Advanced Topics in Computer Science I (420-G40-HR)
Assignment 3 – Graphics and Web Scraping

Date assigned:	Friday October 17, 2025
Date due:	Monday November 10, 2025 @11:59PM

Learning Objectives

Upon successful completion of this assignment, the student will be able to:

- Use the turtle, tkinter and numpy libraries, and
- Scrape data from a website.

Marking Scheme

Question	**Out Of**
Part A - 4 In a Row Functionality	
Setup of game	5
Pause, Quit, Reset work as expected	6
Draw grid using turtle	8
Fill grid properly with disks	6
Run simulation – win detected on row/column/diagonal – draw detected on tie	10
Tests for all functionality	8
Part B Movie Trivia Game Functionality	
Scraping for top 25 movies	12
Scraping individual movie links	10
Scraping trivia	8
Storing/manipulating movie data	8
Creating questions from data	10
Randomly selecting questions	5
CLI gameplay	10
Unit tests for all functionality	20
Pythonic Coding	
Uses comprehensions	4
Uses classes and magic methods	6
Uses exceptions for control	6
Broken into functions with one purpose per function	4
Proper PEP 8 standard coding	4
Self-Assessment Complete	5
Total	155

Read the following instructions CAREFULLY. Be sure to TEST ALL YOUR CODE
Ensure all code is Pythonic and follows PEP-8 standards. Remember to complete and submit your self-reflection at the end.

Part A – Graphics - 4-In-A-Row

1) Create a Python program called ***four_in_a_row.py*** which runs a simulation and determines if there is a winning condition in a Four-In-A-Row game. (This means that the game runs automatically, through to completion. I will show a demo in class.). It works like this:
2) Draw a 6x7 (6 rows, 7 columns) grid of circles using ***turtle***.
3) Randomly add red and yellow disks to the grid by dropping them in the top and letting them "fall" as far as they can (you do not have to show the animation).
4) After each disk has landed, check if there is a winner and if there is, stop the simulation and report the winner to the (turtle) screen. A win occurs if there are four circles of the same colour either horizontally, vertically or diagonally.
5) Continue randomly playing disks of alternating colours until there is a winner or the grid is full. If the grid fills up, report "Draw" to the (turtle) screen.
6) The user can start the simulation using 's', pause the simulation using 'p', quit using 'q' and start over using 'r'.
7) You must use the **turtle** and **tkinter** libraries for the graphics. Use the pigs and chickens example from class as a guide.
8) You must use classes for this program, and at a minimum have an __init__ method for each class. Use magic functions appropriately for any classes you create.
9) You may use the numpy library if you would like to learn it, to give you a two-dimensional array for the board.
10) Make sure that you use the efficient methods for outputting strings with the f' format and comprehensions for all lists wherever possible.
11) Do not write massive blocks of code. Each block of code that performs a specific task should be in a function! Remember singularity of purpose.
12) Use exception handling with try/except/else blocks
13) Think Pythonically. That is, follow the guidelines established in the PEP 8 Styling Guide.

Part B: Movie Trivia Scraper and Game

In Part B of this assignment, you will use **Python** to build a **command-line trivia game** by scraping IMDb for information about popular movies. This part of the assignment combines web scraping, data handling, and game logic.

Objective

1. **Scrape IMDb Top Movie Data:** Start from the IMDb Top 250 Movies page.
2. **Extract Top Movies:** Download and parse the HTML data and follow links for each movie until data for 25 of the top 250 movies are extracted
3. **Scrape Each Movie's Details:** Visit each movie's details page and extract specific information.
4. **Collect Trivia:** Follow the **Trivia link** from the movie's details page and save the trivia information.
5. **Build a Trivia Game:** Use the scraped data to create a fun movie trivia game that can be played in the terminal.

Tasks

1. Web Scraping

1. **Start from the given movie URL:**
   Scrape the **IMDb Top 250 Movies** list starting from the provided IMDb Top 250 Movies URL.
   - Find the **href link** to the Top movies list within this page.
   - Access the Top 250 movies list and extract **the top 25 movies**.
2. **Scrape Movie Details:**
   - For each movie in the top 25, follow the **movie's href link** to its details page.
   - Extract the following information from each movie's page:
     - **Title**
     - **Year of release**
     - **IMDb rating**
     - **Director name**
     - **Main Cast Members** -- Don't bother with the entire cast, just the top cast
3. **Scrape Trivia:**
   - On each movie's details page, find and follow the **Trivia link**.
   - Extract and save at least **three trivia facts** per movie.

2. Game Logic

1. **Designing the Command-Line Game:**
    - Ask the player **five random questions** from the movie data.
    - Use **different types of questions** (for example, guessing the year of release, director, or an actor).
    - You MUST include at least 1 **trivia-based question** to make the game more interesting.
2. **Scorekeeping:**
    - Track the player's score based on correct answers.
    - Display the final score at the end of the game.

---

3. Starter Code

I have started you off with starter code for this assignment. You can find it in `imdb_html_parser.py`. The important part is the `headers` which ensures the request is successful. NOTE that this is NOT the final parsing code - you will need to add more to this. Remember to code Pythonically and remember the Clean Code Principles.

4: Sample Output, Gameplay, Minimum Requirements

Below is a sample output to help you understand how the game should work. This is just an example—you are free to improve or modify the game design as you see fit! Just ensure you include the following minimum requirements:

a) Game MUST use CLI with user inputs
b) Each run of the game MUST have 5 RANDOMLY generated questions based off the top 25 movies data you scraped. Note that the example run only has 4 questions instead of 5!
c) You MUST have at least 1 trivia-based question, and at least 1 question based on the movie details data in each run of the game
d) You MUST format your questions as a readable/answerable game question, you cannot simply parse the website and put raw data for the questions.
e) You MUST give instant feedback after the user inputs their answer for each question

f) You MUST give a final score at the end of the game run, and allow the user to choose to play again or exit
g) When the user plays again, it must NOT ask the same questions of the previous game run.

```bash $ python movie_trivia_game.py

🎬 Welcome to the Movie Trivia Game! 🎬 Try to answer questions about some of the top IMDb movies.

Question 1: When was the movie 'The Godfather' released?

1973 ❌ Wrong! The correct answer is 1972.

Question 2: Who directed the movie 'The Dark Knight'?

Christopher Nolan ☑ Correct!

Question 3: Based on the following piece of trivia answer which movie it is from: "Morgan Freeman's favorite film of his own."

The Shawshank Redemption ☑ Correct!

Question 4: Who starred in the movie 'Pulp Fiction'?

John Travolta ☑ Correct!

🎉 Game Over! Your final score is 3/4 🎉 Would you like to play again? Enter Y to play again or N to exit:
```

To submit

Submit a ZIP file (*YourUserName_G40_A3.zip*) containing all submitted files (including the self-assessment, all programs, and all test code) on Moodle.