**Course: Python**

1. **Final Project (code with comments)**

import tkinter as tk

from tkinter import ttk, scrolledtext, messagebox

import requests

from bs4 import BeautifulSoup

import threading

import pandas as pd

import time

class MovieScraperApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Movie Suggestion App")

self.root.geometry("600x500")

self.style = ttk.Style()

self.style.configure('TLabel', font=('Arial', 12))

self.style.configure('TButton', font=('Arial', 12))

self.create\_widgets()

def create\_widgets(self):

self.genre\_label = ttk.Label(self.root, text="Select a movie genre:")

self.genre\_label.pack(pady=10)

self.genres = ['Action', 'Adventure', 'Comedy', 'Drama', 'Fantasy', 'Horror', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller']

self.genre\_combobox = ttk.Combobox(self.root, values=self.genres, state='readonly', width=30)

self.genre\_combobox.pack()

self.top\_label = ttk.Label(self.root, text="Select number of top movies:")

self.top\_label.pack(pady=10)

self.top\_combobox = ttk.Combobox(self.root, values=[3, 5, 10], state='readonly', width=10)

self.top\_combobox.pack()

self.fetch\_button = ttk.Button(self.root, text="Fetch Movies", command=self.fetch\_and\_display)

self.fetch\_button.pack(pady=20)

self.clear\_button = ttk.Button(self.root, text="Clear Screen", command=self.clear\_screen)

self.clear\_button.pack(pady=10)

self.results\_text = scrolledtext.ScrolledText(self.root, height=15, width=70, wrap=tk.WORD)

self.results\_text.pack(pady=20)

def fetch\_and\_display(self):

genre = self.genre\_combobox.get()

if not genre:

messagebox.showerror("Error", "Please select a genre.")

return

top\_number = self.top\_combobox.get()

if not top\_number:

messagebox.showerror("Error", "Please select number of top movies.")

return

self.results\_text.delete('1.0', tk.END)

self.results\_text.insert(tk.END, f"Fetching movies for {genre} genre..\n")

self.root.update() **# Update GUI to show fetching message**

**# Call method to scrape and display movies using a separate thread**

threading.Thread(target=self.fetch\_movies, args=(genre, int(top\_number))).start()

def fetch\_movies(self, genre, top\_number):

url = f'https://www.rottentomatoes.com/browse/movies\_in\_theaters/genres:{genre.lower()}'

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'

}

try:

response = requests.get(url, headers=headers)

response.raise\_for\_status() # Check if the request was successful

soup = BeautifulSoup(response.content, 'html.parser')

movies = []

movie\_tags = soup.select('span.p--small[data-qa="discovery-media-list-item-title"]')

for tag in movie\_tags:

movie\_name = tag.get\_text(strip=True)

movies.append(movie\_name)

**# Export movies to an Excel file**

self.export\_to\_excel(movies, genre)

**# Clear results text and show movies**

self.results\_text.delete('1.0', tk.END)

self.results\_text.insert(tk.END, f"Catalog Received for {genre} genre:\n\n")

self.results\_text.insert(tk.END, f"Top {top\_number} {genre} movies:\n\n")

self.root.update() # Update GUI to show catalog received message

**# Read from Excel file and display top movies**

self.read\_from\_excel\_and\_display(genre, top\_number)

except requests.exceptions.RequestException as e:

messagebox.showerror("Network Error", f"Failed to fetch movies: {e}")

except Exception as e:

messagebox.showerror("Error", f"An error occurred: {e}")

def export\_to\_excel(self, movies, genre):

"""Export the list of movies to an Excel file"""

df = pd.DataFrame(movies, columns=['Movie Name'])

excel\_filename = f"scraped\_raw\_{genre.lower()}\_movies.xlsx"

df.to\_excel(excel\_filename, index=False)

self.results\_text.insert(tk.END, f"Movies exported to {excel\_filename}\n")

self.root.update()

def read\_from\_excel\_and\_display(self, genre, top\_number):

"""Read movies from the Excel file and display the top movies"""

excel\_filename = f"scraped\_raw\_{genre.lower()}\_movies.xlsx"

df = pd.read\_excel(excel\_filename)

top\_movies = df['Movie Name'].head(top\_number)

**# Print each movie with a delay**

for idx, movie in enumerate(top\_movies, start=1):

self.results\_text.insert(tk.END, f"{idx}. {movie}\n")

self.root.update() # Update the GUI to show each movie

time.sleep(0.5) # Add a delay between displaying each movie

**# Add final message after displaying all movies**

self.results\_text.insert(tk.END, "\nPlease enjoy the movies!\n")

def clear\_screen(self):

self.genre\_combobox.set('') # Clear genre selection

self.top\_combobox.set('') # Clear top number selection

self.results\_text.delete('1.0', tk.END) # Clear text area

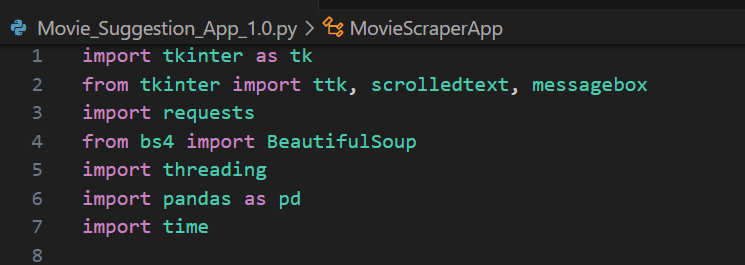
if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = MovieScraperApp(root)

root.mainloop()

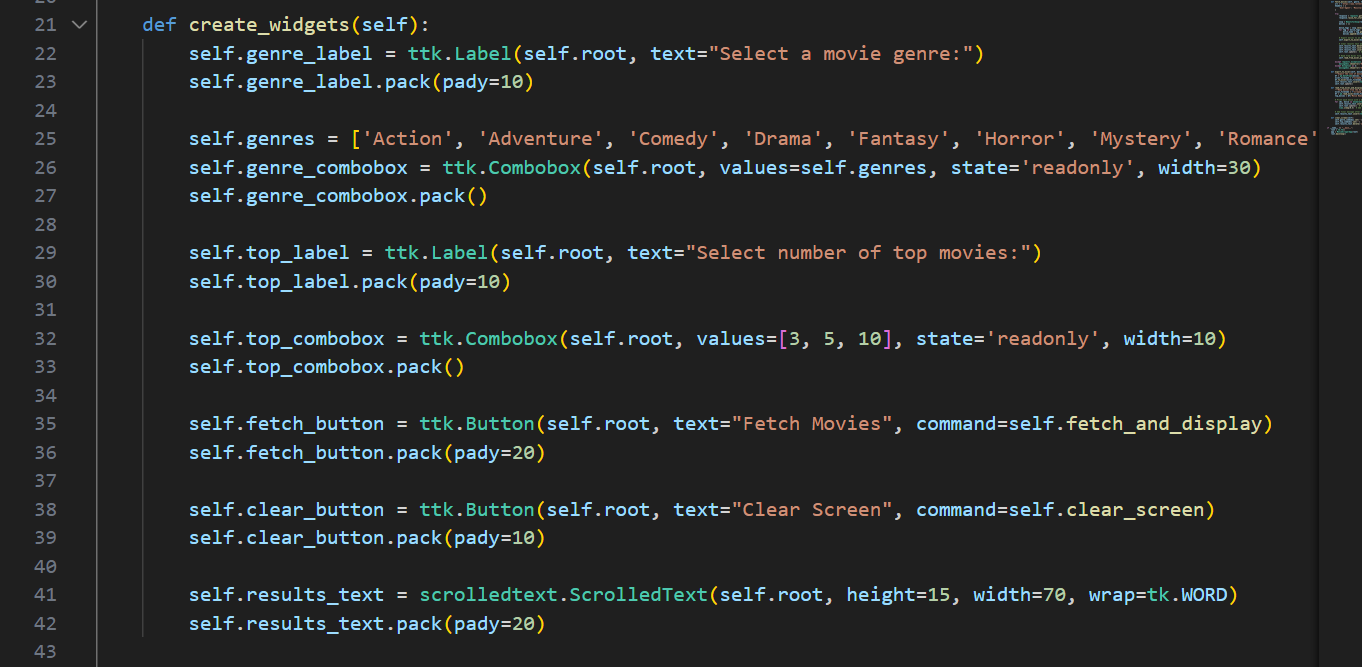
1. **Explanation (Step by step)**



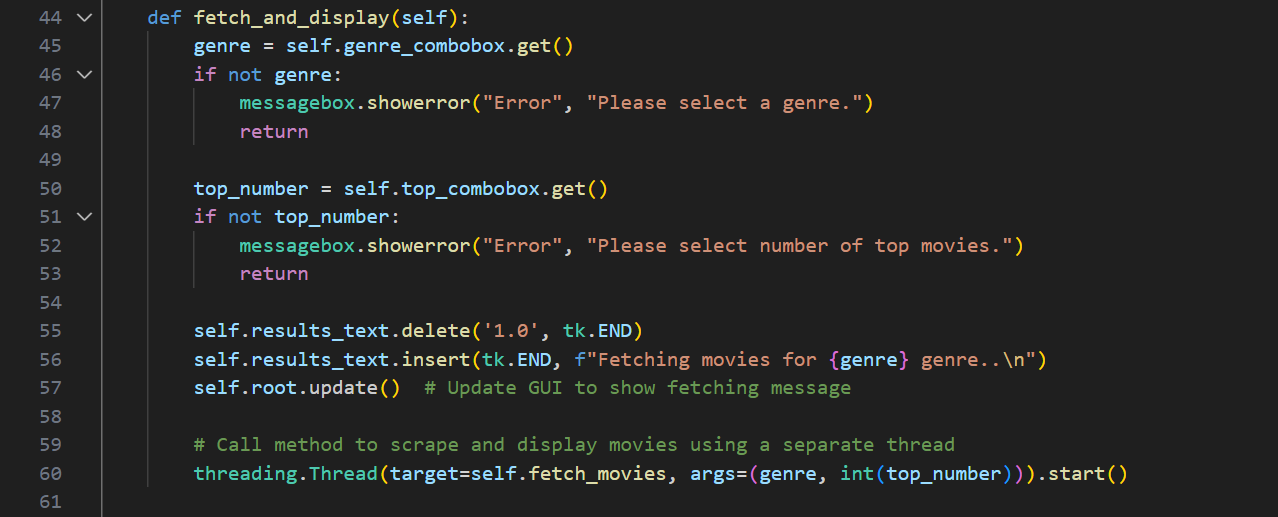
* **Libraries Import**: This block imports necessary libraries for GUI creation (tkinter), web scraping (requests, BeautifulSoup), threading, data handling (pandas), and introducing delays (time).



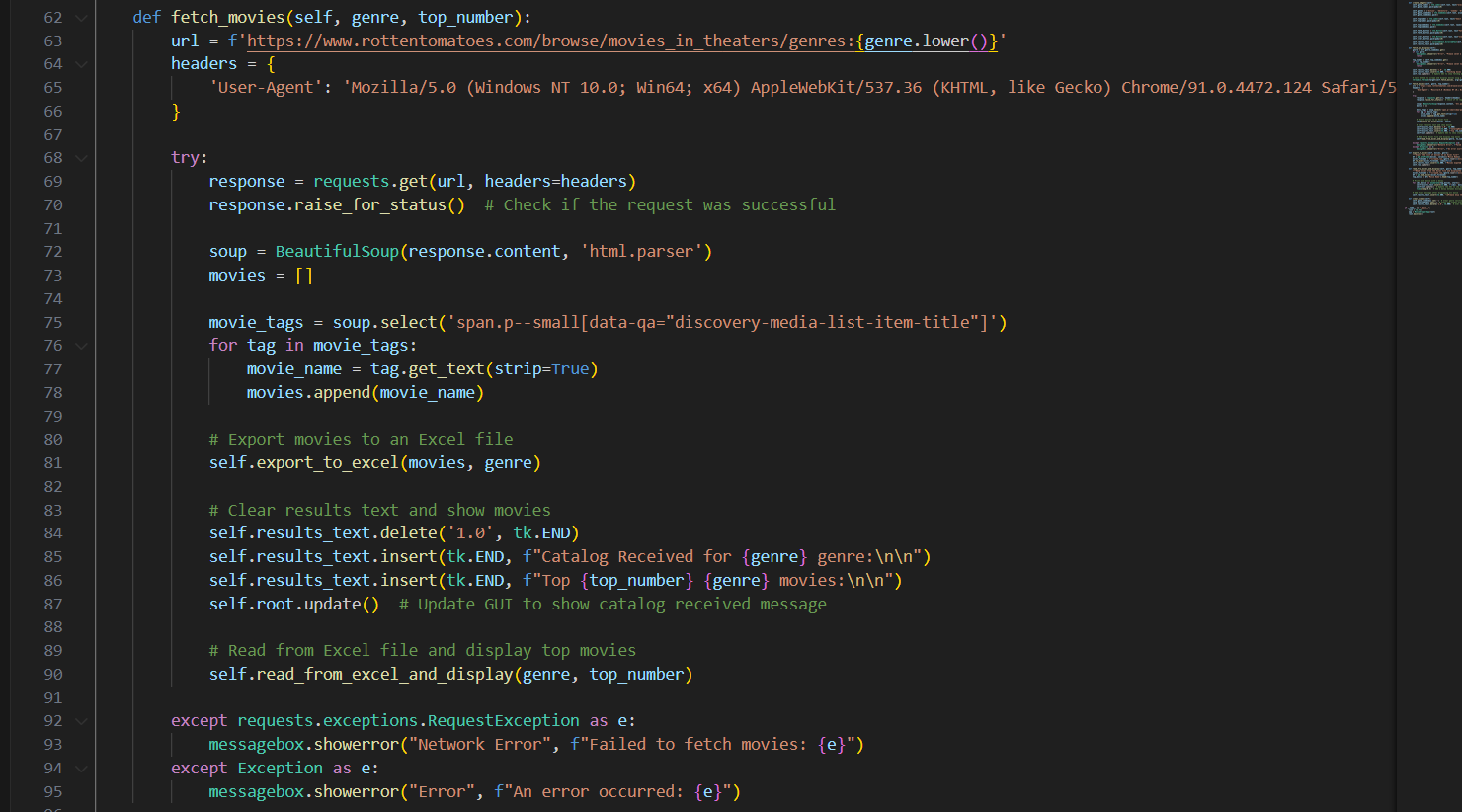
* **Initializing a class**: The class named MovieScraperApp initializes the GUI window, sets its title, dimensions, and styles using tkinter and ttk.



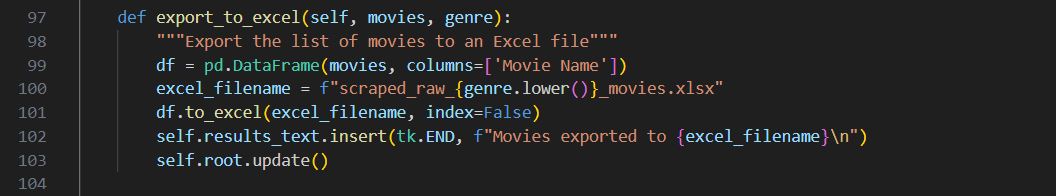
* **Creating Widgets**: This function creates and packs the widgets (labels, combo boxes, buttons, and text area) into the window. It includes selection options for genres and the number of movies to fetch, as well as buttons to fetch movies and clear the screen.



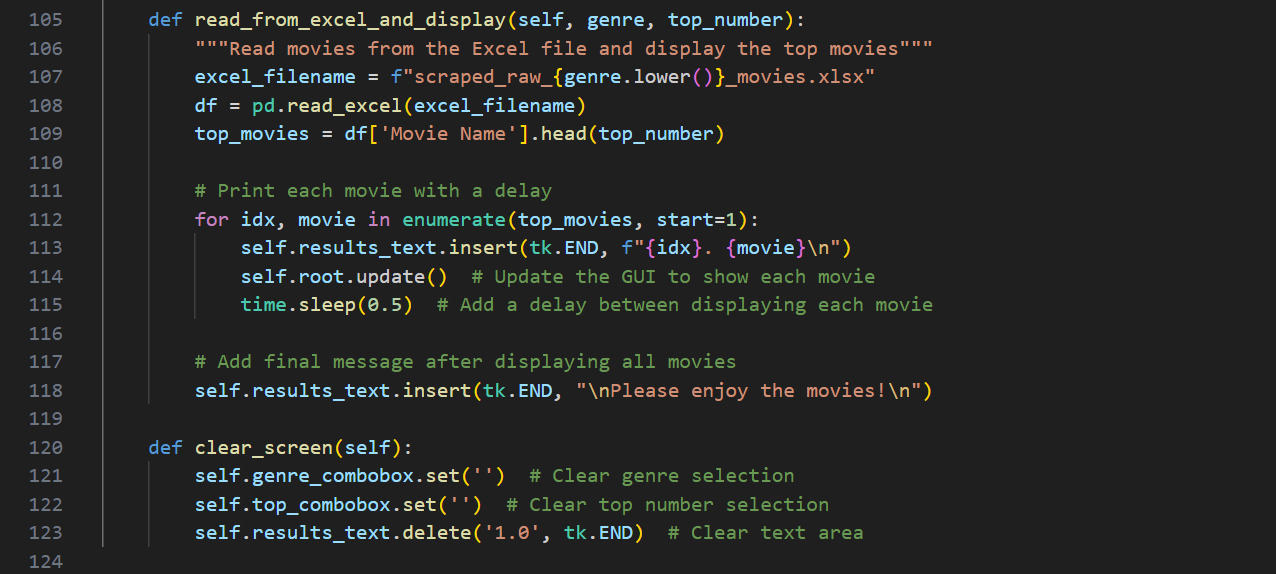
* **Fetching and Displaying Movies**: This function gets the selected genre and number of movies, displays a fetching message, and starts a new thread to fetch movies to keep the GUI responsive.



* **Fetching Movies**: This function builds the URL for scraping, makes the HTTP request, parses the HTML response to extract movie titles, and exports the data to an Excel file. It handles errors and updates the GUI.



* **Exporting to Excel**: This function takes a list of movies and exports it to an Excel file named based on the genre.

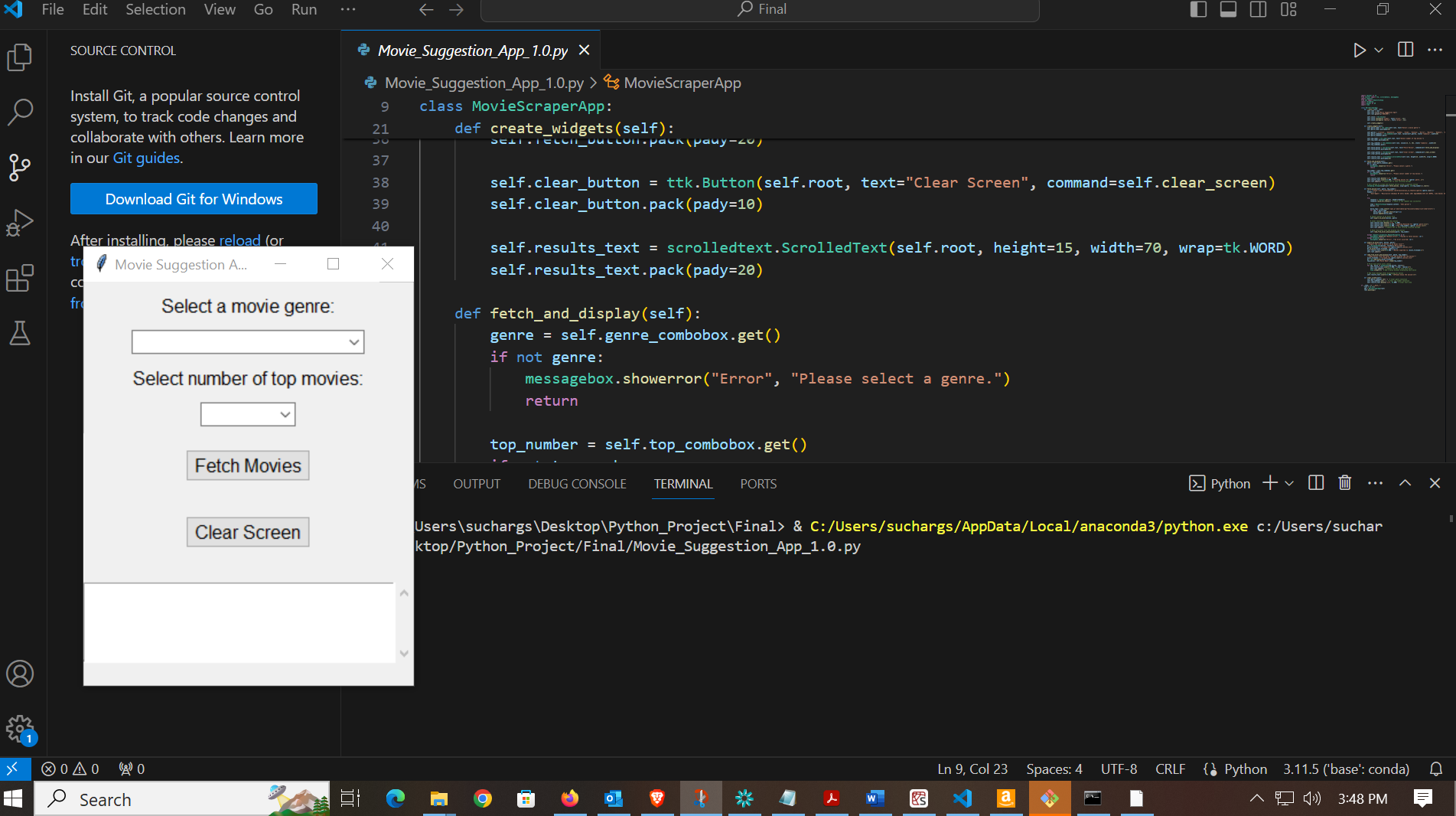


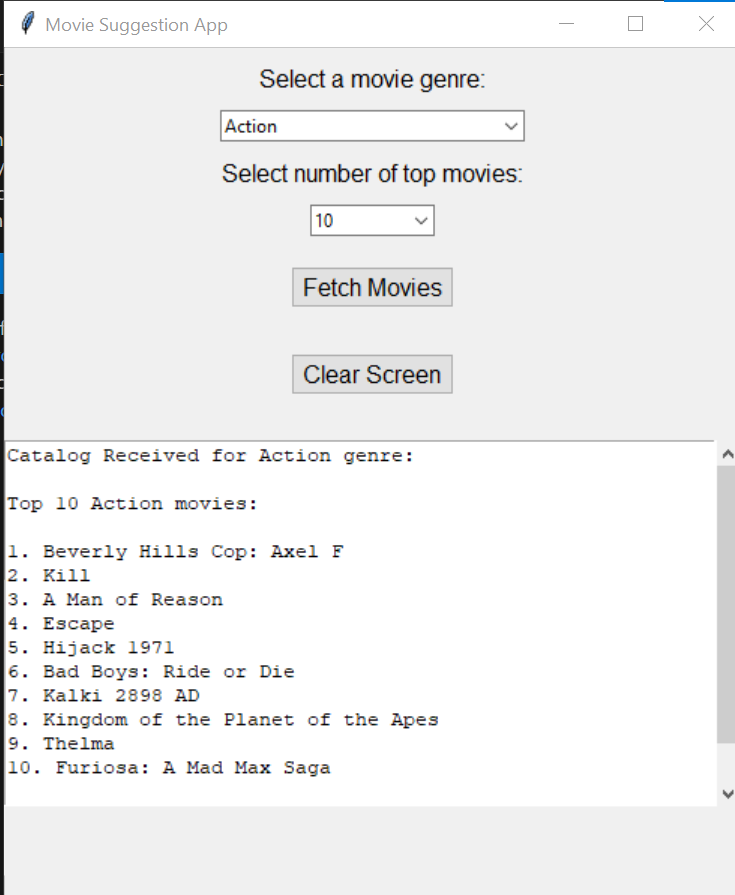
* **Reading from Excel and Displaying**: This function reads the movie list from the Excel file using pandas, extracts the top movies and displays them in the GUI.



* **Clearing the Screen**: This function resets the genre and top number selections and clears the text area in the GUI and then creates the main Tkinter window, initializes the MovieScraperApp class, and starts the Tkinter main loop.

1. **Output**





1. <https://github.com/suchargs/Python-Project-Submission> - GITHUB link