evolucion de la siguiente función mostrar_aplicaciones.

Línea 20: def mostrar_aplicaciones():

- Define una función llamada mostrar_aplicaciones. Esta función no acepta ningún parámetro. Su propósito es mostrar un cuadro de mensaje con información sobre las aplicaciones de la animación por computadora.

Línea 21: mensaje = (

- Dentro de la función mostrar_aplicaciones: Inicia la definición de una cadena de texto multilinea para la variable mensaje.

Línea 22: "* Cine y TV: Efectos especiales, personajes virtuales.\\n"

- Parte de la cadena multilinea asignada a mensaje: Describe las aplicaciones en cine y TV.

Línea 23: "* Videojuegos: Modelado 3D, cinemáticas, mundos interactivos.\\n"

- Parte de la cadena multilinea asignada a mensaje: Describe las aplicaciones en videojuegos.

Línea 24: "* Medicina: Visualización de órganos, simulaciones quirúrgicas.\\n"

- Parte de la cadena multilinea asignada a mensaje: Describe las aplicaciones en medicina.

Línea 25: "* Arquitectura: Recorridos virtuales, simulaciones estructurales.\\n"

- Parte de la cadena multilinea asignada a mensaje: Describe las aplicaciones en arquitectura.

Línea 26: "* Educación: Videos explicativos, e-learning, animaciones didácticas."

- Parte final de la cadena multilinea asignada a mensaje: Describe las aplicaciones en educación.

Línea 27:)

- Cierra la definición de la cadena multilinea asignada a mensaje.

Línea 28: messagebox.showinfo("Aplicaciones", mensaje)

- Dentro de la función mostrar_aplicaciones: Llama a la función showinfo() para mostrar el cuadro de diálogo informativo.
- "Aplicaciones": Es el título del cuadro de diálogo.
- mensaje: Es el contenido del mensaje.

Línea 29: (En blanco)

- Línea en blanco: Separa la definición de las funciones de lógica de la creación de la ventana principal de Tkinter.

Línea 30: # Crear ventana principal

- Comentario que indica el inicio de la sección donde se crea y configura la ventana principal de la interfaz gráfica.

Línea 31: ventana = tk.Tk()

- Crea una instancia de la clase Tk() del módulo tkinter. Esta instancia representa la ventana principal de la aplicación GUI y se asigna a la variable ventana.

Línea 32: ventana.title("Animación por Computadora")

- Llama al método title() del objeto ventana para establecer el texto que aparecerá en la barra de título de la ventana.

Línea 33: ventana.geometry("400x300")

- Llama al método geometry() del objeto ventana para establecer el tamaño inicial de la ventana en píxeles. La ventana tendrá un ancho de 400 píxeles y un alto de 300 píxeles.

Línea 34: ventana.config(bg="#f0f0f0")

- Llama al método config() del objeto ventana para configurar sus propiedades.
- bg="#f0f0f0": Establece el color de fondo de la ventana a un gris claro, utilizando un código hexadecimal RGB.

Línea 35: (En blanco)

- Línea en blanco: Separa la configuración de la ventana principal de la creación del widget de título.

Línea 36: # Título

- Comentario que indica la creación del widget que mostrará el título principal dentro de la ventana.

Línea 37: titulo = tk.Label(ventana, text="Animación por Computadora", font=("Arial", 16, "bold"), bg="#f0f0f0")

- Crea una instancia de la clase Label de tkinter. Los parámetros son:
    - ventana: El widget padre (la ventana principal) donde se colocará esta etiqueta.
    - text="Animación por Computadora": El texto que se mostrará en la etiqueta.
    - font=("Arial", 16, "bold"): Una tupla que define la fuente: familia "Arial", tamaño 16, y estilo "bold" (negrita).
    - bg="#f0f0f0": El color de fondo de la etiqueta, coincidiendo con el de la ventana.
- El objeto Label resultante se asigna a la variable titulo.

Línea 38: titulo.pack(pady=20)

- Llama al método pack() del widget titulo. pack() es un administrador de geometría que organiza los widgets en bloques antes de colocarlos en el widget padre.

- pady=20: Añade un relleno vertical de 20 píxeles por encima y por debajo del widget.

Línea 39: (En blanco)

- Línea en blanco: Separa el widget de título de la creación de los botones.

Línea 40: # Botones

- Comentario que indica el inicio de la sección donde se crearán los widgets de botón.

Línea 41: btn_historia = tk.Button(ventana, text="Historia", command=mostrar_historia, width=30)

- Crea una instancia de la clase Button de tkinter. Los parámetros son:
    - ventana: El widget padre.
    - text="Historia": El texto que se mostrará en el botón.
    - command=mostrar_historia: Especifica la función que se llamará cuando se haga clic en el botón. En este caso, llamará a la función mostrar_historia.
    - width=30: Establece el ancho del botón en 30 unidades de texto (no píxeles).
- El objeto Button resultante se asigna a la variable btn_historia.

Línea 42: btn_historia.pack(pady=5)

- Llama al método pack() del botón btn_historia.
- pady=5: Añade un relleno vertical de 5 píxeles por encima y por debajo del botón.

Línea 43: (En blanco)

- Línea en blanco: Separa la configuración del botón de historia del botón de evolución.

Línea 44: btn_evolucion = tk.Button(ventana, text="Evolución", command=mostrar_evolucion, width=30)

- Crea un botón similar a btn_historia.
- text="Evolución": Texto del botón.

- command=mostrar_evolucion: Al hacer clic, llama a la función mostrar_evolucion.
- width=30: Ancho del botón.

Línea 45: btn_evolucion.pack(pady=5)

- Llama al método pack() del botón btn_evolucion con un relleno vertical de 5 píxeles.

Línea 46: (En blanco)

- Línea en blanco: Separa la configuración del botón de evolución del botón de aplicaciones.

Línea 47: btn_aplicaciones = tk.Button(ventana, text="Aplicaciones", command=mostrar_aplicaciones, width=30)

- Crea un botón similar a los anteriores.
- text="Aplicaciones": Texto del botón.
- command=mostrar_aplicaciones: Al hacer clic, llama a la función mostrar_aplicaciones.
- width=30: Ancho del botón.

Línea 48: btn_aplicaciones.pack(pady=5)

- Llama al método pack() del botón btn_aplicaciones con un relleno vertical de 5 píxeles.

Línea 49: (En blanco)

- Línea en blanco: Separa la configuración del botón de aplicaciones del botón de salir.

Línea 50: btn_salir = tk.Button(ventana, text="Salir", command=ventana.quit, width=30, fg="white", bg="red")

- Crea el botón para salir de la aplicación.
- text="Salir": Texto del botón.
- command=ventana.quit: Al hacer clic, llama al método quit() del objeto ventana. Este método cierra la ventana principal de Tkinter y termina el bucle principal de la aplicación.

- width=30: Ancho del botón.
- fg="white": Establece el color del texto del botón a blanco (foreground).
- bg="red": Establece el color de fondo del botón a rojo (background).

Línea 51: btn_salir.pack(pady=20)

- Llama al método pack() del botón btn_salir.
- pady=20: Añade un relleno vertical de 20 píxeles por encima y por debajo, separándolo más de los otros botones.

Línea 52: (En blanco)

- Línea en blanco: Separa la creación de los botones de la ejecución del bucle principal de Tkinter.

Línea 53: ventana.mainloop()

- Llama al método mainloop() del objeto ventana. Este método inicia el bucle de eventos principal de Tkinter. La aplicación GUI permanece abierta y responde a las interacciones del usuario (como clics de botón) hasta que se cierra la ventana. Esta es la última línea de código que se ejecuta en el flujo principal del script y bloquea la ejecución hasta que la ventana se cierra.

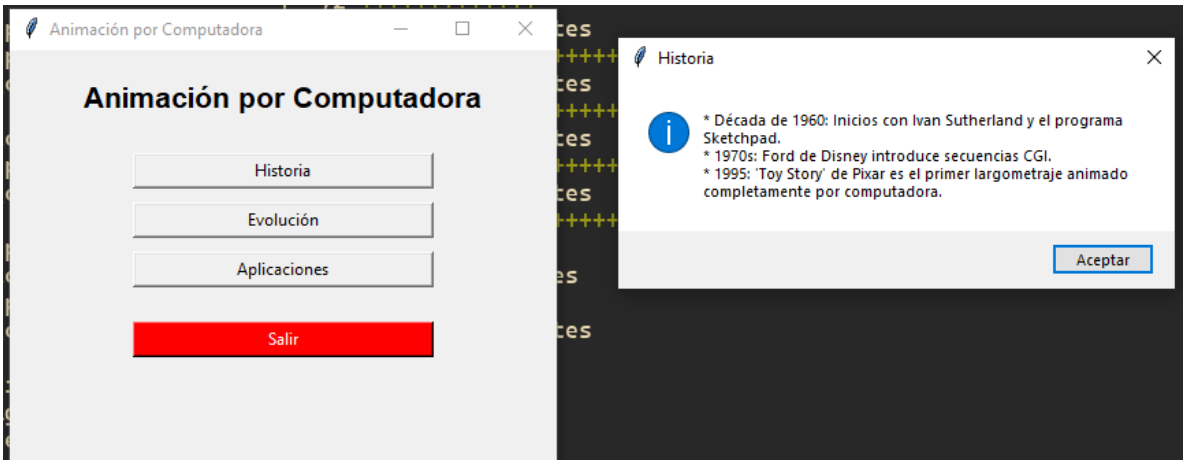
Resumen del Código

Este script de Python crea una sencilla aplicación de interfaz gráfica de usuario (GUI) utilizando la biblioteca tkinter. Su propósito es proporcionar información sobre la animación por computadora a través de ventanas de diálogo.

1. **Importaciones:** Importa los módulos tkinter (para la GUI) y tkinter.messagebox (para cuadros de diálogo).
2. **Funciones de Contenido:**
    - Define tres funciones (mostrar_historia, mostrar_evolucion, mostrar_aplicaciones).
    - Cada una de estas funciones prepara una cadena de texto multilinea con información relevante.
    - Cuando se llama, cada función utiliza messagebox.showinfo() para mostrar el texto en un cuadro de diálogo emergente con un título específico ("Historia", "Evolución", "Aplicaciones").

3. **Configuración de la Ventana Principal:**
    - Crea la ventana principal de la aplicación (tk.Tk()).
    - Establece el título de la ventana ("Animación por Computadora").
    - Define el tamaño inicial de la ventana (400x300 píxeles).
    - Configura el color de fondo de la ventana a un gris claro.
4. **Widgets de la Interfaz:**
    - Crea una etiqueta (tk.Label) para servir como título principal dentro de la ventana.
    - Crea cuatro botones (tk.Button): "Historia", "Evolución", "Aplicaciones" y "Salir".
    - A los botones "Historia", "Evolución" y "Aplicaciones" se les asigna una de las funciones de contenido (mostrar_historia, mostrar_evolucion, mostrar_aplicaciones) para que se ejecuten cuando se hace clic en ellos.
    - El botón "Salir" está configurado para llamar al método ventana.quit(), que cierra la aplicación.
    - Todos los widgets se organizan verticalmente utilizando el administrador de geometría pack(), con rellenos para una mejor distribución visual.
5. **Bucle Principal de la GUI:**
    - Finalmente, ventana.mainloop() inicia el bucle de eventos de Tkinter, lo que hace que la ventana se muestre, esté activa y responda a las interacciones del usuario hasta que se cierre.

En resumen, el código crea una interfaz de usuario simple con botones. Al hacer clic en un botón, se abre un cuadro de mensaje informativo que detalla aspectos sobre la historia, evolución o aplicaciones de la animación por computadora.


Animación por Computadora
Animación por Computadora
Historia
Evolución
Aplicaciones
Salir
Historia
* Década de 1960: Inicios con Ivan Sutherland y el programa Sketchpad.
* 1970s: Ford de Disney introduce secuencias CGI.
* 1995: 'Toy Story' de Pixar es el primer largometraje animado completamente por computadora.
Aceptar