Assignment 1:
Reflexive Web Agent with Tools Use

Student ID: 111502562, Name:陳伯榕

1. **(10%) Describe your Agentic AI application scenario, including target users, use cases, and problems to be solved.**
   Provide a comprehensive description of your application with at least 2-3 specific use cases, clearly defined target users, and concrete problems to be solved.
   Target Users:
   The target users for this application are:

1. **Beginners in Programming:** Individuals who are just starting to learn programming and need guidance on what resources to use and how to structure their learning.
2. **Self-learners:** Individuals who prefer online courses and want a structured learning plan to guide their journey.
3. **Students and Professionals Looking to Upskill:** Those who want to improve their programming skills in specific languages (like Python) or concepts.

Use Cases:

1. **Use Case 1: Personalized Programming Learning Plan Generation**
   - **Description:** A user selects a programming language, and the system fetches relevant programming courses from YouTube and generates a personalized 10-week study plan based on those courses. The study plan includes suggested resources, weekly goals, and tasks.
   - **Target User:** Beginners or self-learners who need structured learning paths.
   - **Problem Solved:** Many new learners struggle with choosing the right courses or knowing how to structure their learning. The system automatically selects resources and organizes them into a clear, actionable plan.
2. **Use Case 2: Custom Course Search for Advanced Learners**
   - **Description:** Advanced users, such as professionals or students, input specific topics or skills they want to focus on (e.g., data structures, algorithms). The system then retrieves related advanced courses and generates a tailored learning path.

    - **Target User:** Intermediate to advanced learners who want to deepen their knowledge in specific programming concepts.
    - **Problem Solved:** Advanced learners often lack time to search for and evaluate various learning resources. This system automates the search and suggests courses that match their advanced learning goals.

Problems to Be Solved:

- **Finding Relevant Courses:** Users may not have the time or expertise to search for and evaluate courses. The agent automates this by fetching YouTube courses based on the user’ s query.
- **Personalized Study Plans:** Users often struggle with creating a structured study plan. The system generates a 10-week, detailed study plan for learning programming, based on available resources.

2. (10%) Analyze at least 2 potential technical challenges in implementation and propose preliminary solutions.

For each technical challenge, provide detailed analysis including impact assessment and step-by-step solution proposals with feasibility evaluation.

Challenge 1: Efficient and Accurate Web Scraping

- **Description:** Web scraping is central to the agent's ability to fetch YouTube courses. However, YouTube has anti-scraping measures such as CAPTCHA and dynamic page loads. This could limit the agent’ s ability to consistently collect up-to-date information.
- **Impact Assessment:** Without reliable course data, the quality of the generated learning plans will suffer, as the agent may fail to retrieve all relevant or the most recent courses.

Solution Proposal:

1. **Using Headless Browsers (Selenium):** The current implementation uses Selenium with headless Chrome to simulate human browsing behavior, bypassing simple bot protections.
2. **Anti-scraping Mitigation:** Implementing rotation of user-agent headers, proxies, and IP addresses will help to avoid detection by YouTube’ s anti-bot measures.
3. **Fallback Mechanism:** In case scraping fails, the system could try using an API (if available) for YouTube courses or use an alternate web scraping technique like Puppeteer.

Feasibility Evaluation: Using headless browsers and rotating proxies is feasible, but it may add complexity and slow down the scraping process.

However, this can be mitigated by optimizing wait times and error-handling mechanisms.

3. (20%) Explain how your system implements the complete cycle of environment perception, decision making, and action execution.

Detail the complete workflow of your system, demonstrating how each component interacts within the perception-brain-action cycle.

Cycle Overview:

The system follows a perception-decision-action cycle to provide users with personalized programming learning plans:

1. **Environment Perception:**
   - The agent perceives the environment by gathering user input (subject or programming language) and browsing the web for relevant courses (YouTube).
   - The web scraping process collects relevant videos by searching for the user’ s requested programming topic.
2. **Decision Making:**
   - The agent processes the user input (subject) and the data gathered from web scraping. It then generates a detailed study plan using a language model (Hugging Face) that synthesizes the courses and organizes them into a 10-week learning path.
   - The decision-making process is driven by a natural language processing (NLP) model, which organizes and refines the data to create a personalized plan for the user.
3. **Action Execution:**
   - The system outputs the study plan to the user, displaying a list of relevant courses and a structured learning timeline.
   - It also displays the course titles, URLs, and descriptions, enabling the user to access the courses directly from the plan.

4. **(30%) Design and execute 3 test tasks, analyze the results, and propose potential improvements based on the current implementation.**

Document the execution of three test cases with comprehensive analysis of results and specific improvement suggestions.

Test Case 1: Search for Python Programming Courses

- **Description:** Test whether the agent can successfully search for Python programming courses on YouTube and generate a relevant 10-week study plan.
- **Expected Outcome:** The system should display at least 5 relevant Python courses and a study plan that includes specific weekly tasks.
- **Results:** The system successfully fetched 7 Python courses and created a 10-week study plan. However, the plan was slightly too general, lacking specifics like project suggestions.

Improvements:

- Integrate more personalized suggestions based on the learner's skill level and preferred learning style.
- Include projects or hands-on tasks in the study plan to make it more engaging.

Test Case 2: Search for Advanced Data Structures Courses

- **Description:** Test the system's ability to find advanced courses on data structures and generate a learning path for intermediate learners.
- **Expected Outcome:** The system should fetch courses on data structures, such as algorithms, trees, and graphs, and create a detailed study plan.
- **Results:** The system found relevant courses, but the study plan lacked adequate depth for an advanced learner.

Improvements:

- Use advanced filtering criteria for more focused searches, such as course difficulty level or topic depth.
- Enhance the model's ability to generate content that caters to users with prior knowledge or experience.

Test Case 3: Handling User Input for a New Language (e.g., JavaScript)

- **Description:** Test how well the agent handles a new input like JavaScript and generates a customized study plan.
- **Expected Outcome:** The system should adapt to the new subject, find relevant courses, and provide a new, actionable learning plan.
- **Results:** The system generated a relevant study plan with 5 JavaScript courses. However, the structure was too similar to other languages and didn’t emphasize JavaScript-specific features.
  Improvements:
- Customize the study plans more deeply by considering the unique aspects of each language.
- Add more dynamic input processing that recognizes specific language features and includes those in the plan (e.g., asynchronous programming in JavaScript).