**Book Recommendation Bot**

**GitHub Project Link:** [**https://github.com/sahithi-2803/Python-Project-Submission**](https://github.com/sahithi-2803/Python-Project-Submission)

**Description of the Project:**

**T**his is a Python project that helps to find books to read based on the favorite genre. We can also choose to filter books by year or popularity and get a random book suggestion.

**Scope of this Project:**

* It suggests you pick a book genre (like horror, fantasy, or cooking)
* Gets real book data from the internet using Open Library
* It filters books by:
  + Published Year of the book
  + Popularity (how many editions the book has)
* Shows a random book recommendation
* Saves the list of books to a CSV file (so you can open it in Excel)

**Software & Libraries Used:**

* Python 3 - Programming language
* Requests - To get book data from a website
* Pandas - To work with book data easily

**Install Dependencies:**

* pip install pandas & requests

**How to Run:**

* Book\_recommendation\_bot.py

**How It Works:**

1. Welcome User

* Shows a greeting and list of available genres.

2. Get Genre Input

* Accepts genre input from the user.
* Keeps prompting if input is invalid.

3. Fetch Data from Open Library

* Sends an API request like: <https://openlibrary.org/subjects/fantasy.json?limit=300>

4. Filter Books

* User can filter the book list:
  + By year
  + By edition count (popularity)

5. Recommend a Book

* Picks one book at random from the (filtered) list.
* Displays the title, author(s), publication year, and edition count.

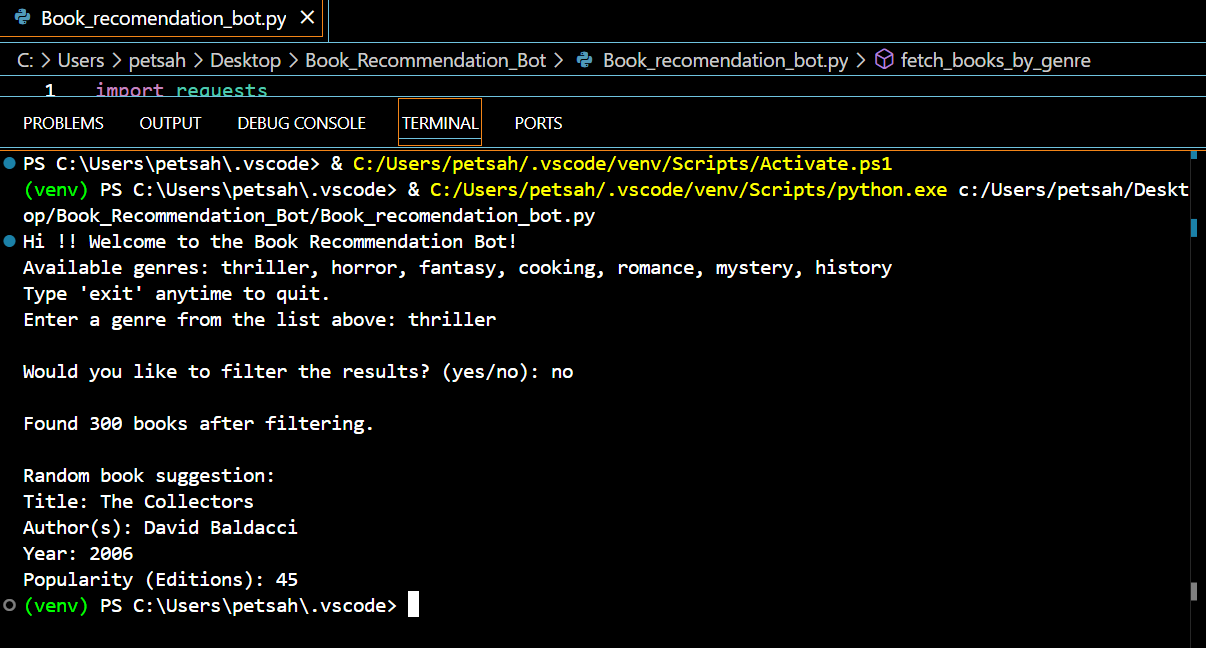
6. Save to CSV

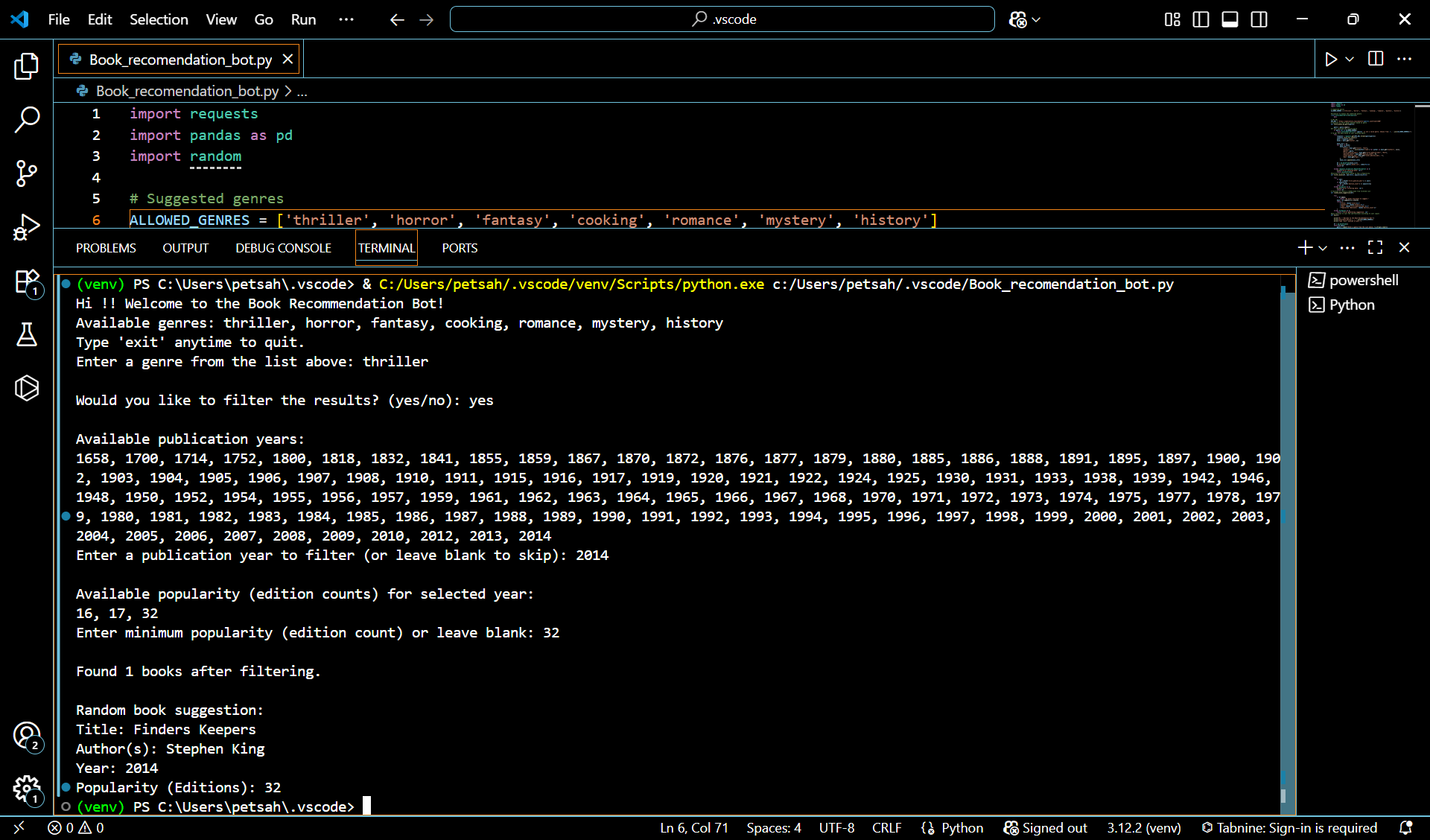
* The full dataset is optionally saved as a CSV file like fantasy\_books.csv.

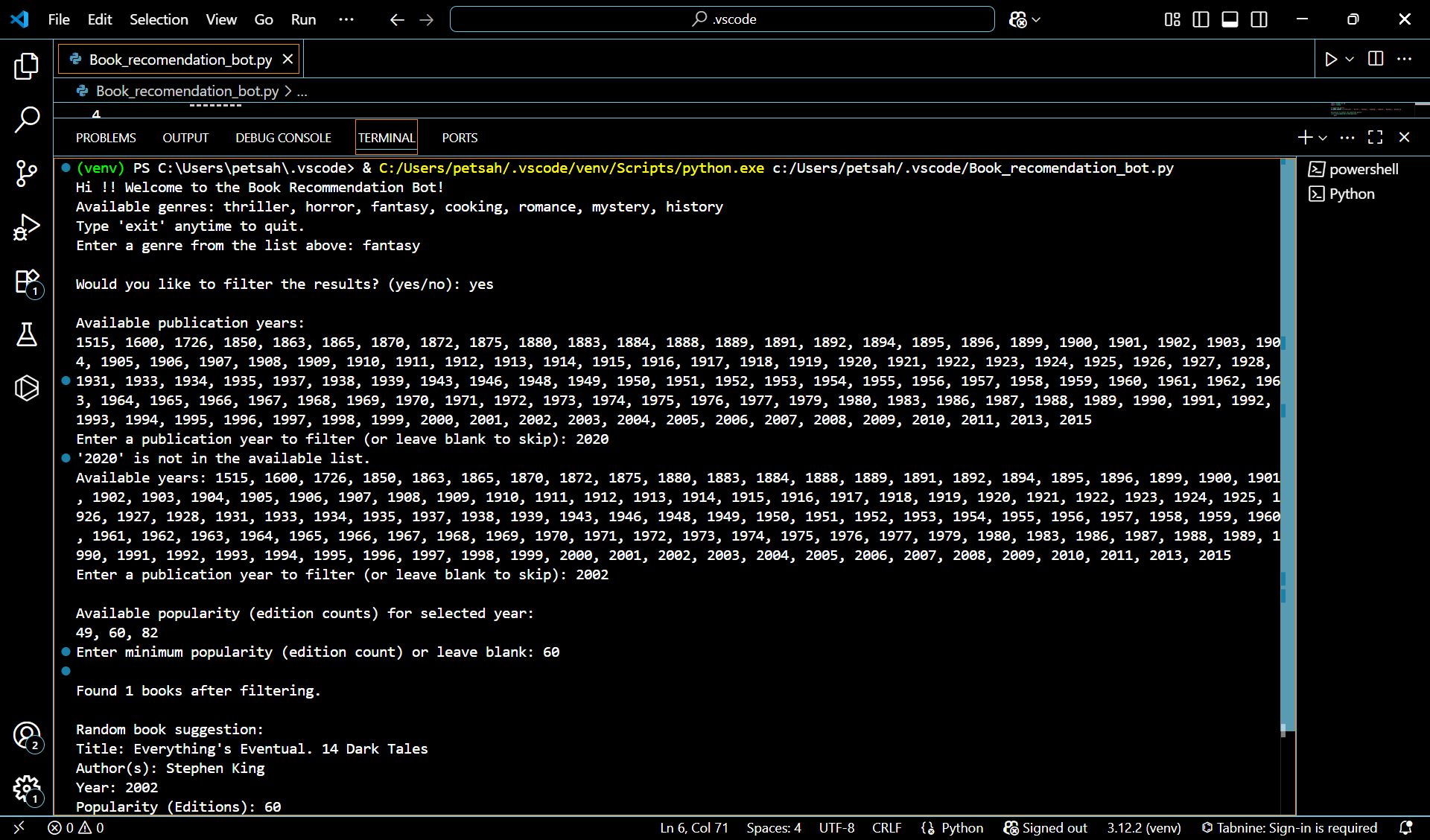
**Sample Data Fields (for Random book):**

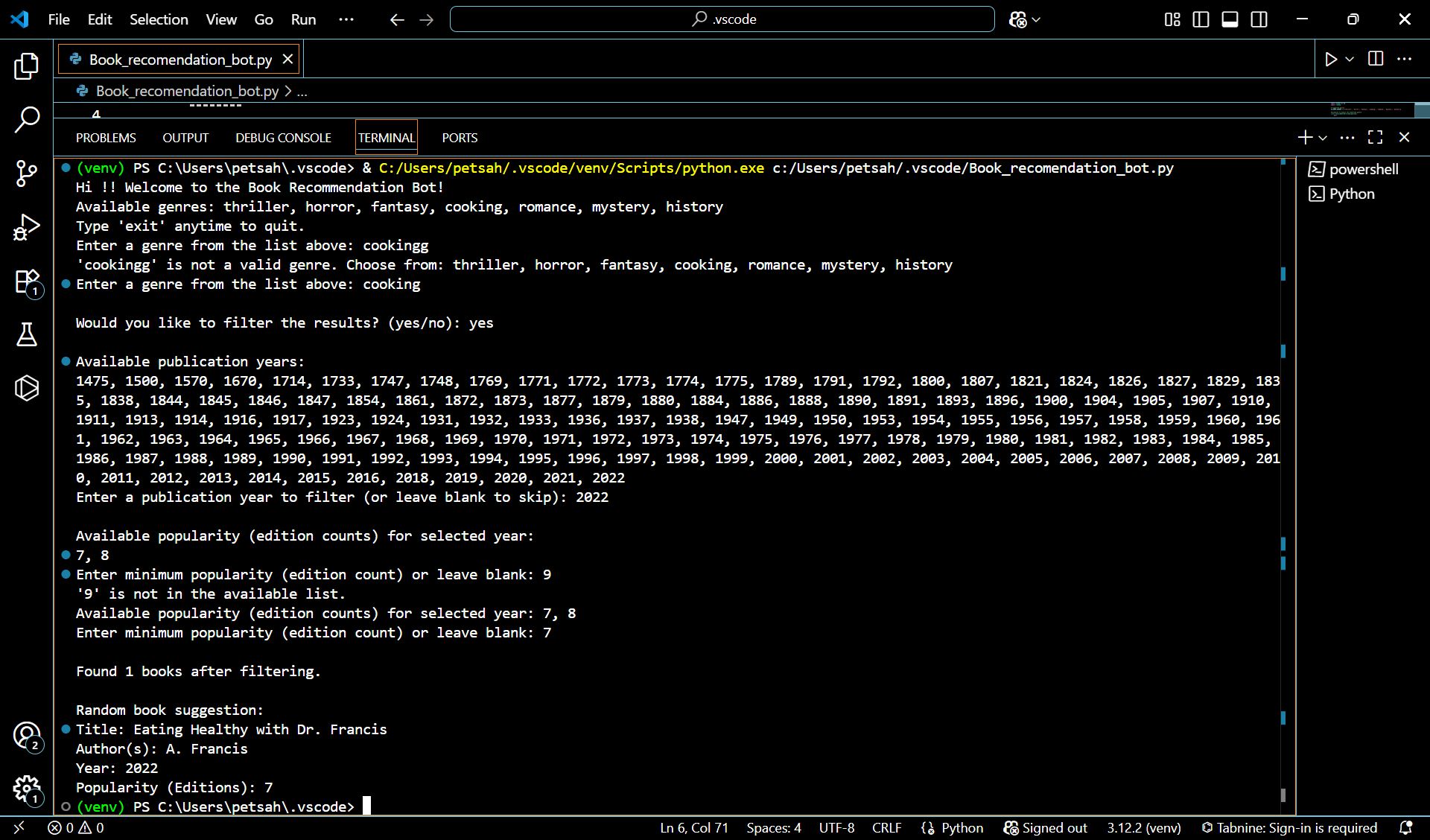
| **Field** | **Example** |
| --- | --- |
| Title | "The Hobbit" |
| Authors | "J.R.R. Tolkien" |
| Genre | "fantasy" |
| Year | 1937 |
| Popularity | 56 |

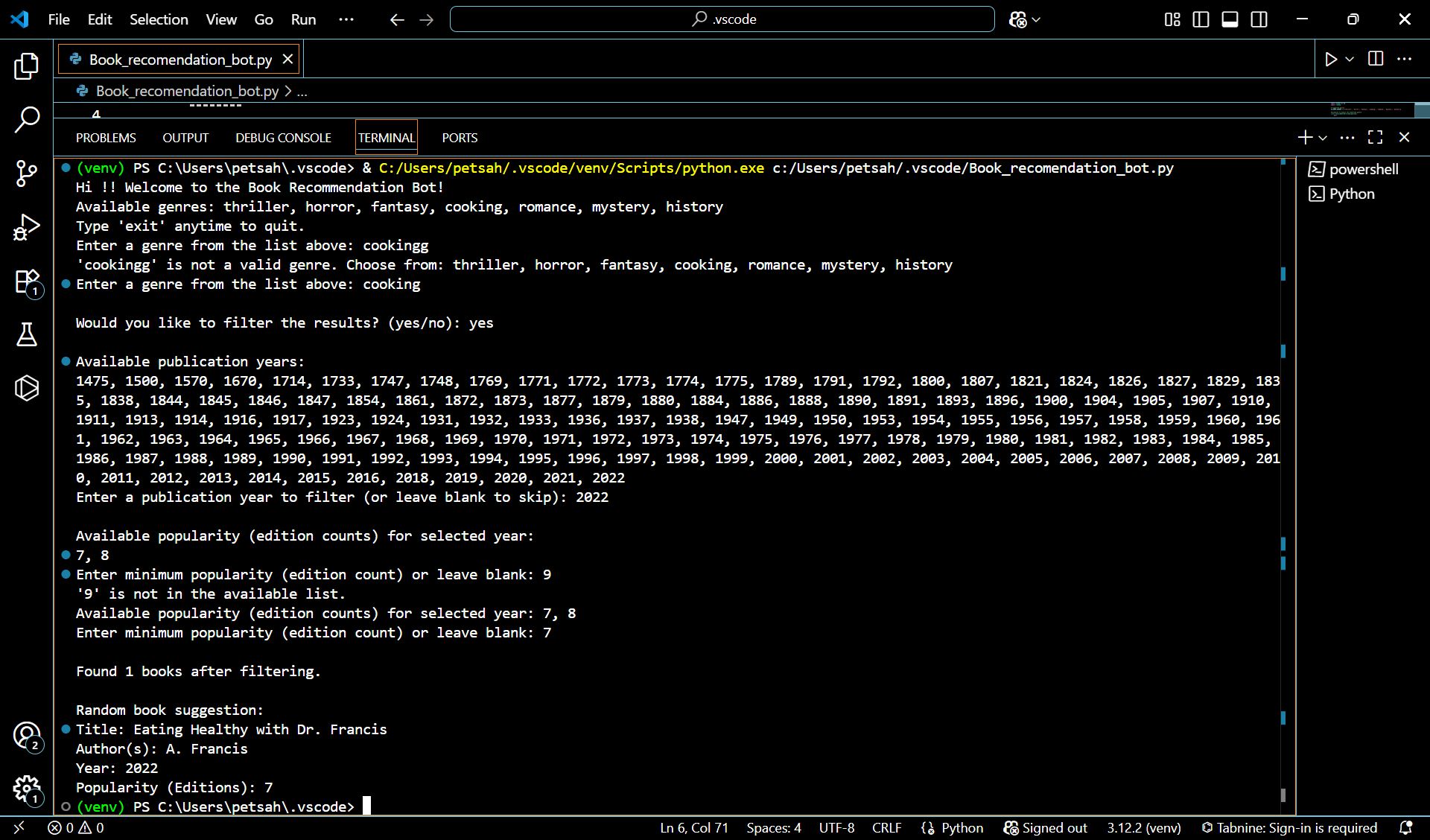
**EXAMPLE OUTPUTS:**

****

****

****

****

****

**Book\_recomendation\_bot.py**

**#CodeBase**

import requests

import pandas as pd

import random

# Suggested genres

ALLOWED\_GENRES = ['thriller', 'horror', 'fantasy', 'cooking', 'romance', 'mystery', 'history']

#exception to handle the undefined genre

class InvalidGenreError(Exception):

pass

# API URL

API\_URL = "https://openlibrary.org/subjects/{genre}.json?limit=300"

# Function to fetch books dataset based on genre

def fetch\_books\_by\_genre(genre):

genre = genre.lower()

#If user selected the Invalid Genre

if genre not in ALLOWED\_GENRES:

raise InvalidGenreError(f"'{genre}' is not a valid genre. Choose from: {', '.join(ALLOWED\_GENRES)}")

# To get the Data based on user selected Genre

try:

response = requests.get(API\_URL.format(genre=genre))

response.raise\_for\_status()

data = response.json()

books = data.get("works", [])

book\_list = []

for book in books:

book\_info = {

"title": book.get("title", "N/A"),

"authors": ", ".join([author['name'] for author in book.get("authors", [])]),

"genre": genre,

"first\_publish\_year": book.get("first\_publish\_year", "N/A"),

"edition\_count": book.get("edition\_count", 0),

"cover\_edition\_key": book.get("cover\_edition\_key", ""),

"key": book.get("key", "")

}

book\_list.append(book\_info)

df = pd.DataFrame(book\_list)

df.to\_csv(f"{genre}\_books.csv", index=False)

return df

except requests.exceptions.RequestException as e:

print(f"Error fetching data: {e}")

return pd.DataFrame()

#Function to filter books based on Year & Popularity

def filter\_books(df, year=None, popularity=None):

try:

if year:

df = df[df['first\_publish\_year'] == year]

if popularity:

df = df[df['edition\_count'] == popularity]

return df

except Exception as e:

print(f"Error filtering data: {e}")

return df

# Function to return a random book from filtered list

def random\_book\_suggestion(df):

try:

if df.empty:

return "No books available to suggest."

book = df.sample(n=1).iloc[0]

return (

f"Title: {book['title']}\n"

f"Author(s): {book['authors']}\n"

f"Year: {book['first\_publish\_year']}\n"

f"Popularity (Editions): {book['edition\_count']}"

)

except Exception as e:

return f"Error generating suggestion: {e}"

#main function to call the sub-functions according to user inputs

def main():

print("Hi !! Welcome to the Book Recommendation Bot!")

print("Available genres:", ", ".join(ALLOWED\_GENRES))

print("Type 'exit' anytime to quit.")

df = pd.DataFrame()

while df.empty:

genre = input("Enter a genre from the list above: ").strip().lower()

if genre == 'exit':

print("Thanks for using the Book Recommendation Bot.")

return # Exits the program

try:

df = fetch\_books\_by\_genre(genre)

if df.empty:

print(f"No data found for genre '{genre}'. Try another genre.")

except InvalidGenreError as e:

print(e)

continue

filter\_choice = input("\nWould you like to filter the results? (yes/no): ").strip().lower()

year = None

popularity = None

if filter\_choice.lower() == 'exit':

print("Exiting as requested.")

return

if filter\_choice == 'yes':

# Show available years

available\_years = sorted(df['first\_publish\_year'].dropna().unique())

print("\nAvailable publication years:")

print(", ".join(str(int(y)) for y in available\_years if str(y).isdigit()))

# Validate year input

while True:

year\_input = input("Enter a publication year to filter (or leave blank to skip): ").strip()

if year\_input.lower() == 'exit':

print("Exiting as requested.")

return

if not year\_input:

break

try:

year = int(year\_input)

if year in available\_years:

break

else:

print(f"'{year}' is not in the available list.")

print("Available years:", ", ".join(str(int(y)) for y in available\_years))

except ValueError:

print("Invalid input. Please enter a number.")

# Only ask for popularity if year is selected

if year:

year\_df = df[df['first\_publish\_year'] == year]

year\_popularity = sorted(year\_df['edition\_count'].dropna().unique())

if year\_popularity:

print("\nAvailable popularity (edition counts) for selected year:")

print(", ".join(str(int(p)) for p in year\_popularity))

while True:

popularity\_input = input("Enter minimum popularity (edition count) or leave blank: ").strip()

if popularity\_input.lower() == 'exit':

print("Exiting as requested.")

return

if not popularity\_input:

break

try:

popularity = int(popularity\_input)

if popularity in year\_popularity:

break

else:

print(f"'{popularity}' is not in the available list.")

print("Available popularity (edition counts) for selected year:", ", ".join(str(int(y)) for y in year\_popularity))

except ValueError:

print("Invalid popularity input. Skipping popularity filter.")

# Apply filters

filtered\_df = filter\_books(df, year=year, popularity=popularity)

print(f"\nFound {len(filtered\_df)} books after filtering.")

# Show random book suggestion

print("\nRandom book suggestion:")

print(random\_book\_suggestion(filtered\_df))

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

main()