NAME	V G MANASA
REGISTRATION NO	24BEC1419
CLASS	610

PYTHON WEB SCRAPING PROJECT

"MOVIE INSIGHT ASSISTANT"

CODE:

```
import requests

import tkinter as tk

from tkinter import messagebox

from imdb import IMDb


# Entering API_KEY with actual TMDb API key

API_KEY = '2c6df7c1160103f34cf0aed9c55fcd86'


# Genre IDs from TMDb

GENRES = {

    "Action": 28,

    "Comedy": 35,

    "Drama": 18,

    "Horror": 27,

    "Sci-Fi": 878,

    "Thriller": 53,

    "Animation": 16,

    "Adventure": 12,

    "Crime": 80,

    "Fantasy": 14,

    "Mystery": 9648,
    "Musicals": 10402,
    "Sports": 16,
    "Western": 37,
    "Romance": 10749
}

class MovieInfoChatApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Movie Insight Assistant")
        self.root.geometry("750x700")  # Initial size
        self.root.configure(bg="#FAF7FC")

        self.fullscreen = False
        self.create_widgets()
        self.root.bind('<Return>', self.on_enter_key)  # Bind Enter key to send
        self.root.bind('<F11>', self.toggle_fullscreen)  # Bind F11 key to toggle fullscreen

    def create_widgets(self):
        # Title Header Button
        self.header_button = tk.Button(
            self.root,
            text="Movie Insight Assistant",
            anchor='center',
            command=None,
            relief='raised',
            bg="#4D0F4B",
            fg="white",
            font=("Helvetica", 22, 'bold'),
            padx=10,                    #padding for better alignment
            pady=10,
            width=30
        )
        self.header_button.pack(side=tk.TOP, fill=tk.X, padx=10, pady=(5, 10))

        # Main Frame
        main_frame = tk.Frame(self.root, bg="#f0f0f0")
        main_frame.pack(fill=tk.BOTH, expand=True)

        # Chat History Frame
        chat_frame = tk.Frame(main_frame, bg="#FAF7FC")
        chat_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True, padx=10, pady=(10, 5))

        # Chat History Canvas
        self.chat_canvas = tk.Canvas(chat_frame, bg="#FAF7FC", bd=0)
        self.chat_canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        # Scrollbar for Canvas
        self.scrollbar = tk.Scrollbar(chat_frame, orient=tk.VERTICAL,
command=self.chat_canvas.yview)
        self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self.chat_canvas.configure(yscrollcommand=self.scrollbar.set)

        # Frame inside Canvas for messages
        self.chat_frame = tk.Frame(self.chat_canvas, bg="#FAF7FC")
        self.chat_canvas.create_window((0, 0), window=self.chat_frame, anchor="nw")

        # Clear History Button
        self.clear_history_button = tk.Button(
            main_frame,
            text="Clear History",
            command=self.clear_history,
            bg="#D11547",
            fg="white",
            font=("Helvetica", 12, 'bold')
        )
        self.clear_history_button.pack(side=tk.TOP, fill=tk.X, padx=10, pady=(0, 5))

        # Genre Buttons Frame
        genre_buttons_frame = tk.Frame(main_frame, bg="#f0f0f0")
        genre_buttons_frame.pack(side=tk.TOP, fill=tk.X, padx=10, pady=(0, 5))

        # Add New Button Above Genre Buttons
        self.select_genre_button = tk.Button(
            genre_buttons_frame,
            text="Choose a genre below",
            bg="#0193A5",  # Button background #0193A5
            fg="white",
            font=("Helvetica", 12, 'bold')
        )
        self.select_genre_button.grid(row=0, column=0, columnspan=4, pady=(0, 10), sticky="ew")

        # Create Genre Buttons
        for idx, (genre, genre_id) in enumerate(GENRES.items()):
            button = tk.Button(
                genre_buttons_frame,
                text=genre,
                command=lambda g=genre: self.handle_genre_click(g),
                bg="#F6A278",  # Button background #F6A278
                fg="black",
                font=("Helvetica", 12, 'bold')
            )
            row = (idx + 1) // 4  # Adjust row index for proper placement
            col = (idx + 1) % 4
            button.grid(row=row + 1, column=col, padx=5, pady=5, sticky="ew")

        # Button to Prompt User to Enter a Movie
        self.enter_movie_button = tk.Button(
            main_frame,
            text="Enter a movie",
            bg="#FBC1AD",  # Button color #FBC1AD
            fg="black",
            font=("Helvetica", 12, 'bold')
        )
        self.enter_movie_button.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=(0, 5))

        # Input Frame
        input_frame = tk.Frame(main_frame, bg="#f0f0f0")
        input_frame.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=(0, 10))

        # Entry Field
        self.movie_title_entry = tk.Entry(input_frame, font=("Helvetica", 12))
        self.movie_title_entry.pack(side=tk.LEFT, fill=tk.X, expand=True, padx=(0, 5), pady=5)

        # Send Button
        self.send_button = tk.Button(
            input_frame,
            text="Send",
            command=self.get_movie_info,
            bg="#FF5050",
            fg="white",
            font=("Helvetica", 12, 'bold')
        )
        self.send_button.pack(side=tk.LEFT, padx=5, pady=5)
```

```
        # Full-Screen Button
        self.fullscreen_button = tk.Button(
            self.root,
            text="Full-Screen",
            command=self.toggle_fullscreen,
            bg="#111441",
            fg="white",
            font=("Helvetica", 12, 'bold')
        )
        self.fullscreen_button.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=(5, 10))

        # Update scroll region
        self.update_scroll_region()

    def handle_genre_click(self, genre):
        # Append the user's query (genre) to chat history
        self.update_chat_history(f"You: {genre}", tag="user")
        genre_id = GENRES.get(genre)
        if genre_id:
            self.fetch_movies_by_genre(genre_id)
        else:
            self.update_chat_history("Bot: Genre not found.", tag="bot")

    def fetch_movies_by_genre(self, genre_id):
        url =
f"https://api.themoviedb.org/3/discover/movie?api_key={API_KEY}&with_genres={genre_id}"
        response = requests.get(url)
        data = response.json()

        if 'results' in data and data['results']:
            # Prepare table format
            serial_no_width = 10
            movie_name_width = 40
            rating_width = 10
            line_char = '─'  # Character for drawing lines

            header = f"{'Serial No':<{serial_no_width}} {'Movie Name':<{movie_name_width}}
{'Rating':<{rating_width}}"
            divider = line_char * len(header)
            table_data = f" ┌{divider}┐ \n"
            table_data += f" │ {header} │ \n"
            table_data += f" ├{divider}┤ \n"

            for i, item in enumerate(data['results'], start=1):
                serial_no = f"{i:<{serial_no_width}}"
                movie_name = f"{item['title'][:movie_name_width]:<{movie_name_width}}"
                rating = f"{item['vote_average']:<{rating_width}}"
                table_data += f" │ {serial_no} {movie_name} {rating} │ \n"

            table_data += f" └{divider}┘ "

            # Display table in chat history
            self.update_chat_history(f"Bot:\n{table_data}", tag="bot")
        else:
            self.update_chat_history("Bot: No movies found for this genre.", tag="bot")

    def update_chat_history(self, message, tag):
        # Create a new frame for the message
        message_frame = tk.Frame(self.chat_frame, bg="#e0e0e0", padx=10, pady=5, relief="ridge",
borderwidth=2)
        message_frame.pack(fill=tk.X, pady=(0, 5), anchor="w")
        # Add delete button
        delete_button = tk.Button(message_frame, text="Delete", command=lambda:
self.delete_message(message_frame), bg="#D11547", fg="white", font=("Helvetica", 10, 'bold'))
        delete_button.pack(side=tk.RIGHT, padx=5, pady=5)

        # Add message text
        message_label = tk.Label(message_frame, text=message, bg=self.get_message_bg(tag),
font=("Courier", 12), justify="left", anchor="w", wraplength=self.chat_canvas.winfo_width() - 70)
        message_label.pack(side=tk.LEFT, fill=tk.X, expand=True)

        # Update canvas scroll region
        self.update_scroll_region()

    def get_message_bg(self, tag):
        if tag == "user":
            return "#d1ffd1"
        elif tag == "bot":
            return "#FCDCDC"
        return "#e0e0e0"

    def delete_message(self, frame):
        frame.destroy()
        self.update_scroll_region()

    def clear_entry(self):
        self.movie_title_entry.delete(0, tk.END)

    def clear_history(self):
        # Clear the chat history
        for widget in self.chat_frame.winfo_children():
            widget.destroy()
        self.update_scroll_region()

    def update_scroll_region(self):
        self.chat_canvas.update_idletasks()
        self.chat_canvas.config(scrollregion=self.chat_canvas.bbox("all"))

    def get_movie_info(self):
        movie_title = self.movie_title_entry.get()
        if not movie_title:
            messagebox.showwarning("Input Error", "Please enter a movie title.")
            return

        # Append the user's query to chat history
        self.update_chat_history(f"You: {movie_title}", tag="user")

        ia = IMDb()
        try:
            # Search for the movie
            movies = ia.search_movie(movie_title)
            if not movies:
                self.update_chat_history(f"Bot: No results found for '{movie_title}'", tag="bot")
                self.clear_entry()
                return

            # Select the first result
            movie = movies[0]
            ia.update(movie)

            # Fetch movie details
            title = movie.get('title', 'N/A')
            year = movie.get('year', 'N/A')
            genres = ', '.join(movie.get('genres', []))
            directors = ', '.join([director.get('name') for director in movie.get('directors', [])])
            cast = ', '.join([person.get('name') for person in movie.get('cast', [])[:20]])  # Limit to first 
5 cast members
            plot = movie.get('plot', ['N/A'])[0]
            rating = movie.get('rating', 'N/A')
            languages = ', '.join(movie.get('languages', []))

            # Format the response
            movie_info = f"Title: {title}\nYear: {year}\nGenres: {genres}\nDirector(s): 
{directors}\nCast: {cast}\nPlot: {plot}\nRating: {rating}\nLanguages: {languages}"

            # Append the bot's response to chat history
            self.update_chat_history(f"Bot: {movie_info}", tag="bot")
        except Exception as e:
            self.update_chat_history(f"Bot: Error fetching movie details. {str(e)}", tag="bot")

        self.clear_entry()

    def on_enter_key(self, event):
        self.get_movie_info()

    def toggle_fullscreen(self, event=None):
        self.fullscreen = not self.fullscreen
        self.root.attributes('-fullscreen', self.fullscreen)
        if self.fullscreen:
            self.fullscreen_button.config(text="Exit Full-Screen")
        else:
            self.fullscreen_button.config(text="Full-Screen")

if __name__ == "__main__":
    root = tk.Tk()
```

```
app = MovieInfoChatApp(root)

root.mainloop()
```