In [ ]:

```
'''1. **Hello World: Display a simple "Hello, World!" message box.**

Here's a code snippet to display a "Hello, World!" message box using Tkinter:

```python'''
import tkinter as tk
from tkinter import messagebox

root = tk.Tk()
root.withdraw()

messagebox.showinfo("Hello", "Hello, World!")

root.mainloop()


'''2. **Button: Create a button that displays a message when clicked.**

You can create a button in Tkinter that displays a message when clicked using th

```python'''
import tkinter as tk
from tkinter import messagebox

def display_message():
    messagebox.showinfo("Button Clicked", "Button was clicked!")

root = tk.Tk()

button = tk.Button(root, text="Click me!", command=display_message)
button.pack()

root.mainloop()

'''3. **Entry: Create a text entry field and display the entered text.**

To create a text entry field and display the entered text, you can use the `Entr

```python'''
import tkinter as tk

def display_text():
    text = entry.get()
    label.config(text="Entered Text: " + text)

root = tk.Tk()

entry = tk.Entry(root)
entry.pack()

button = tk.Button(root, text="Display", command=display_text)
button.pack()

label = tk.Label(root)
label.pack()

root.mainloop()
```
```

```
'''4. **Check button: Create a checkbox and display the selected options.**

To create a checkbox and display the selected options, you can use the `Checkbut

```python'''
import tkinter as tk

def display_selected():
    options = []
    if option1_var.get():
        options.append("Option 1")
    if option2_var.get():
        options.append("Option 2")
    if option3_var.get():
        options.append("Option 3")

    label.config(text="Selected Options: " + ", ".join(options))

root = tk.Tk()

option1_var = tk.IntVar()
option2_var = tk.IntVar()
option3_var = tk.IntVar()

checkbutton1 = tk.Checkbutton(root, text="Option 1", variable=option1_var)
checkbutton1.pack()

checkbutton2 = tk.Checkbutton(root, text="Option 2", variable=option2_var)
checkbutton2.pack()

checkbutton3 = tk.Checkbutton(root, text="Option 3", variable=option3_var)
checkbutton3.pack()

button = tk.Button(root, text="Display", command=display_selected)
button.pack()

label = tk.Label(root)
label.pack()

root.mainloop()
```

In [ ]:

```
'''5. **Radio button: Create radio buttons and display the selected option.**

To create radio buttons and display the selected option, you can use the `Radiob

```python'''
import tkinter as tk

def display_selected():
    selected_option = option_var.get()
    label.config(text="Selected Option: " + selected_option)

root = tk.Tk()

option_var = tk.StringVar()

radiobutton1 = tk.Radiobutton(root, text="Option 1", variable=option_var, value=
radiobutton1.pack()
```

```
radiobutton2 = tk.Radiobutton(root, text="Option 2", variable=option_var, value=
radiobutton2.pack()

radiobutton3 = tk.Radiobutton(root, text="Option 3", variable=option_var, value=
radiobutton3.pack()

button = tk.Button(root, text="Display", command=display_selected)
button.pack()

label = tk.Label(root)
label.pack()

root.mainloop()
```

6. **List box: Create a list box and display the selected items.**

To create a list box and display the selected items, you can use the `Listbox` w

```python
import tkinter as tk

def display_selected():
    selected_items = [listbox.get(idx) for idx in listbox.curselection()]
    label.config(text="Selected Items: " + ", ".join(selected_items))

root = tk.Tk()

listbox = tk.Listbox(root, selectmode=tk.MULTIPLE)
listbox.pack()

for item in ["Item 1", "Item 2", "Item 3", "Item 4"]:
    listbox.insert(tk.END, item)

button = tk.Button(root, text="Display", command=display_selected)
button.pack()

label = tk.Label(root)
label.pack()

root.mainloop()
```

In [ ]:

```
'''7. **Text: Create a text area and display the entered text.**

To create a text area and display the entered text, you can use the `Text` widge

```python'''
import tkinter as tk

def display_text():
    entered_text = text.get("1.0", tk.END)
    label.config(text="Entered Text:\n" + entered_text)

root = tk.Tk()

text = tk.Text(root, height=5, width=30)
text.pack()

button = tk.Button(root, text="Display", command=display_text)
button.pack()
```
```

```
label = tk.Label(root)
label.pack()

root.mainloop()


'''8. **Menu: Create a menu with different options.**

To create a menu with different options, you can use the `Menu` widget in Tkinte

```python'''
import tkinter as tk

def file_new():
    label.config(text="New option selected")

def file_open():
    label.config(text="Open option selected")

def file_save():
    label.config(text="Save option selected")

root = tk.Tk()

menu_bar = tk.Menu(root)

file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="New", command=file_new)
file_menu.add_command(label="Open", command=file_open)
file_menu.add_command(label="Save", command=file_save)

menu_bar.add_cascade(label="File", menu=file_menu)

root.config(menu=menu_bar)

label = tk.Label(root, text="Select an option from the menu")
label.pack()

root.mainloop()


'''9. **Message: Display a message in a dialog box.**

To display a message in a dialog box, you can use the `messagebox` module from t

```python'''
import tkinter as tk
from tkinter import messagebox

def display_message():
    messagebox.showinfo("Message", "This is a message box.")

root = tk.Tk()
root.withdraw()

button = tk.Button(root, text="Display Message", command=display_message)
button.pack()

root.mainloop()
```

In [ ]:

'''10. **Progress bar: Create a progress bar that updates over time.**

To create a progress bar that updates over time, you can use the `Progressbar` w

```python'''
```python
import tkinter as tk
from tkinter import ttk

def start_progress():
    progress['maximum'] = 100
    for i in range(101):
        progress['value'] = i
        root.update_idletasks()
        progress_label.config(text="Progress: {}%".format(i))
        progress_bar.step(1)

root = tk.Tk()

progress_label = tk.Label(root, text="Progress: 0%")
progress_label.pack()

progress_bar = ttk.Progressbar(root, length=200, mode='determinate')
progress_bar.pack()

start_button = tk.Button(root, text="Start", command=start_progress)
start_button.pack()

root.mainloop()
```

'''11. **Scale: Create a scale widget and display the selected value.**

To create a scale widget and display the selected value, you can use the `Scale`

```python'''
```python
import tkinter as tk

def display_value(value):
    label.config(text="Selected Value: {}".format(value))

root = tk.Tk()

scale = tk.Scale(root, from_=0, to=100, orient=tk.HORIZONTAL, command=display_va
scale.pack()

label = tk.Label(root)
label.pack()

root.mainloop()
```

'''12. **Spin box: Create a spin box and display the selected value.**

To create a spin box (also known as a spin button or spin control) and display t

```python'''
```python
import tkinter as tk

def display_value():
```
```

```
    value = spinbox.get()
    label.config(text="Selected Value: {}".format(value))

root = tk.Tk()

spinbox = tk.Spinbox(root, from_=0, to=100, command=display_value)
spinbox.pack()

label = tk.Label(root)
label.pack()

root.mainloop()
```

In [ ]:

```
'''13. **Canvas: Create a canvas and draw shapes on it.**

To create a canvas and draw shapes on it, you can use the `Canvas` widget in Tki

```python'''
import tkinter as tk

def draw_shapes():
    # Clear the canvas
    canvas.delete(tk.ALL)

    # Draw a rectangle
    canvas.create_rectangle(50, 50, 150, 100, fill="red")

    # Draw an oval
    canvas.create_oval(200, 50, 300, 100, fill="blue")

    # Draw a line
    canvas.create_line(350, 50, 450, 100, fill="green", width=2)

    # Draw a polygon
    points = [500, 50, 550, 100, 500, 150, 450, 100]
    canvas.create_polygon(points, fill="yellow")

root = tk.Tk()

canvas = tk.Canvas(root, width=600, height=200)
canvas.pack()

button = tk.Button(root, text="Draw Shapes", command=draw_shapes)
button.pack()



'''14. **Label Frame: Create a labeled frame with widgets inside.**

To create a labeled frame with widgets inside, you can use the `LabelFrame` widg

```python'''
import tkinter as tk

def display_options():
    selected_options = []
    if check_var1.get():
        selected_options.append("Option 1")
    if check_var2.get():
        selected_options.append("Option 2")
```

```
    if check_var3.get():
        selected_options.append("Option 3")

    label.config(text="Selected Options: " + ", ".join(selected_options))

root = tk.Tk()

frame = tk.LabelFrame(root, text="Options")
frame.pack(padx=10, pady=10)

check_var1 = tk.IntVar()
check_button1 = tk.Checkbutton(frame, text="Option 1", variable=check_var1)
check_button1.pack()

check_var2 = tk.IntVar()
check_button2 = tk.Checkbutton(frame, text="Option 2", variable=check_var2)
check_button2.pack()

check_var3 = tk.IntVar()
check_button3 = tk.Checkbutton(frame, text="Option 3", variable=check_var3)
check_button3.pack()

button = tk.Button(root, text="Display", command=display_options)
button.pack()

label = tk.Label(root)
label.pack()

root.mainloop()
```

In [ ]:

```
'''15. **Scrollbar: Add a scrollbar to a widget like a text area or list box.**

To add a scrollbar to a widget like a text area or list box, you can use the `Sc

```python'''
import tkinter as tk

def scroll_text(*args):
    text.yview(*args)

root = tk.Tk()

scrollbar = tk.Scrollbar(root)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

text = tk.Text(root, yscrollcommand=scrollbar.set)
text.pack(side=tk.LEFT, fill=tk.BOTH)

scrollbar.config(command=scroll_text)

root.mainloop()


'''16. **Frame: Create a frame and place widgets inside it.**

To create a frame and place widgets inside it, you can use the `Frame` widget in

```python'''
import tkinter as tk
```

```
def greet():
    label.config(text="Hello, " + entry.get())

root = tk.Tk()

frame = tk.Frame(root)
frame.pack(padx=10, pady=10)

label = tk.Label(frame, text="Enter your name:")
label.pack()

entry = tk.Entry(frame)
entry.pack()

button = tk.Button(frame, text="Greet", command=greet)
button.pack()

root.mainloop()
```

In [ ]:

```
'''17. **Treeview: Create a tree view widget to display hierarchical data.**

To create a tree view widget to display hierarchical data, you can use the `Tree

```python'''
import tkinter as tk
from tkinter import ttk

def add_item():
    selected_item = tree.selection()
    if selected_item:
        tree.insert(selected_item, "end", text="New Item")

def remove_item():
    selected_item = tree.selection()
    if selected_item:
        tree.delete(selected_item)

root = tk.Tk()

tree = ttk.Treeview(root)
tree.pack()

# Insert parent items
parent1 = tree.insert("", "end", text="Parent 1")
parent2 = tree.insert("", "end", text="Parent 2")

# Insert child items
child1 = tree.insert(parent1, "end", text="Child 1")
child2 = tree.insert(parent1, "end", text="Child 2")
child3 = tree.insert(parent2, "end", text="Child 3")

add_button = tk.Button(root, text="Add Item", command=add_item)
add_button.pack()

remove_button = tk.Button(root, text="Remove Item", command=remove_item)
remove_button.pack()

root.mainloop()
'''```
```
```

````
18. **Notebook: Create a notebook widget with tabs.**

To create a notebook widget with tabs, you can use the `Notebook` widget from th

```python'''
import tkinter as tk
from tkinter import ttk

root = tk.Tk()

notebook = ttk.Notebook(root)
notebook.pack()

# Create tabs
tab1 = ttk.Frame(notebook)
notebook.add(tab1, text="Tab 1")

tab2 = ttk.Frame(notebook)
notebook.add(tab2, text="Tab 2")

tab3 = ttk.Frame(notebook)
notebook.add(tab3, text="Tab 3")

root.mainloop()
````

In [ ]:

````
'''19. **File Dialog: Open a file dialog to select a file.**

To open a file dialog and allow the user to select a file, you can use the `file

```python'''
import tkinter as tk
from tkinter import filedialog

def open_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        file_label.config(text="Selected File: " + file_path)

root = tk.Tk()

open_button = tk.Button(root, text="Open File", command=open_file)
open_button.pack()

file_label = tk.Label(root)
file_label.pack()

root.mainloop()
'''```

20. **Color Dialog: Open a color dialog to select a color.**

To open a color dialog and allow the user to select a color, you can use the `co

```python'''
import tkinter as tk
from tkinter import colorchooser

def choose_color():
    color = colorchooser.askcolor()
    if color[1]:
````
````

```
        color_label.config(text="Selected Color: " + color[1])

root = tk.Tk()

color_button = tk.Button(root, text="Choose Color", command=choose_color)
color_button.pack()

color_label = tk.Label(root)
color_label.pack()

root.mainloop()
```

In [ ]:

```
'''21. **Calculator: Create a basic calculator with buttons for arithmetic opera

To create a basic calculator with buttons for arithmetic operations, you can use

```python'''
import tkinter as tk

def button_click(number):
    current = result_label.get()
    result_label.delete(0, tk.END)
    result_label.insert(tk.END, current + str(number))

def button_clear():
    result_label.delete(0, tk.END)

def button_equal():
    expression = result_label.get()
    result = eval(expression)
    result_label.delete(0, tk.END)
    result_label.insert(tk.END, result)

root = tk.Tk()
root.title("Calculator")

result_label = tk.Entry(root, width=20, justify="right")
result_label.grid(row=0, column=0, columnspan=3, padx=10, pady=10)

button_1 = tk.Button(root, text="1", padx=10, pady=5, command=lambda: button_cli
button_2 = tk.Button(root, text="2", padx=10, pady=5, command=lambda: button_cli
button_3 = tk.Button(root, text="3", padx=10, pady=5, command=lambda: button_cli
button_4 = tk.Button(root, text="4", padx=10, pady=5, command=lambda: button_cli
button_5 = tk.Button(root, text="5", padx=10, pady=5, command=lambda: button_cli
button_6 = tk.Button(root, text="6", padx=10, pady=5, command=lambda: button_cli
button_7 = tk.Button(root, text="7", padx=10, pady=5, command=lambda: button_cli
button_8 = tk.Button(root, text="8", padx=10, pady=5, command=lambda: button_cli
button_9 = tk.Button(root, text="9", padx=10, pady=5, command=lambda: button_cli
button_0 = tk.Button(root, text="0", padx=10, pady=5, command=lambda: button_cli

button_add = tk.Button(root, text="+", padx=10, pady=5, command=lambda: button_c
button_subtract = tk.Button(root, text="-", padx=10, pady=5, command=lambda: but
button_multiply = tk.Button(root, text="*", padx=10, pady=5, command=lambda: but
button_divide = tk.Button(root, text="/", padx=10, pady=5, command=lambda: buttc

button_equal = tk.Button(root, text="=", padx=10, pady=5, command=button_equal)
button_clear = tk.Button(root, text="C", padx=10, pady=5, command=button_clear)

button_7.grid(row=1, column=0)
button_8.grid(row=1, column=1)
```
```

```
button_9.grid(row=1, column=2)
button_4.grid(row=2, column=0)
button_5.grid(row=2, column=1)
button_6.grid(row=2, column=2)
button_1.grid(row=3, column=0)
button_2.grid(row=3, column=1)
button_3.grid(row=3, column=2)
button_0.grid(row=4, column=0)
button_add.grid(row=1, column=3)
button_subtract.grid(row=2, column=3)
button_multiply.grid(row=3, column=3)
button_divide.grid(row=4, column=3)
button_equal.grid(row=4, column=2)
button_clear.grid(row=4, column=1)

root.mainloop()
```

In [ ]:

```
'''22. Temperature Converter: Build a program to convert between Celsius, Fahren

import tkinter as tk

def convert_temperature():
    temperature = float(entry_temperature.get())
    unit_from = variable_from.get()
    unit_to = variable_to.get()

    if unit_from == "Celsius":
        if unit_to == "Fahrenheit":
            result = temperature * 9/5 + 32
        elif unit_to == "Kelvin":
            result = temperature + 273.15
        else:
            result = temperature
    elif unit_from == "Fahrenheit":
        if unit_to == "Celsius":
            result = (temperature - 32) * 5/9
        elif unit_to == "Kelvin":
            result = (temperature - 32) * 5/9 + 273.15
        else:
            result = temperature
    elif unit_from == "Kelvin":
        if unit_to == "Celsius":
            result = temperature - 273.15
        elif unit_to == "Fahrenheit":
            result = (temperature - 273.15) * 9/5 + 32
        else:
            result = temperature
    else:
        result = temperature

    label_result.config(text="Result: " + str(result))

root = tk.Tk()
root.title("Temperature Converter")

label_temperature = tk.Label(root, text="Temperature:")
label_temperature.pack()

entry_temperature = tk.Entry(root)
entry_temperature.pack()
```

```
label_from = tk.Label(root, text="From:")
label_from.pack()

variable_from = tk.StringVar(root)
variable_from.set("Celsius")
option_menu_from = tk.OptionMenu(root, variable_from, "Celsius", "Fahrenheit", "
option_menu_from.pack()

label_to = tk.Label(root, text="To:")
label_to.pack()

variable_to = tk.StringVar(root)
variable_to.set("Fahrenheit")
option_menu_to = tk.OptionMenu(root, variable_to, "Celsius", "Fahrenheit", "Kelv
option_menu_to.pack()

convert_button = tk.Button(root, text="Convert", command=convert_temperature)
convert_button.pack()

label_result = tk.Label(root)
label_result.pack()

root.mainloop()
```

In [ ]:

```
'''23. Length Converter: Develop a tool to convert between different units of le

import tkinter as tk

length_units = {
    "Millimeter": 0.001,
    "Centimeter": 0.01,
    "Meter": 1.0,
    "Kilometer": 1000.0,
    "Inch": 0.0254,
    "Foot": 0.3048,
    "Yard": 0.9144,
    "Mile": 1609.34
}

def convert_length():
    length = float(entry_length.get())
    unit_from = variable_from.get()
    unit_to = variable_to.get()

    result = length * length_units[unit_from] / length_units[unit_to]
    label_result.config(text="Result: " + str(result))

root = tk.Tk()
root.title("Length Converter")

label_length = tk.Label(root, text="Length:")
label_length.pack()

entry_length = tk.Entry(root)
entry_length.pack()

label_from = tk.Label(root, text="From:")
label_from.pack()
```

```
variable_from = tk.StringVar(root)
variable_from.set("Meter")
option_menu_from = tk.OptionMenu(root, variable_from, *length_units.keys())
option_menu_from.pack()

label_to = tk.Label(root, text="To:")
label_to.pack()

variable_to = tk.StringVar(root)
variable_to.set("Centimeter")
option_menu_to = tk.OptionMenu(root, variable_to, *length_units.keys())
option_menu_to.pack()

convert_button = tk.Button(root, text="Convert", command=convert_length)
convert_button.pack()

label_result = tk.Label(root)
label_result.pack()

root.mainloop()
```

In [ ]:

```
'''24. **Currency Converter: Create a currency converter with real-time exchange

```python'''
import tkinter as tk
import requests

def convert_currency():
    amount = float(entry_amount.get())
    from_currency = variable_from.get()
    to_currency = variable_to.get()

    response = requests.get(f"https://api.exchangerate-api.com/v4/latest/{from_c
    exchange_rates = response.json()["rates"]
    rate = exchange_rates[to_currency]

    converted_amount = amount * rate
    label_result.config(text="Converted Amount: " + str(converted_amount))

root = tk.Tk()
root.title("Currency Converter")

label_amount = tk.Label(root, text="Amount:")
label_amount.pack()

entry_amount = tk.Entry(root)
entry_amount.pack()

label_from = tk.Label(root, text="From Currency:")
label_from.pack()

variable_from = tk.StringVar(root)
variable_from.set("USD")
option_menu_from = tk.OptionMenu(root, variable_from, "USD", "EUR", "GBP")
option_menu_from.pack()

label_to = tk.Label(root, text="To Currency:")
label_to.pack()

variable_to = tk.StringVar(root)
```
```

```
variable_to.set("EUR")
option_menu_to = tk.OptionMenu(root, variable_to, "USD", "EUR", "GBP")
option_menu_to.pack()

convert_button = tk.Button(root, text="Convert", command=convert_currency)
convert_button.pack()

label_result = tk.Label(root)
label_result.pack()

root.mainloop()
```'''

In this code, the program creates a GUI with an entry field to enter the amount,

When the convert button is clicked, the `convert_currency` function is called. I

Please note that for this code to work, you need an active internet connection t
'''

In [ ]:

In [ ]:
```
'''25. **Simple Notepad: Build a basic text editor with features like open, save

```python'''
import tkinter as tk
from tkinter import filedialog

current_file = None

def open_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        global current_file
        current_file = file_path
        with open(file_path, "r") as file:
            text_editor.delete("1.0", tk.END)
            text_editor.insert(tk.END, file.read())

def save_file():
    if current_file:
        with open(current_file, "w") as file:
            file.write(text_editor.get("1.0", tk.END))
    else:
        save_file_as()

def save_file_as():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt")
    if file_path:
        global current_file
        current_file = file_path
        with open(file_path, "w") as file:
            file.write(text_editor.get("1.0", tk.END))

root = tk.Tk()
root.title("Simple Notepad")

text_editor = tk.Text(root)
text_editor.pack()
```
```

```
menu_bar = tk.Menu(root)
file_menu = tk.Menu(menu_bar, tearoff=False)
file_menu.add_command(label="Open", command=open_file)
file_menu.add_command(label="Save", command=save_file)
file_menu.add_command(label="Save As", command=save_file_as)
file_menu.add_separator()
file_menu.add_command(label="Exit", command=root.quit)
menu_bar.add_cascade(label="File", menu=file_menu)

root.config(menu=menu_bar)
root.mainloop()
```

In this code, the program creates a simple notepad GUI with a text editor area.

The `open_file` function uses the `filedialog` module to open a file dialog and

The `save_file` function checks if there is a currently open file (`current_file

The `save_file_as` function uses the `filedialog` module to open a file dialog a

The menu bar is created using the `Menu` widget. The file menu is added to the m

Please note that this basic notepad implementation does not include advanced tex

'''

Out[ ]: '```\n\nIn this code, the program creates a simple notepad GUI with a text edit
or area. It also provides a menu bar with options to open, save, save as, and e
xit.\n\nThe `open_file` function uses the `filedialog` module to open a file di
alog and allows the user to select a file. If a file is selected, its contents
are read and displayed in the text editor.\n\nThe `save_file` function checks i
f there is a currently open file (`current_file`), and if so, it saves the cont
ents of the text editor to that file. If there is no current file, it calls the
`save_file_as` function.\n\nThe `save_file_as` function uses the `filedialog` m
odule to open a file dialog and allows the user to choose a file name and locat
ion to save the contents of the text editor.\n\nThe menu bar is created using t
he `Menu` widget. The file menu is added to the menu bar with options for open,
save, save as, and exit.\n\nPlease note that this basic notepad implementation
does not include advanced text editing features like undo, redo, or formatting.
It provides a starting point that can be expanded upon based on specific requir
ements.\n\nLet me know if you have any questions or if you would like to see co
de examples for the next topics.'

In [ ]:

```
'''26. **Image Viewer: Create an application to display images from a specified

```python'''
import tkinter as tk
from PIL import ImageTk, Image
import os

image_folder = "path/to/image/folder"
images = []
current_index = 0

def load_images():
    global images
    images = []
    for filename in os.listdir(image_folder):
```
```

```
        if filename.endswith(".jpg") or filename.endswith(".png"):
            images.append(os.path.join(image_folder, filename))

def show_image(index):
    if 0 <= index < len(images):
        image_path = images[index]
        image = Image.open(image_path)
        image = image.resize((500, 500))  # Adjust the size as needed
        photo = ImageTk.PhotoImage(image)
        image_label.config(image=photo)
        image_label.image = photo
        status_label.config(text=f"Image {index+1} of {len(images)}")
    else:
        image_label.config(image="")
        status_label.config(text="No Image")

def next_image():
    global current_index
    current_index = (current_index + 1) % len(images)
    show_image(current_index)

def previous_image():
    global current_index
    current_index = (current_index - 1) % len(images)
    show_image(current_index)

root = tk.Tk()
root.title("Image Viewer")

image_label = tk.Label(root)
image_label.pack()

button_frame = tk.Frame(root)
button_frame.pack()

previous_button = tk.Button(button_frame, text="Previous", command=previous_imag
previous_button.pack(side="left")

next_button = tk.Button(button_frame, text="Next", command=next_image)
next_button.pack(side="left")

status_label = tk.Label(root, text="")
status_label.pack()

load_images()
show_image(current_index)

root.mainloop()
'''

In this code, the program creates an image viewer GUI that displays images from

Make sure to replace `"path/to/image/folder"` with the actual path to the folder

The `load_images` function scans the image folder and populates the `images` lis

The `show_image` function takes an index parameter and displays the correspondin

The `next_image` and `previous_image` functions update the `current_index` varia
```

```
The GUI includes a label to display the image, buttons to navigate to the next o

Please note that you need to have the PIL (Python Imaging Library) module instal
```

In [ ]:

```
'''27. **Stopwatch: Design a stopwatch with start, stop, and reset functionaliti

```python'''
import tkinter as tk
import time

class Stopwatch:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Stopwatch")

        self.elapsed_time = 0
        self.running = False

        self.time_label = tk.Label(self.root, text="00:00:00", font=("Helvetica"
        self.time_label.pack(pady=20)

        button_frame = tk.Frame(self.root)
        button_frame.pack()

        self.start_button = tk.Button(button_frame, text="Start", width=10, comm
        self.start_button.pack(side="left")

        self.stop_button = tk.Button(button_frame, text="Stop", width=10, commar
        self.stop_button.pack(side="left")

        self.reset_button = tk.Button(button_frame, text="Reset", width=10, comm
        self.reset_button.pack(side="left")

        self.root.mainloop()

    def start(self):
        if not self.running:
            self.start_time = time.time() - self.elapsed_time
            self.update()

    def stop(self):
        if self.running:
            self.root.after_cancel(self.timer)
            self.elapsed_time = time.time() - self.start_time
            self.running = False

    def reset(self):
        self.stop()
        self.elapsed_time = 0
        self.update()

    def update(self):
        self.elapsed_time = time.time() - self.start_time
        hours = int(self.elapsed_time / 3600)
        minutes = int((self.elapsed_time % 3600) / 60)
        seconds = int(self.elapsed_time % 60)
        time_string = f"{hours:02d}:{minutes:02d}:{seconds:02d}"
        self.time_label.config(text=time_string)
        self.timer = self.root.after(1000, self.update)
```
```

```
stopwatch = Stopwatch()
'''

In this code, a stopwatch GUI is created using the Tkinter library. The stopwatc

The `Stopwatch` class is defined with an `__init__` method that sets up the GUI.

The GUI includes a label to display the elapsed time, and three buttons for star

The `start` method is called when the start button is clicked. It checks if the

The `stop` method is called when the stop button is clicked. It cancels the sche

The `reset` method is called when the reset button is clicked. It stops the stop

The `update` method is responsible for updating the displayed time. It calculate

An instance of the `Stopwatch` class is created, which starts the stopwatch and
'''
```