Lars Yarkosky, Abby Gerstner, Semyon Tsyrenov

CS 291

Mini Project 2 - Course Prerequisite Graph

The "Academic Program Analyzer" consists of a Python program utilizing the tkinter and networkx libraries to create a graphical user interface (GUI) for managing an academic program's courses and prerequisites. The core functionality is shown within the AcademicProgram class, which initializes an empty directed graph to represent the courses and their relationships. The class provides methods to add courses with prerequisites, identify courses without prerequisites, find courses with the most and least prerequisites, and check for circular dependencies within the program.

The GUI includes entry fields for course and prerequisite information, along with buttons to perform operations such as adding courses, finding courses without prerequisites, finding courses with the most and least prerequisites, and checking for circular dependencies. These buttons trigger corresponding methods from the AcademicProgram class, and the results are displayed in a labeled section of the GUI. The event-driven nature of tkinter is used through functions like add_course() and event handlers associated with each button, providing an interactive experience for the user.

In the add_course function, a new node of the graph is added, and an edge will be added to the node to represent each of its prerequisites. In the courses_without_prerequisites function, it will check to see which node/s have no edges, which represent prerequisites. For the courses_with_most_prerequisites function, it will check to see which node/s have the most edges. For the courses_with_least_prerequisites function, it will check to see which node/s have the least edges. Lastly, for the has_circular_dependencies function, it will make sure that no nodes are dependent upon each other.

Upon execution, the program creates an instance of the AcademicProgram class named academic_program and initializes a tkinter window titled "Academic Program Analyzer." The various GUI components, including labels, entry fields, buttons, and result labels, are organized within the

window. The program enters the tkinter event loop (window.mainloop()), allowing the user to interact with the GUI and perform operations on the academic program data.

The "Academic Program Analyzer" is designed to facilitate the management and analysis of academic courses and their prerequisites through a user-friendly graphical interface. Here's a step-by-step guide on how to use the provided code:

1. Launch the Program:
   - Run the Python script containing the provided code.
   - A tkinter window titled "Academic Program Analyzer" will appear.

2. Add Courses:
   - Input the name of a course in the "Course" entry field.
   - Specify the prerequisites for the course in the "Prerequisites" entry field, separating multiple prerequisites with commas.
   - Click the "Add Course" button to add the course and its prerequisites to the academic program.

3. Find Courses without Prerequisites:
   - Click the "Courses without prerequisites" button to display a list of courses that do not have any prerequisites in the result label.

4. Find Courses with Most Prerequisites:
   - Click the "Courses with most prerequisites" button to identify courses with the highest number of prerequisites. The result will be displayed in the result label.

5. Find Courses with Least Prerequisites:
   - Click the "Courses with least prerequisites" button to find courses with the fewest prerequisites. The result will be displayed in the result label.

6. Check for Circular Dependencies:

- Click the "Check for Circular Dependencies" button to determine whether the academic program has circular dependencies. The result will indicate if circular dependencies are found or not.

7. Repeat as Needed:

- You can continue adding courses and performing analyses as needed. The GUI provides real-time feedback in the result label for each operation.

8. Close the Program:

- Close the tkinter window when you are finished using the program.

The code combines the functionality of the AcademicProgram class with a user-friendly GUI, allowing for easy interaction with the academic program data. It offers a convenient way to manage course information, identify courses with specific characteristics, and check for potential issues like circular dependencies.