COMP1005/1405 Fall 2024 Tutorial 8

Do not post the tutorial or the tutorial and/or solutions on any website.

Objectives

- Practice writing/running Python code in the VSCode
- Practice coding **functions and dictionaries**
- Further understanding of a graph being represented as a multi-dimensional list

Expectations

To receive full grades for this tutorial, you must complete Problems 1-2.

Grading Scheme for Tutorials

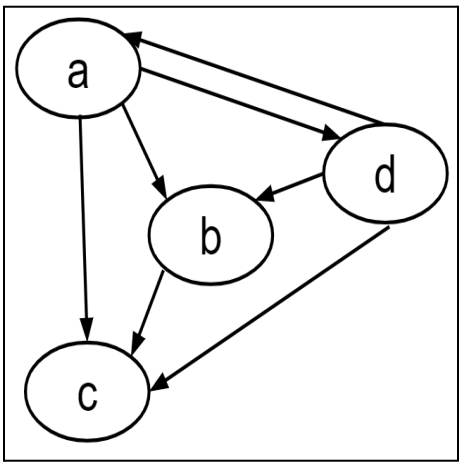
For each tutorial, you will be graded based on the following scale:

- 2/2 for demonstrating problems 1-2 and your **YourStudentNumber_T8.zip** file
    - Section A Tutorial Sessions: submit your work to the tutorial BrightSpace. The zip file should contain your solutions to all the required problems.
    - Section C, D, and E Tutorial Sessions: By the end of the tutorial session, demonstrate your tutorial work in person to a teaching assistant.
- 1/2 if you are missing problems or your solutions need significant improvement.
- 0/2 if you do not submit to BrightSpace or demonstrate to a teaching assistant
    - Section A Tutorial Sessions: no submission to the tutorial BrightSpace
    - Section C, D, and E Tutorial Sessions: not demonstrating your tutorial work to a teaching assistant by the end of the tutorial session

---

Problem 1 (Graph Theory and Multi-Dimensional Lists)

Write a Python file named **`adjacentML.py`** in VSCode to implement the representation of the graph in two ways: adjacency matrix and adjacent List.



1. Based on the provided graph, determine the following values:

Hint: Please refer to Lecture 13, slides 9-19.

a. Answer the following and put them in the comments of the program:
   - Graph = ?
   - Vertices = ?
   - Edges = ?

b. Represent the graph as local variables for the adjacency matrix and adjacency list (use the provided variable names):
   - adjacencyMatrix = ?
   - adjacencyList = ?

2. Create a function called **matrixEdges():**
   a. The function takes the adjacencyMatrix as the parameter and prints the edges for each vertex in the graph.
   b. For each vertex, the function should display all the edges coming out, with each edge represented by a line showing the connection between the vertex and its adjacent vertices.

Sample output:

```
a -> b;     a -> c;     a -> d
b -> c
c has no edges
d -> a;     d -> b;     d -> c;
```

3. Create a function called **listEdges():**
   a. The function takes the adjacencyList as the parameter and prints the edges for each vertex in the graph. Same requirements and output as above.

4. Create a main() function to run the matrixEdges() and listEdges() functions.

Problem 2 (Dictionary)

Write a Python file named **`listToDict.py`** in VSCode to implement a conversion from the multidimensional list to a dictionary.

1. Implement a function called **listToDictionary(aList).**
    a. This function takes a 2-dimensional list containing **sales records**, where each sub-list has the following format:
        i. **The first element:** A unique customer ID (string).
        ii. **The second element:** The product purchased by that customer (string).
        iii. **The third element:** The cost of the product purchased (float or integer).
    b. Convert the above list into a dictionary where:
        i. **Key:** Unique customer ID (string).
        ii. **Value:** A list containing:
            1. A list of all products the customer purchases (list of strings).
            2. The total cost of these products (float or integer).
    c. Return the dictionary.
2. Implement a function called **display(aDictionary).**
    a. The function takes a dictionary (the return value of listToDictionary()) as a parameter.
    b. The function should iterate through the dictionary and print each key-value pair (one pair per line).
    c. No return value is required.
3. Create a main() function to run the functions above using the sales variable as the multi-dimensional list, as seen below.

Sample Test & Output:

```
sales = [ ["customer1", "bread", 5], ["customer2", "bread", 4.5], ["customer1", "egg", 6.75],
       ["customer2", "milk", 4.35], ["customer3", "egg", 3.6], ["customer4", "bread", 4.5],
       ["customer1", "milk", 4.35], ["customer2", "egg", 3.6], ["customer4", "milk", 4.35] ]

record = listToDictionary(sales)

display(record) # prints the following

customer1 : [['bread', 'egg', 'milk'], 16.1]
customer2 : [['bread', 'milk', 'egg'], 12.45]
customer3 : [['egg'], 3.6]
customer4 : [['bread', 'milk'], 8.85]
```

Final Step

For Section A (Submit the work before the tutorial ends):

1. **Submit** your **zip** file to our Merged Tutorial Brightspace. The due date of your submission is aligned with your tutorial session.
2. **After** you submit the file, download your submission from Brightspace and confirm that it is a zip file containing **adjacentML.py, and listToDict.py.**
3. **Extract** the .py files and execute the extracted files again to ensure they work properly. Occasionally, a problem can occur during the upload process, and files can become corrupted.

For Sections C, D and E (Show the TAs your work before the tutorial ends):

1. **Problem 1-2:** Run your Python programs in VS Code to demonstrate they are working.
2. Answer the questions the TA may ask.
3. Show the TA your zip file and extract the files.