Contents

[INTRODUCTION: 2](#_Toc150798301)

[SEMANTIC NET IN AI: 2](#_Toc150798302)

[NetworkX 2](#_Toc150798303)

[Plotly 2](#_Toc150798304)

[How they work together 2](#_Toc150798305)

[DESIGN SEMANTIC NET: 3](#_Toc150798306)

[Components of a Semantic Net: 3](#_Toc150798307)

[SEMANTIC NET: 4](#_Toc150798308)

[EXAMPLE: PART OF SEMANTIC NET (Major 1): 4](#_Toc150798309)

[Code output of the same semantic net above: 5](#_Toc150798310)

[IMPLEMETATION: 5](#_Toc150798311)

[BASIC OPREATIONS: 5](#_Toc150798312)

[ADVANCED IMPLEMENTATION: 13](#_Toc150798313)

[THE OUTPUT PHOTO: 13](#_Toc150798314)

## INTRODUCTION:

AI has empowered machines to mimic human intelligence and perform complex tasks. Knowledge representation is a crucial aspect of AI, and semantic nets, also known as semantic networks, play a vital role in organizing information for machine understanding. But first, let's understand what a semantic net is.

## SEMANTIC NET IN AI:

Semantic net or semantic networks, it a type of knowledge representation tool that helps in visualizing and understanding the connections among different pieces of information within a domain. Graphical representations used to organize relationships between various entities.

## NetworkX

**Overview:** NetworkX is a Python library for creating, analyzing, and visualizing complex networks (graphs). It provides data structures for representing various types of graphs (e.g., directed, undirected, weighted) and algorithms for analyzing them.

**Key Features:**

**Graph Representation:** NetworkX allows you to create graphs and manipulate their nodes and edges. It supports various graph types, including directed and undirected graphs.

**Algorithms:** The library provides a wide range of algorithms for analyzing graphs, such as finding shortest paths, clustering coefficients, and centrality measures.

**Graph Drawing:** While NetworkX itself doesn't have built-in visualization capabilities, it can be easily integrated with other libraries (e.g., Matplotlib, Plotly) for graph visualization.

## Plotly

**Overview:** Plotly is a Python graphing library that enables interactive and visually appealing data visualizations. It supports a variety of chart types, including scatter plots, bar charts, line charts, and more. It also has modules for 3D plotting and geographical mapping.

**Key Features:**

**Interactive Visualizations:** Plotly allows users to create interactive plots with features like zooming, panning, and hover information. This is particularly useful for exploring complex datasets.

**Wide Range of Charts:** Plotly supports a variety of chart types, making it versatile for different data visualization needs.

**Integration:** Plotly can be easily integrated with other Python libraries like Pandas, NumPy, and NetworkX to create dynamic visualizations from various types of data.

## How they work together

**Creating a NetworkX graph:** You can use NetworkX to create and manipulate your graph, defining nodes, edges, and any additional attributes.

**Visualization with Plotly:** Once you have your graph defined in NetworkX, you can use Plotly to visualize it. Plotly provides functions and tools for creating interactive and aesthetically pleasing network visualizations. You may need to convert the NetworkX graph to a format that Plotly understands.

**Integration:** Plotly and NetworkX are not directly integrated, but you can easily use them together by converting a NetworkX graph to a format compatible with Plotly. For example, you might convert the graph to a Plotly Scatter plot with customized node positions and edges.

DESIGN SEMANTIC NET:  
For our assignment, we choose a semantic net about (the domain) Faculty of Computing and Information Technology.

Components of a Semantic Net:  
A semantic net can be defined as a directed graph consisting of nodes (entities) interconnected by labeled links (arcs or relationships). Each node represents a concept or entity, and each link represents a specific relationship or connection between the nodes.

So, in our project we used node as:  
1 Faculty (FCIT)

3 Major (CS, IT, IS)

-in code represented by 1 for CS, 2 for IT, 3 for IS

9 courses (each major have 3 courses)

-in code represented by 1,2, 3 for CS, 4,5, 6 for IT, 7, 8, 9 for IS

6 teachers (major 1(CS) has 3 teachers, major 2(IT) has 2 teachers, major 3(IS) has 1 teacher)

-in code represented by 1,2, 3 for CS, 4,5 for IT, 6 for IS

9 students (each major have 3 students)

-in code represented by 1,2, 3 for CS, 4,5, 6 for IT, 7, 8, 9 for IS

and links(relations) as a specific relationship between the nodes:  
Relation: Faculty 1 -> Major 1 (“Faculty” has major “Major”)

Relation: Teacher 1 -> Course 1 ("Teacher" teaches "Course")

Relation: Student 1 -> Course 1 (“Student” studies “Course”)

Relation: Major 1 -> Course 1 (“Major” has course “Course”)

Relation: Course 1 -> Course 2 ("Course" prerequisite "Course")

## SEMANTIC NET:

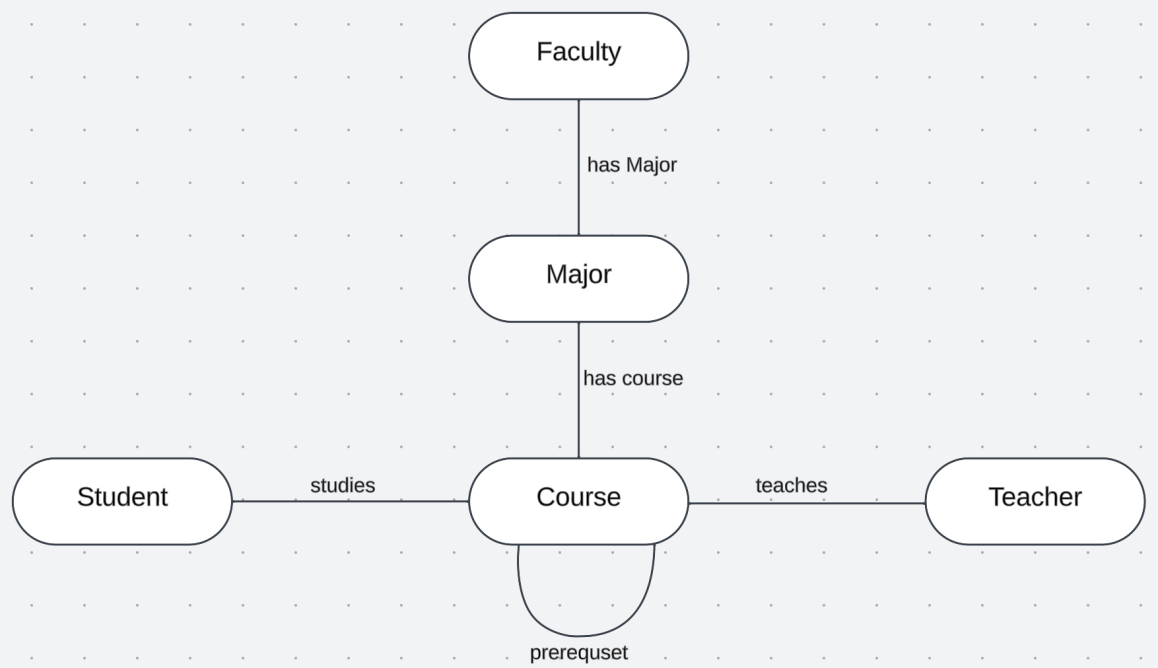
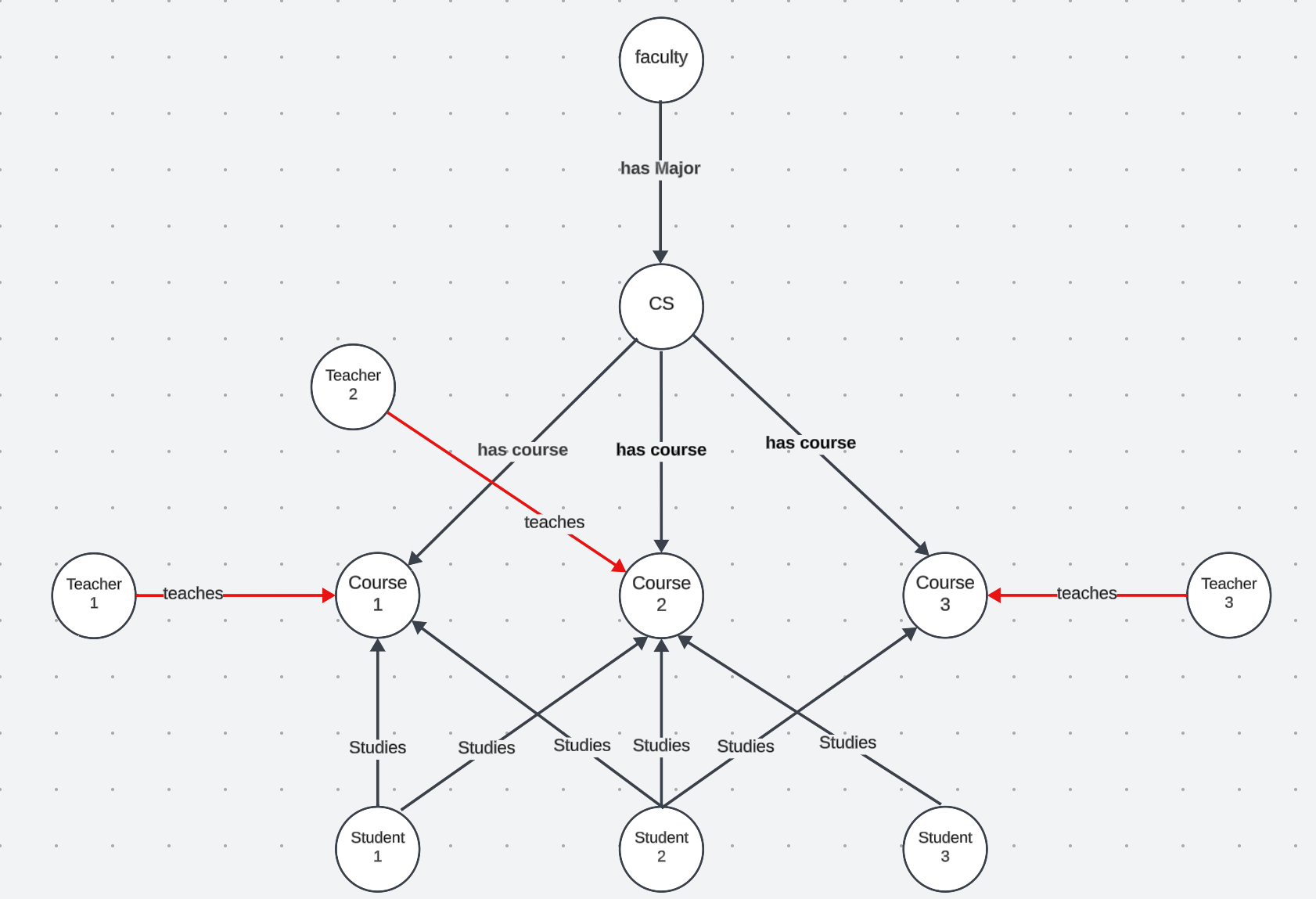


Figure 1- relation between nodes

## EXAMPLE: PART OF SEMANTIC NET (Major 1):



Major 1

Figure 2- example of semantic net

## Code output of the same semantic net above:

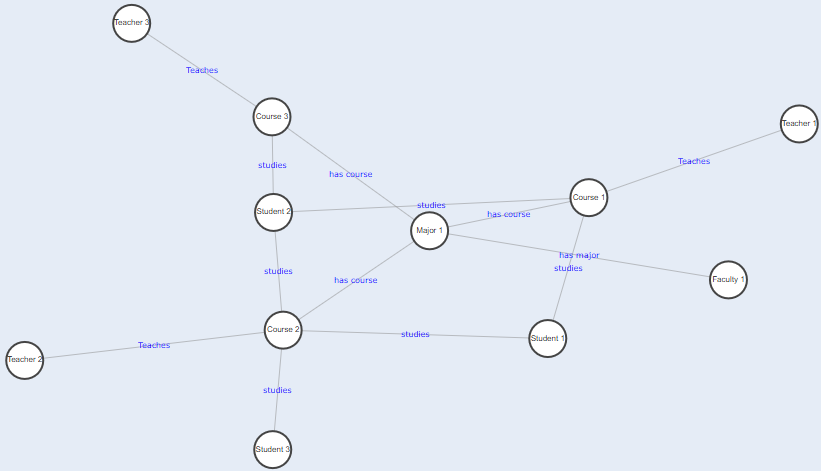


Figure 3- example of semantic net code outputted

## IMPLEMETATION:

We used python programming language because it contains many AI libraries and one of them helps us to implement semantic, we are using NetworkX library to help us visualize the graph and update it according to different operations.

THE CODE WILL BE PROVIDED WITH THIS REPORT

## BASIC OPREATIONS:

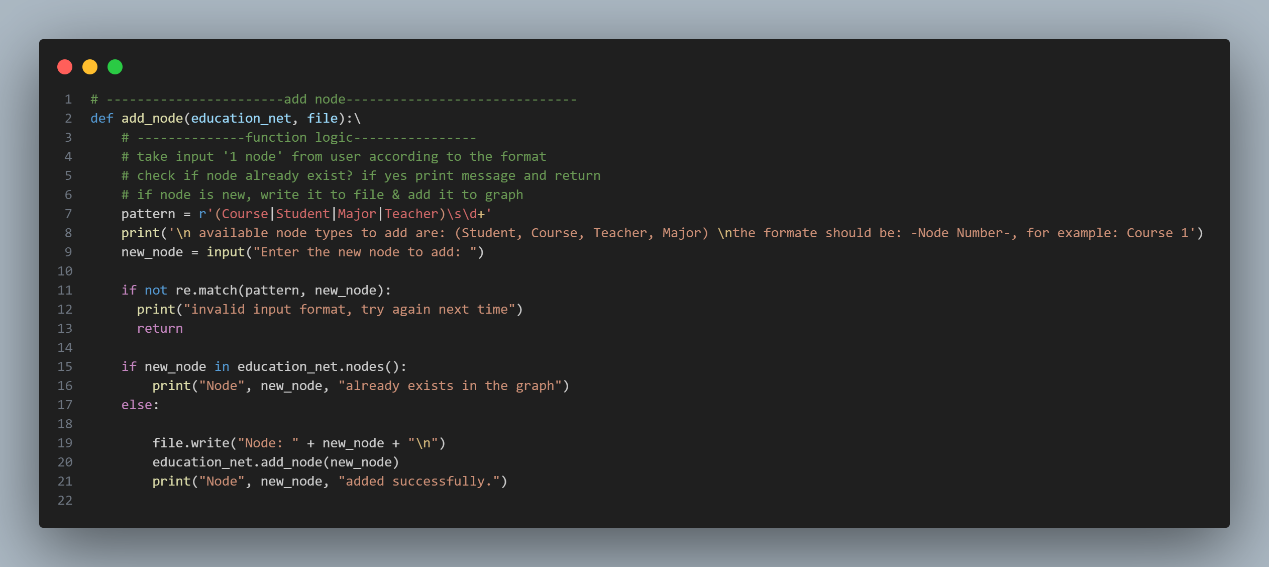
Add node:

Figure - code for add node method

Add node behavior:

1- Already existed

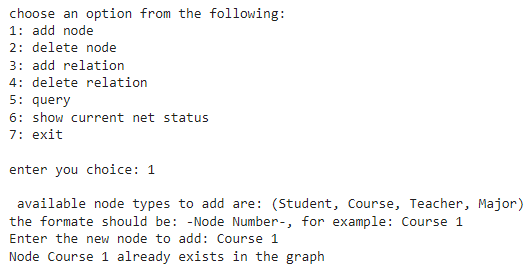


Figure 5- add node sample output 1

2- Invalid input

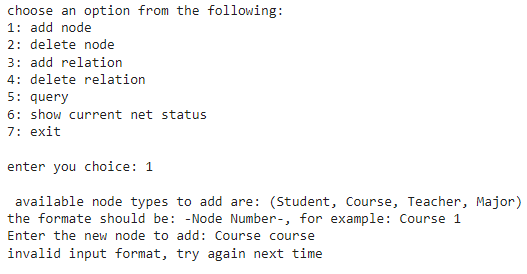


Figure 6- add node sample output 2

3- Node added successfully

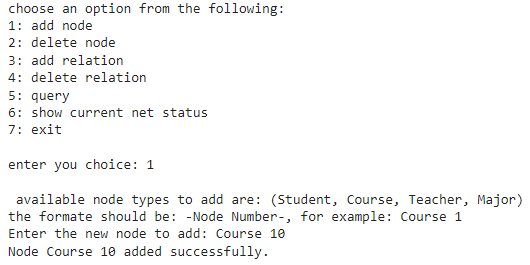
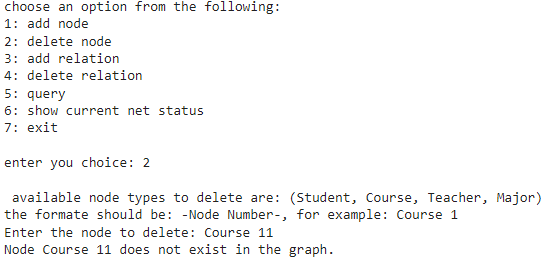


Figure 7- add node sample output 3

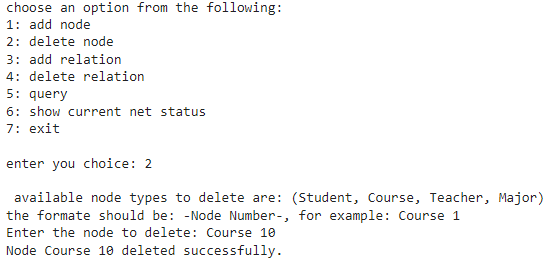
Delete node:  

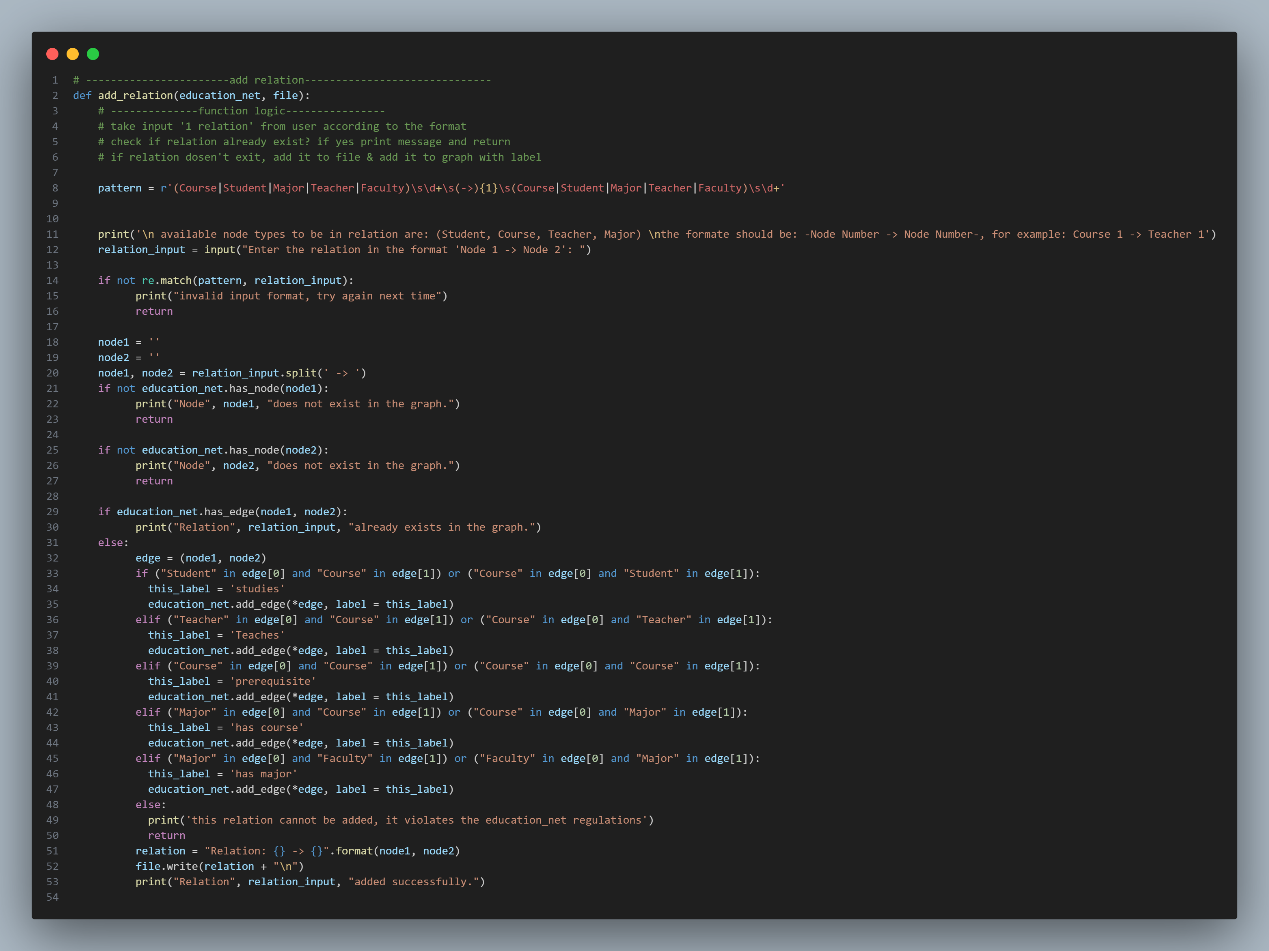

Delete node behavior:

1- Node doesn’t exist



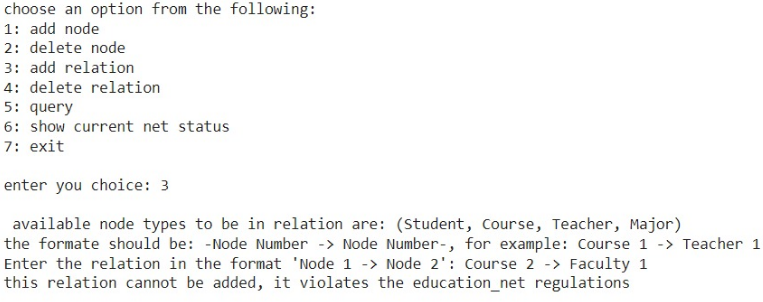
2- Node deleted successfully



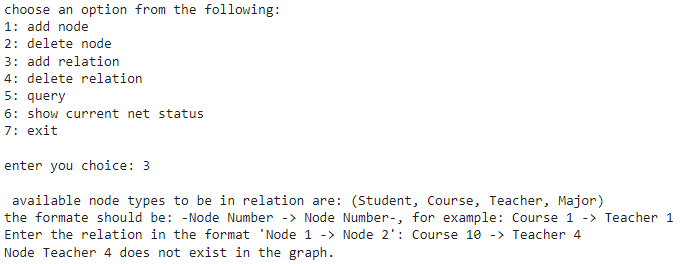
Add relation:  


Add relation behavior:

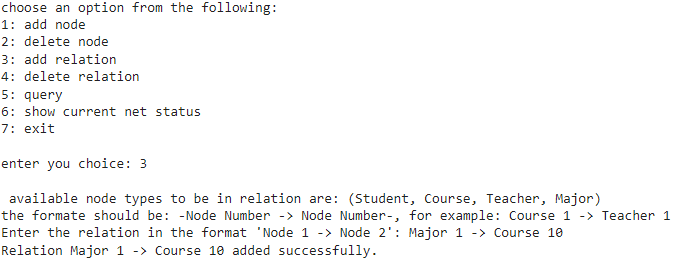
1- invalid format (or unaccepted relation according to policy)

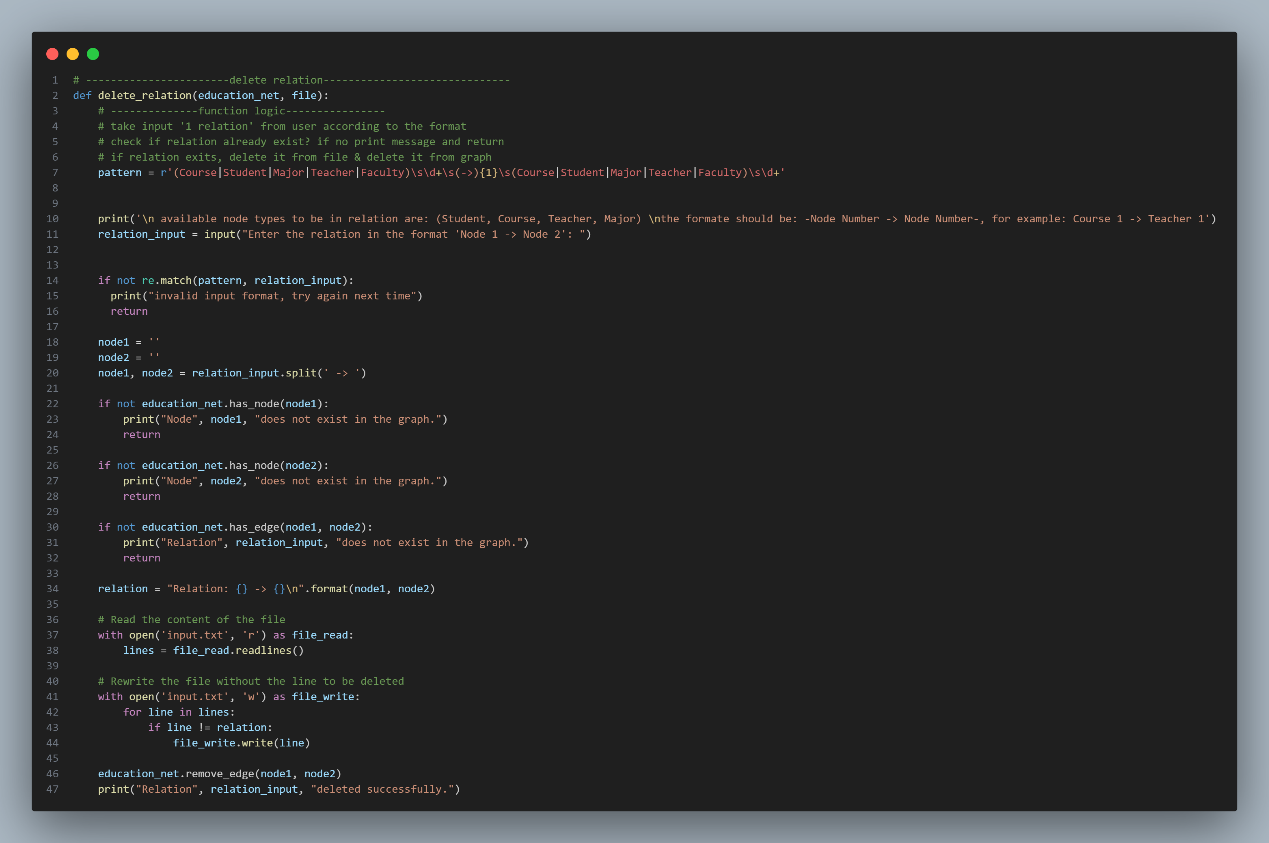


2- A node doesn’t exist



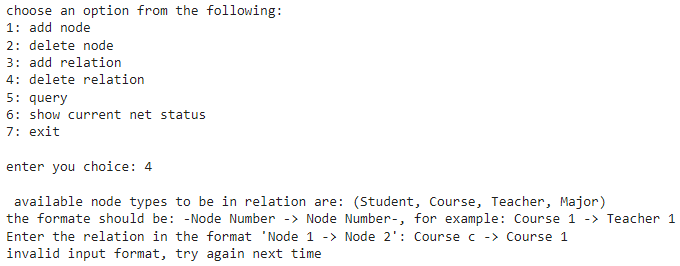
3- Added successfully



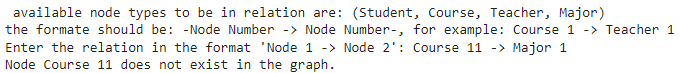
Delete relation:  


Delete relation behavior:

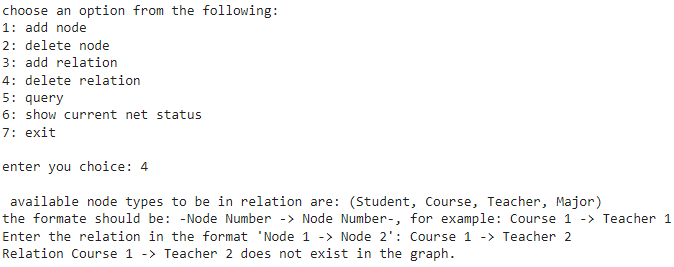
1- Invalid input



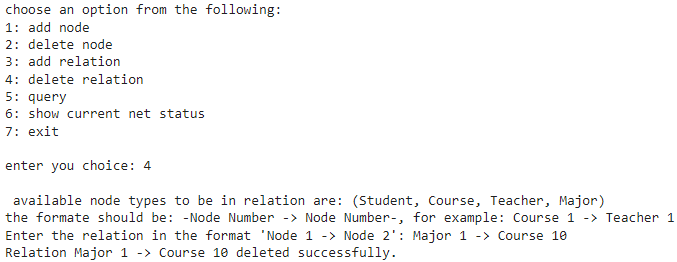
2- Node doesn’t exist

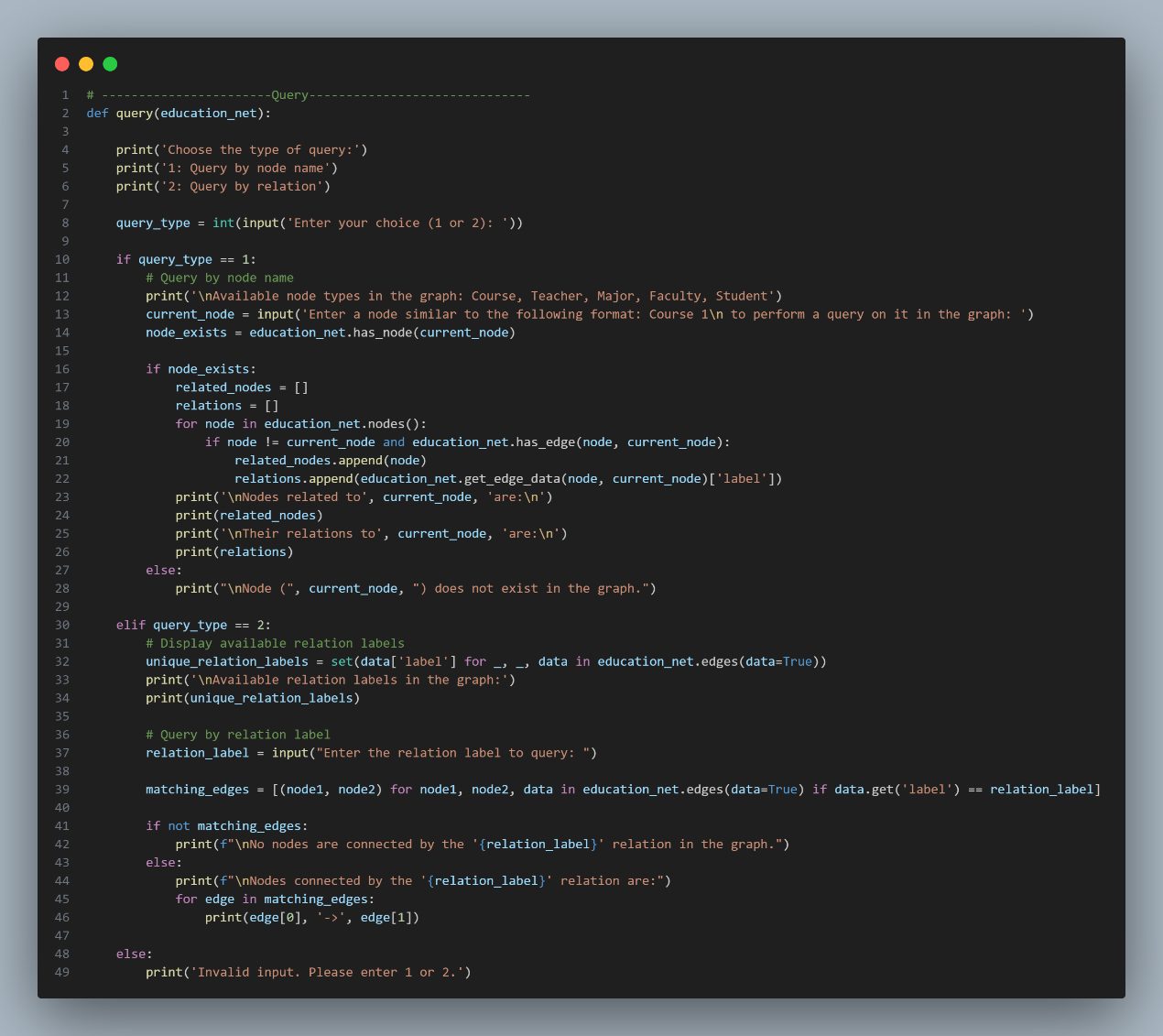


3- Relation doesn’t exist



4- Deleted successfully



Query:  


Here you need to specify the type of query.

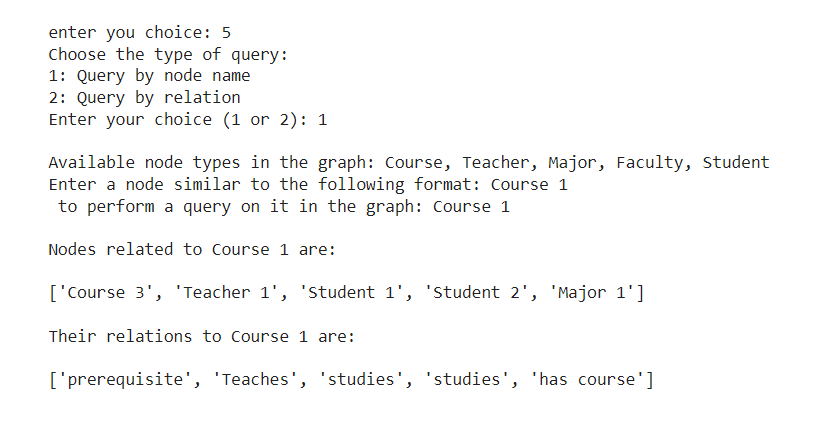
Here is an example:

A white screen with black text

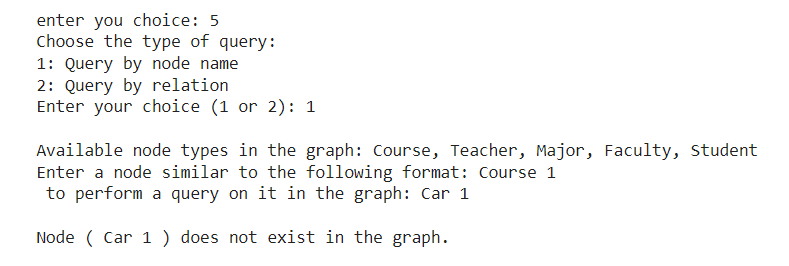
Description automatically generated

If we choose 1 you need to specify a node , then query method will return a list of all related nodes to the selected node with another list which shows the relation of all these nodes.

1. Available node.

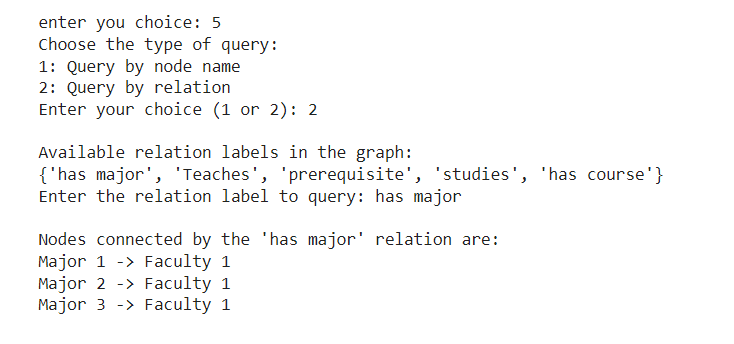


1. Unavailable node.

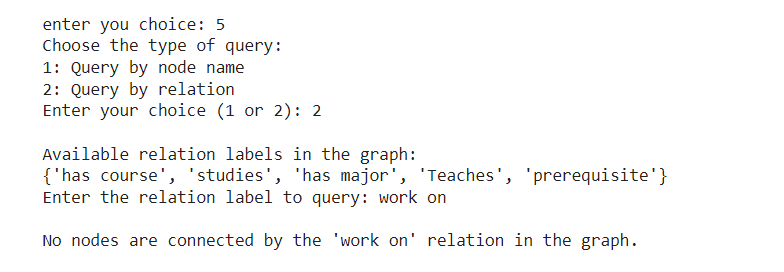


If we choose 2 you need to specify a relation , then query method will return a list of all nodes to the selected relation.

1. Avalibale relation.



1. Unavailable relation.



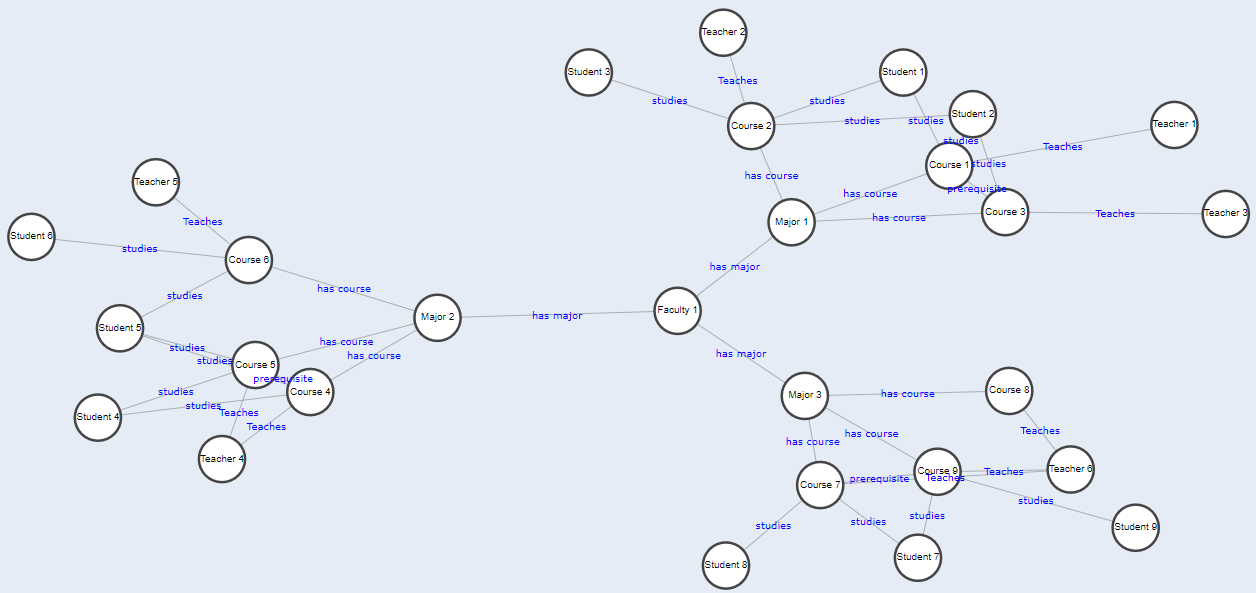
## ADVANCED IMPLEMENTATION:

Handle ambiguities or conflicts in the information:

The user can’t do any operation unless it follows the polices, like the user can’t add a relation that already defined between nodes, these regulations will eliminate duplicated and redundant relations, also we eliminate conflicts between nodes by using if statement to avoid having conflict information in semantic net, basically conflicts will not be accepted in the code, so no conflict will actually exist to be resolved later.

The method (show current net status), choice 6 in the menu of the program (shown in previous section), will visualize the net that is currently in the system, it will be shown first when the program reads data from the file and it will update after every operation and will be displayed whenever the user chooses choice 6, its output will be as the following in the first run:

## THE OUTPUT PHOTO:



Finally, the file that contains the base knowledge used to initialize the net is also modified and updated (rewritten) during the runtime, the next time the program is run, the semantic net will be as it was left at the end of previous run of the program.