

|  |
| --- |
| **GRAPH ALGORITHMS** |
| ASSIGNMENT 01 – NetworkX Python |
| **MUHAMMAD HARRIS**  *BCS203193* |
|  |





Table of Contents

[QUESTION: 2](#_Toc118624279)

[GRAPH 1: 3](#_Toc118624280)

[Code of Graph1.py 3](#_Toc118624281)

[Outputs of Graph1.py 7](#_Toc118624282)

[Graph 2: 13](#_Toc118624283)

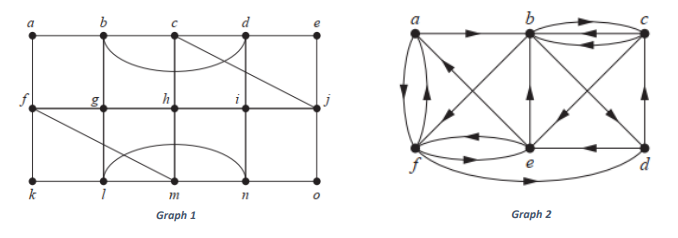
[Code of Graph2.py: 13](#_Toc118624284)

[Outputs of Graph2.py: 17](#_Toc118624285)

GRAPH ALGORITHMS

ASSIGNMENT 01 – NetworkX Python

# QUESTION:

For the given directed and undirected graphs:

Write the PYTHON program for each graph in which:

1. Display Nodes List of the Graph
2. Display Edge List of the Graph
3. Count Connected Components of the Graph
4. Print Connected Components of the Graph
5. Display Incidence Matrix of a Graph
6. Display the Nodes degrees
7. Count Number of Edges
8. Visualize the graphs
9. Check if:
   1. Euler circuit exist or not and Graph is Eulerian
   2. Euler path exists or not
   3. Hamilton Path
   4. Perform Depth First and Breadth First Traversal on graph

Coding Standard to Follow:

* For each graph create different Python Script (.py) will all the functionalities to implement
* There should be proper menu for user to perform the task
* Use proper comments in code for understanding
* Print output of each task with proper messages

## GRAPH 1:

### Code of Graph1.py

# MUHAMMAD HARRIS - BCS203193

# Graph Algorithm Assignment 1

# GRAPH 1

# libraries

import os

import networkx

import matplotlib.pyplot as plot

# functions

def printMenu():

    print("""|-------------------------------- MENU --------------------------------|

1. Display Nodes List of the Graph

2. Display Edge List of the Graph

3. Count Connected Components of the Graph

4. Print Connected Components of the Graph

5. Display Incidence Matrix of a Graph

6. Display the Nodes degrees

7. Count Number of Edges

8. Visualize the graphs

9. Check if:

    a. Euler circuit exist or not and Graph is Eulerian

    b. Euler path exists or not

    c. Hamilton Path

    d. Perform Depth First and Breadth First Traversal on graph""")

def printNodeList(graph):

    os.system('cls')

    nodeList = graph.nodes() # get node list of graph

    print('Node List of Graph:')

    for x in nodeList:

        print(x, end=', ') # print each node

    input('\n\npress enter to return...')

def printEdgeList(graph):

    os.system('cls')

    edgeList = graph.edges() # get edge list of graph

    print('Edge List of Graph:')

    for x in edgeList:

        print(x[0], '<->', x[1], end=', ') # print each edge

    input('\n\npress enter to return...')

def printCountConnectedComponents(graph):

    os.system('cls')

    count = networkx.number\_connected\_components(graph) # get number of connected components

    print('Number of Connected Components in Graph: ', count) # print count

    input('\npress enter to return...')

def printConnectedComponents(graph):

    os.system('cls')

    connectedComponents = [len(c) for c in sorted(networkx.connected\_components(graph), key=len, reverse=True)] # get connected components

    print('Length of Connected Components in Sorted Order:')

    for x in connectedComponents:

        print(x) # print length of each connected component

    input('\npress enter to return...')

def printIncidenceMatrix(graph):

    os.system('cls')

    incidenceMatrix = networkx.to\_numpy\_matrix(graph) # get incidence matix

    incidenceMatrixString = str(incidenceMatrix).replace('[','').replace(']','').replace('.','')

    print('Incidence Matrix of Graph: ')

    print('',incidenceMatrixString) # print incidence matrix

    input('\npress enter to return...')

def printNodeDegrees(graph):

    os.system('cls')

    print('Degree of all Nodes in Graph:')

    nodeN = 'a'

    for x in range(15):

        print(nodeN, ' = ', graph.degree(nodeN)) # print degree of each node

        nodeN = chr(ord(nodeN)+1)

    input('\npress enter to return...')

def printCountEdges(graph):

    os.system('cls')

    print('Number of Edges in Graph: ', networkx.number\_of\_edges(graph)) # print number of edges in graph

    input('\npress enter to return...')

def visualizeGraph(graph):

    os.system('cls')

    networkx.draw(graph, with\_labels = True) # draw graph with labels

    plot.show()

    input('press enter to return...')

def printIsEulerian(graph):

    os.system('cls')

    if networkx.is\_eulerian(graph): # check if graph is euler or not

        print('a. Graph is Euler Graph.')

    else:

        print('a. Graph is not Euler.')

def printHasEulerPath(graph):

    if networkx.has\_eulerian\_path(graph): # check if graph has euler path or not

        print('b. Eulerian Path or Circuit Exists.')

    else:

        print('b. Eulerian Path or Circuit does not Exist.')

def printHamiltonPath(graph):

    try:

        hamiltonPath = networkx.algorithms.tournament.hamiltonian\_path(networkx.DiGraph(graph)) # get hamilton path

        print('c. Hamilton Path: ', )

        print('\t', end='')

        for x in hamiltonPath:

            if x != 'a':

                print(' ->', end=' ')

            print(x, end='') # print hamilton path if exists

    except:

        print('c. Hamilton Path does not exist.') # else print exception

def printGraphTraversals(graph):

    print('\nd. Graph Traversals (source node = a)')

    print('Breath First Search: visualized using matplotlib')

    BFStree = networkx.bfs\_tree(graph, 'a') # create BFS tree

    networkx.draw(BFStree, with\_labels=True)

    plot.show() # visualize BFS tree

    print('Depth First Search: visualized using matlplotlib')

    DFStree = networkx.dfs\_tree(graph, 'a') # create DFS tree

    networkx.draw(DFStree, with\_labels=True)

    plot.show() # visualize DFS tree

    input('\npress enter key to return...')

# main

# initialize graph

graphOne = networkx.Graph()

# creating nodes a - o

nodeName = 'a'

for x in range(15):

    graphOne.add\_node(nodeName)

    nodeName = chr(ord(nodeName)+1)

# creating edges

graphOne.add\_edge('a','b')

graphOne.add\_edge('a','f')

graphOne.add\_edge('b','g')

graphOne.add\_edge('b','c')

graphOne.add\_edge('b','d')

graphOne.add\_edge('c','d')

graphOne.add\_edge('c','j')

graphOne.add\_edge('c','h')

graphOne.add\_edge('d','e')

graphOne.add\_edge('d','i')

graphOne.add\_edge('e','j')

graphOne.add\_edge('f','g')

graphOne.add\_edge('f','k')

graphOne.add\_edge('f','m')

graphOne.add\_edge('g','h')

graphOne.add\_edge('g','l')

graphOne.add\_edge('h','i')

graphOne.add\_edge('h','m')

graphOne.add\_edge('i','j')

graphOne.add\_edge('i','n')

graphOne.add\_edge('j','o')

graphOne.add\_edge('k','l')

graphOne.add\_edge('l','m')

graphOne.add\_edge('l','n')

graphOne.add\_edge('m','n')

graphOne.add\_edge('n','o')

# menu system

while True:

    os.system('cls')

    printMenu()

    option = input('\nOption: ')

    if option == '1':

        printNodeList(graphOne)

    elif option == '2':

        printEdgeList(graphOne)

    elif option == '3':

        printCountConnectedComponents(graphOne)

    elif option == '4':

        printConnectedComponents(graphOne)

    elif option == '5':

        printIncidenceMatrix(graphOne)

    elif option == '6':

        printNodeDegrees(graphOne)

    elif option == '7':

        printCountEdges(graphOne)

    elif option == '8':

        visualizeGraph(graphOne)

    elif option == '9':

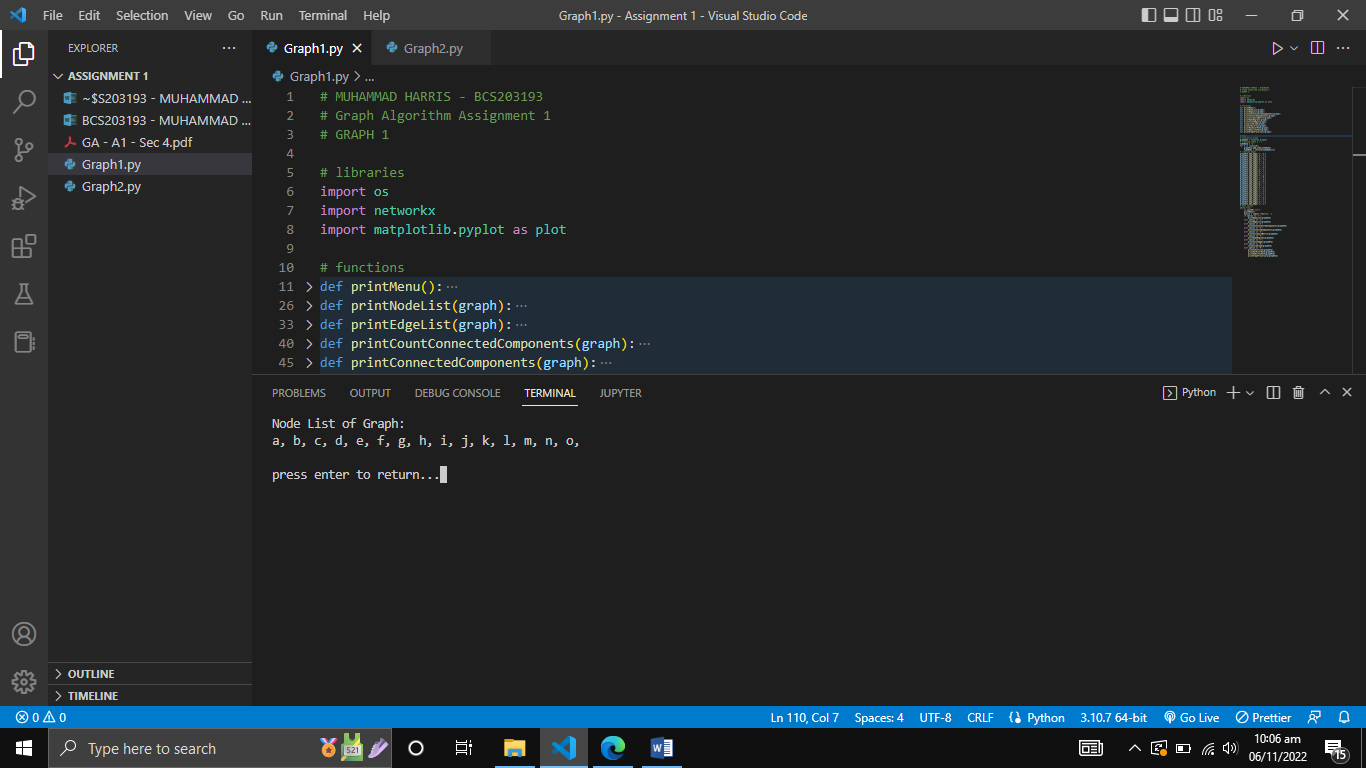
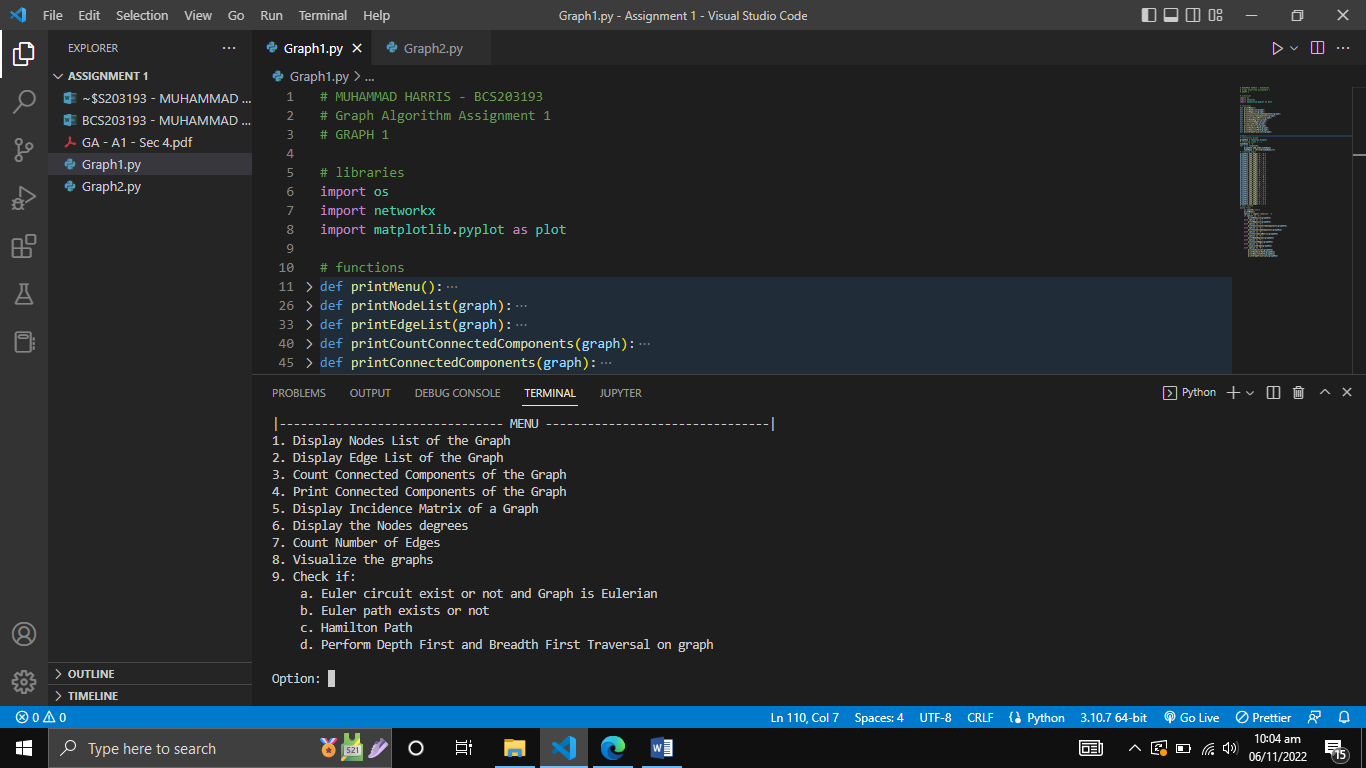
        printIsEulerian(graphOne)

        printHasEulerPath(graphOne)

        printHamiltonPath(graphOne)

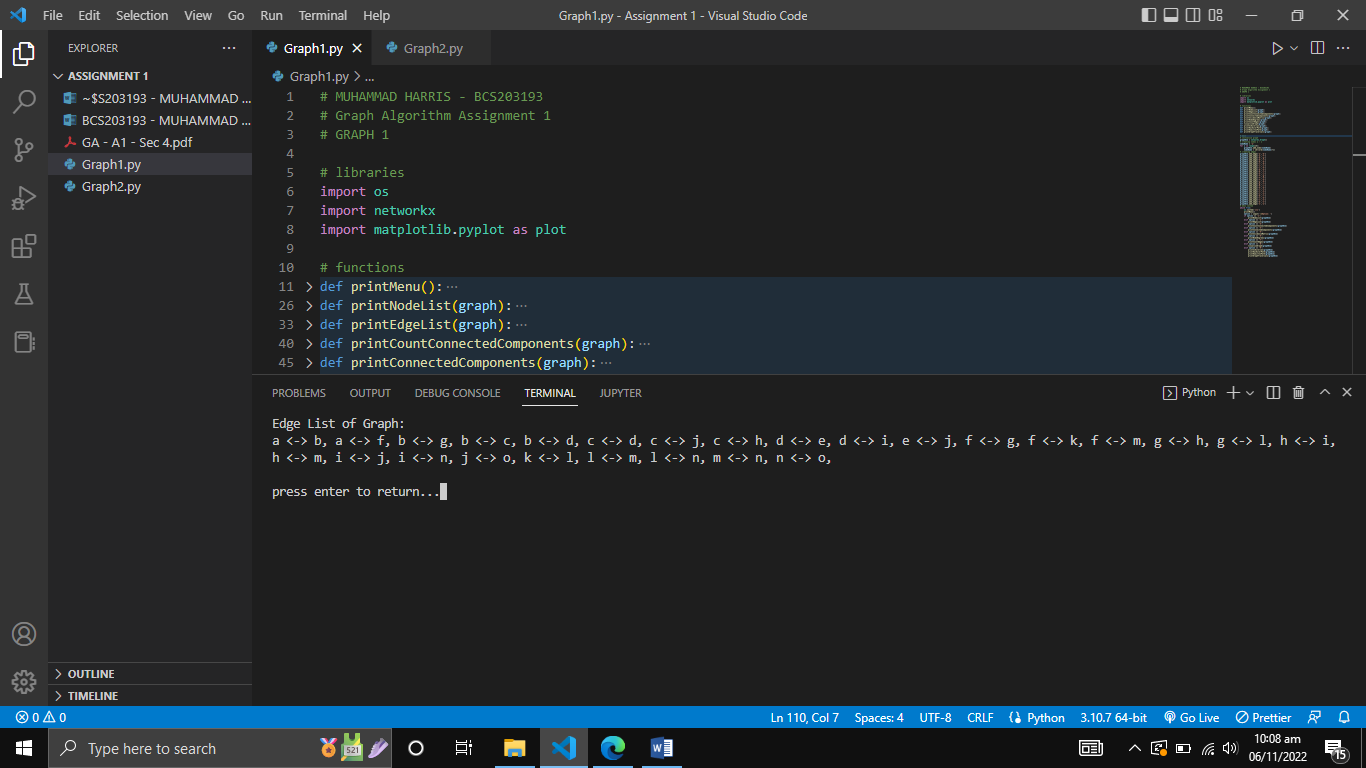
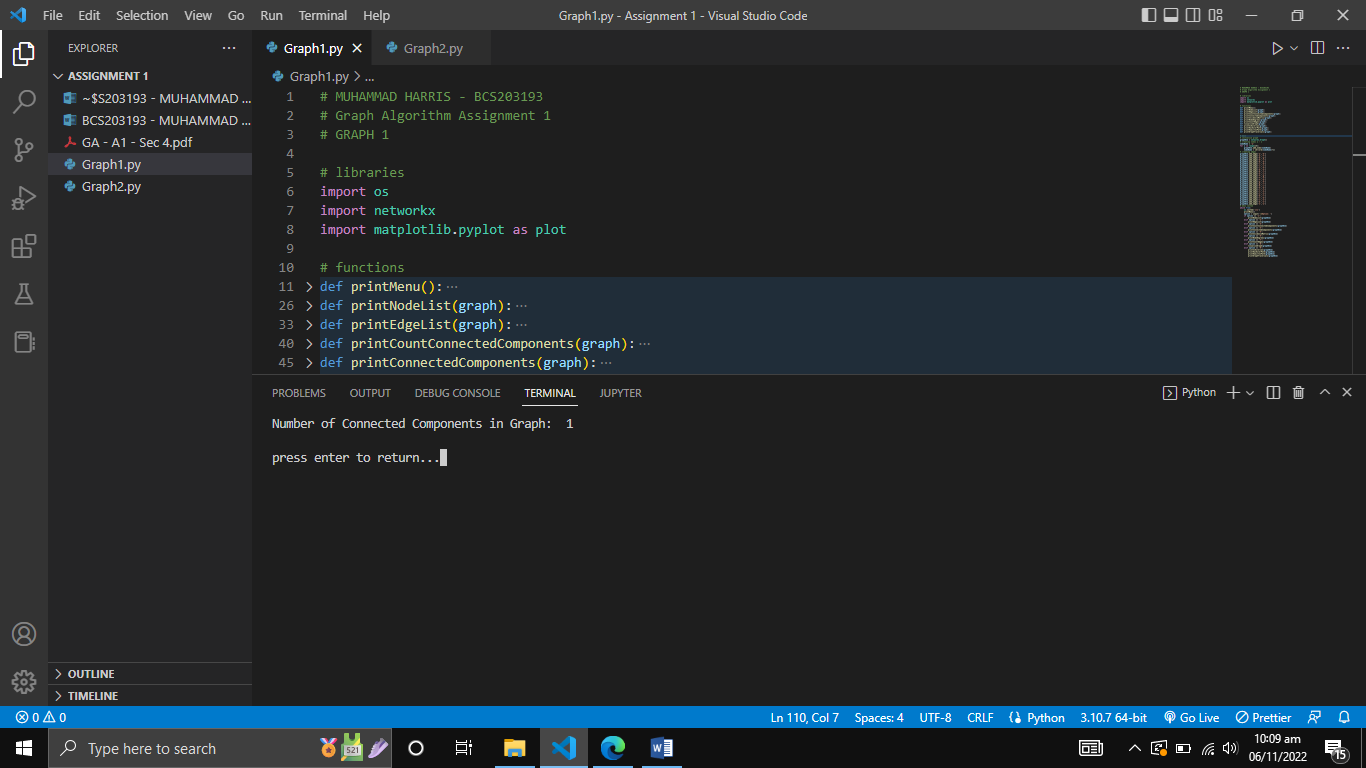
        printGraphTraversals(graphOne)

### Outputs of Graph1.py



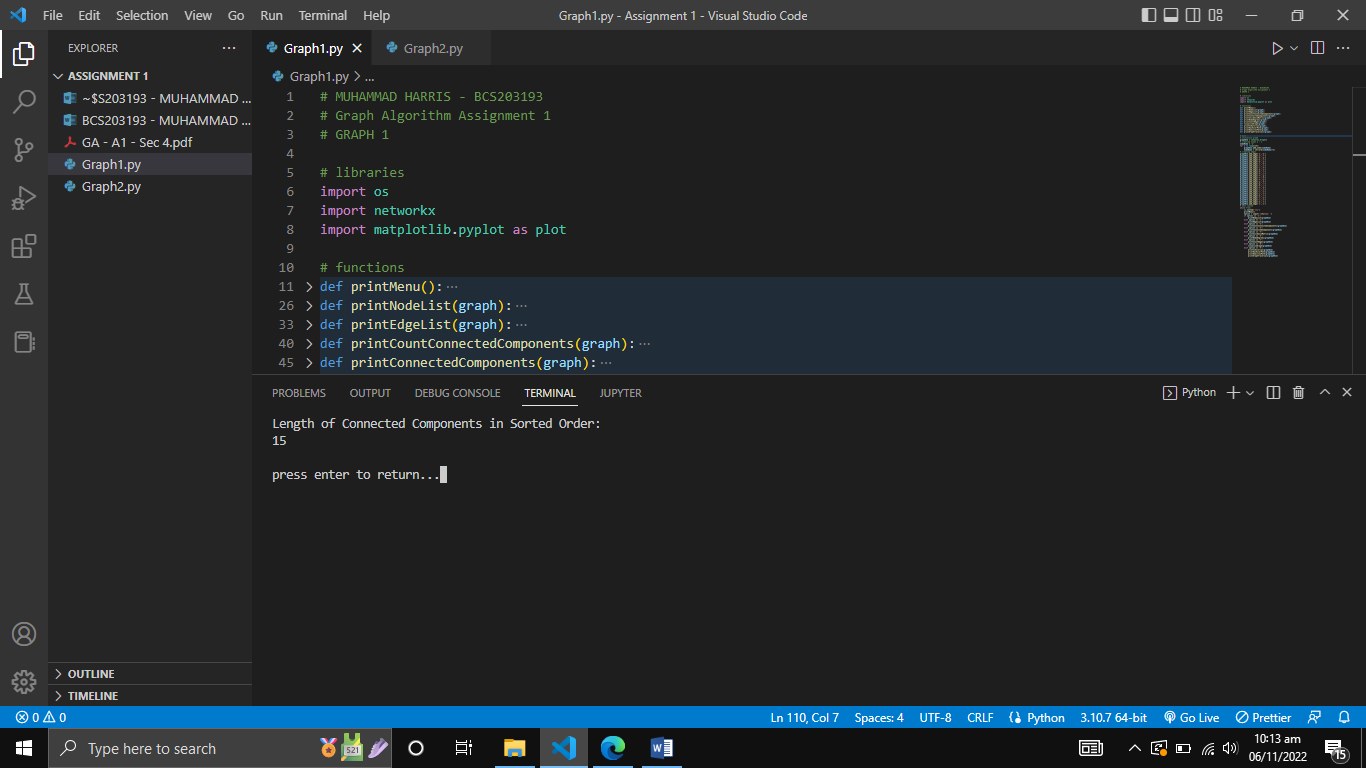
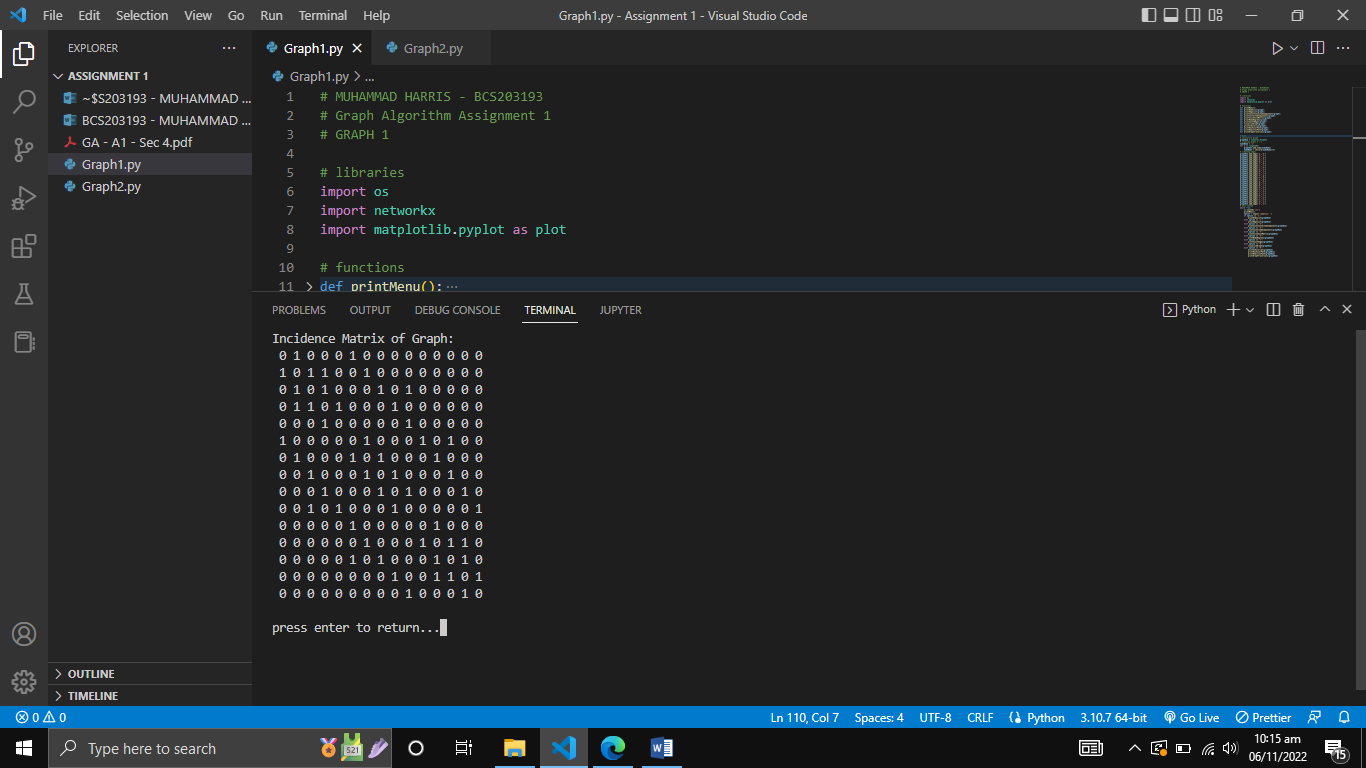
Output 1: main menu

Output 2: node list of graph



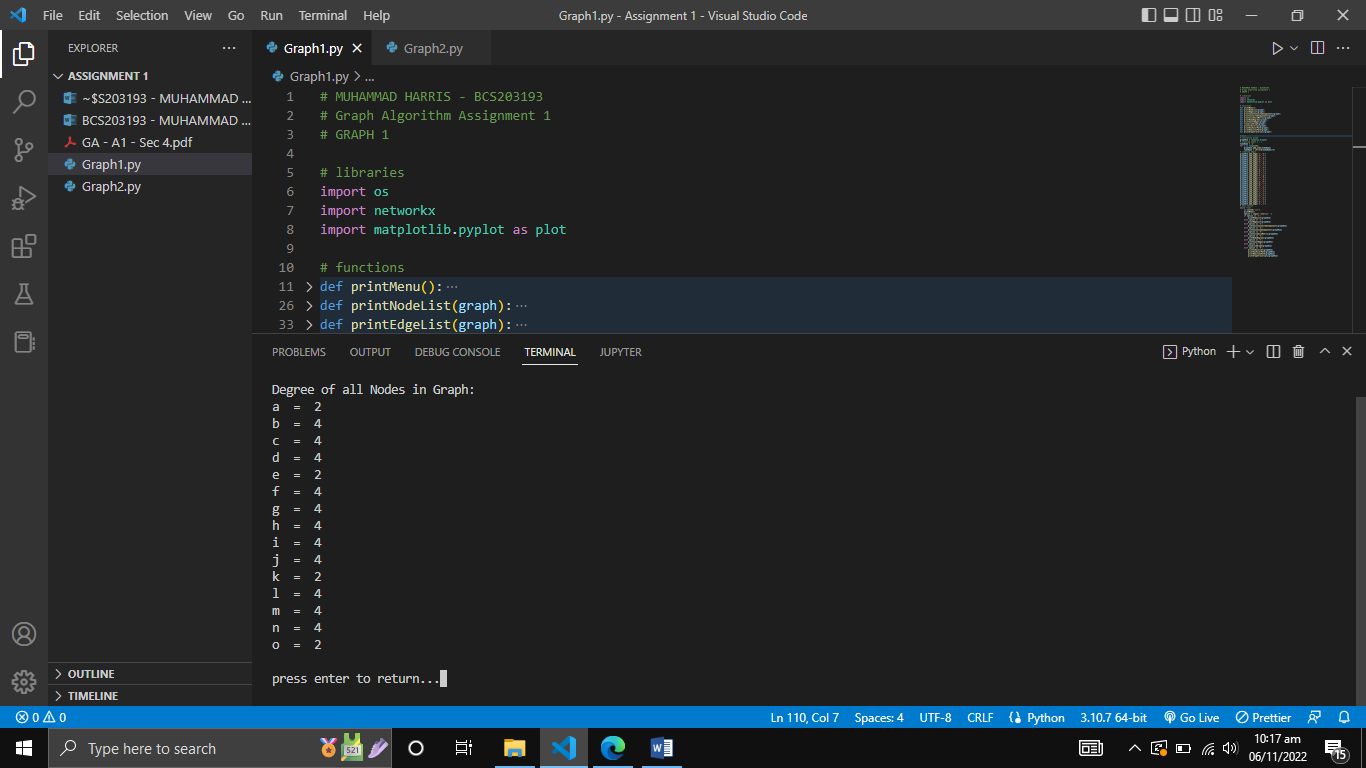
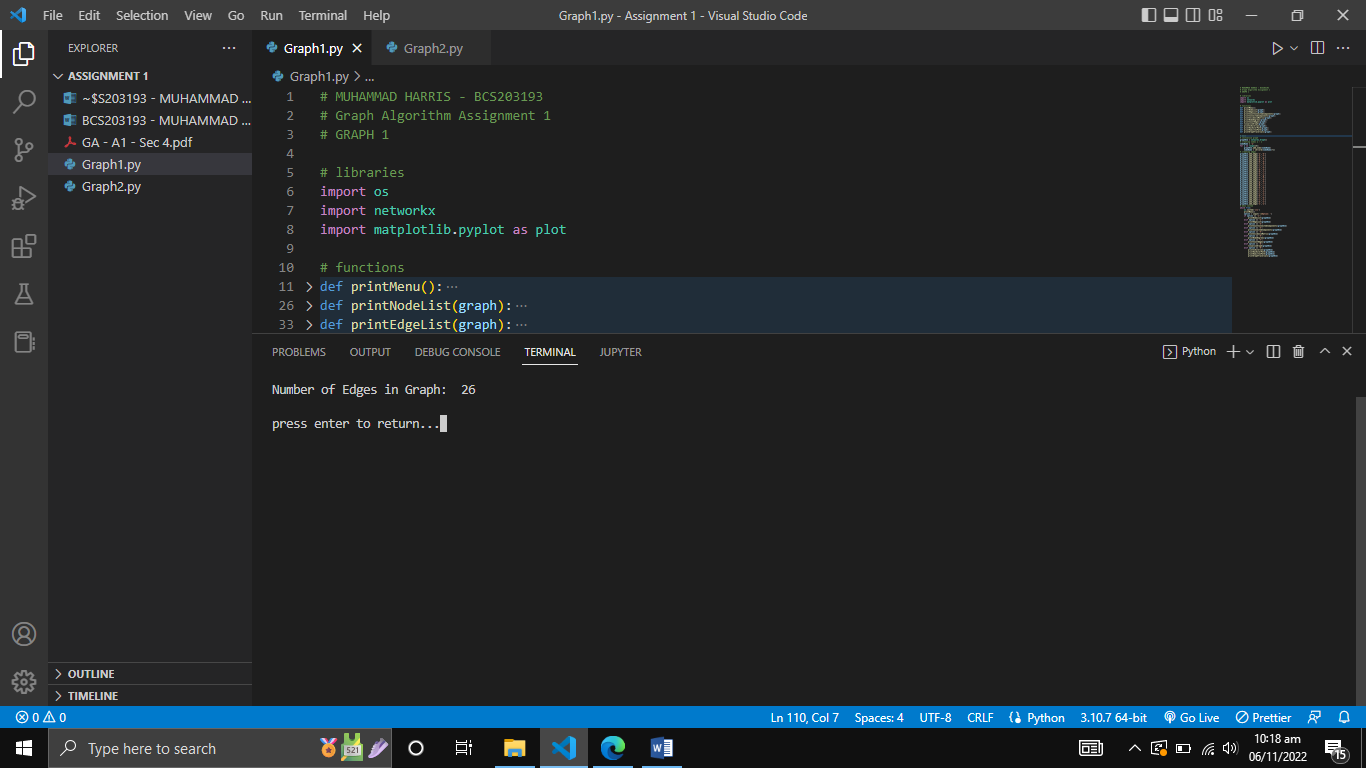
Output 4: number of connected components in graph

Output 3: edge list of graph



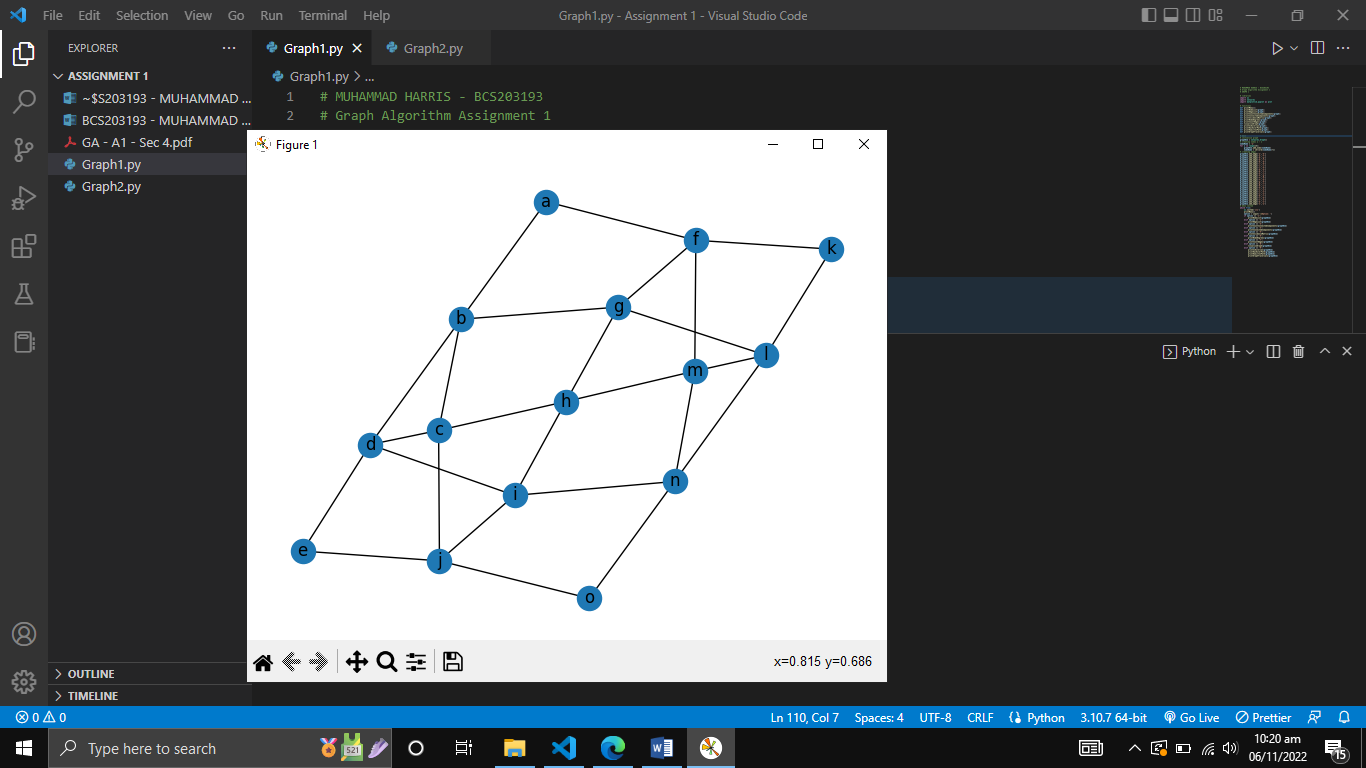
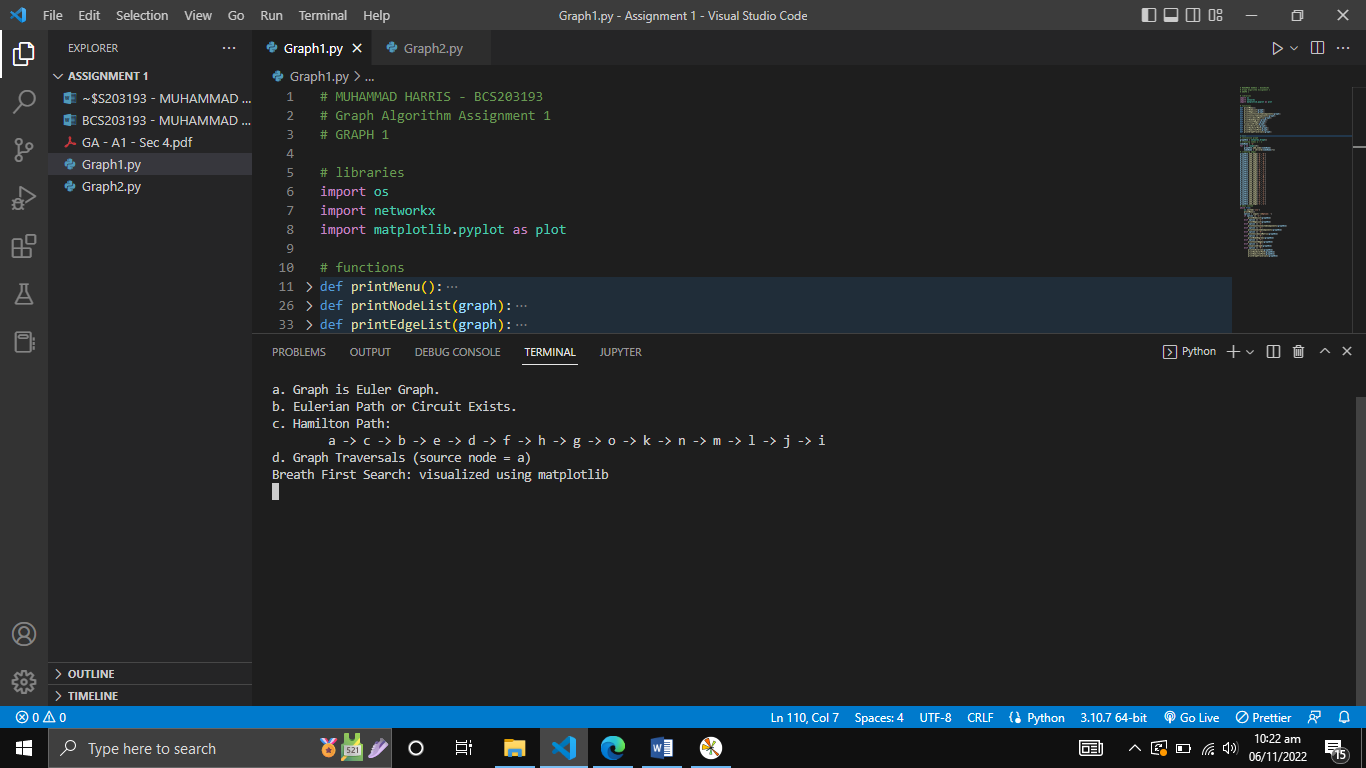
Output 6: incidence matrix of graph

Output 5: connected components of the graph



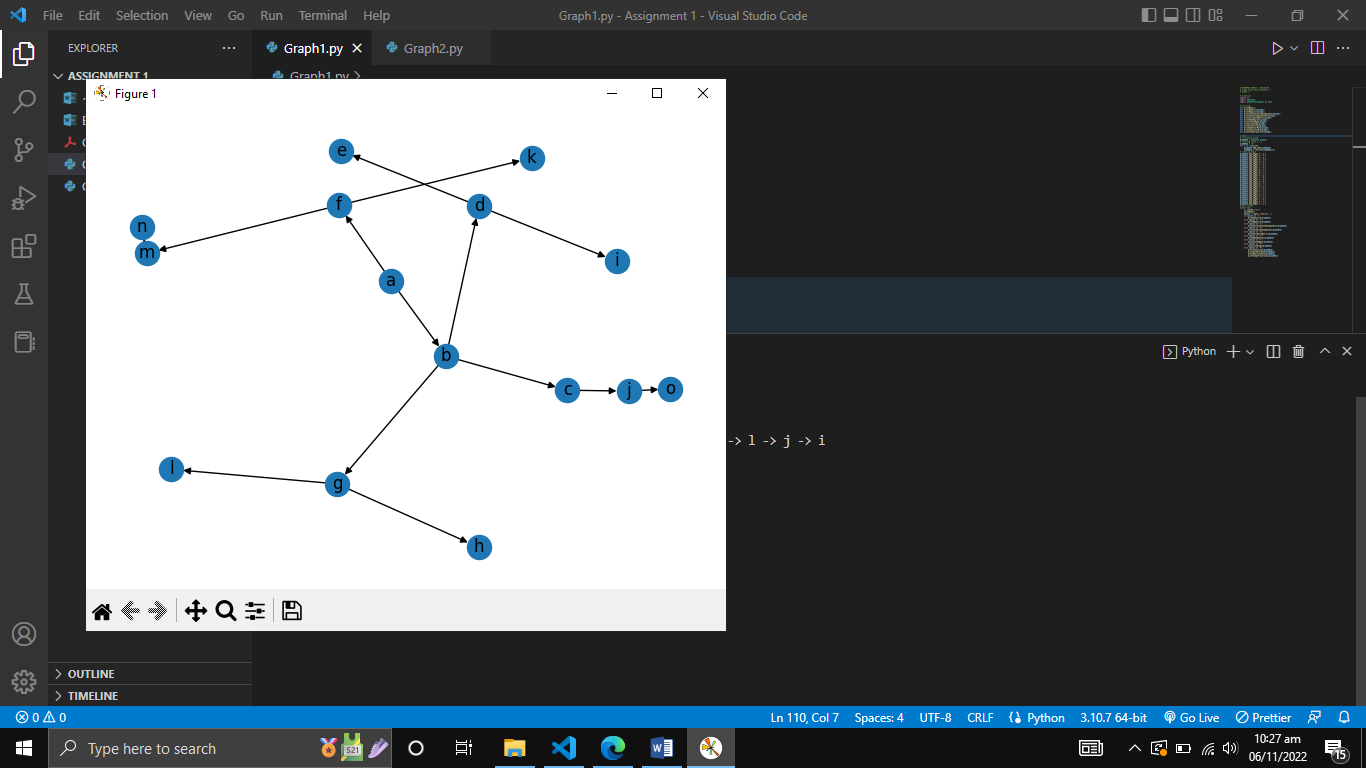
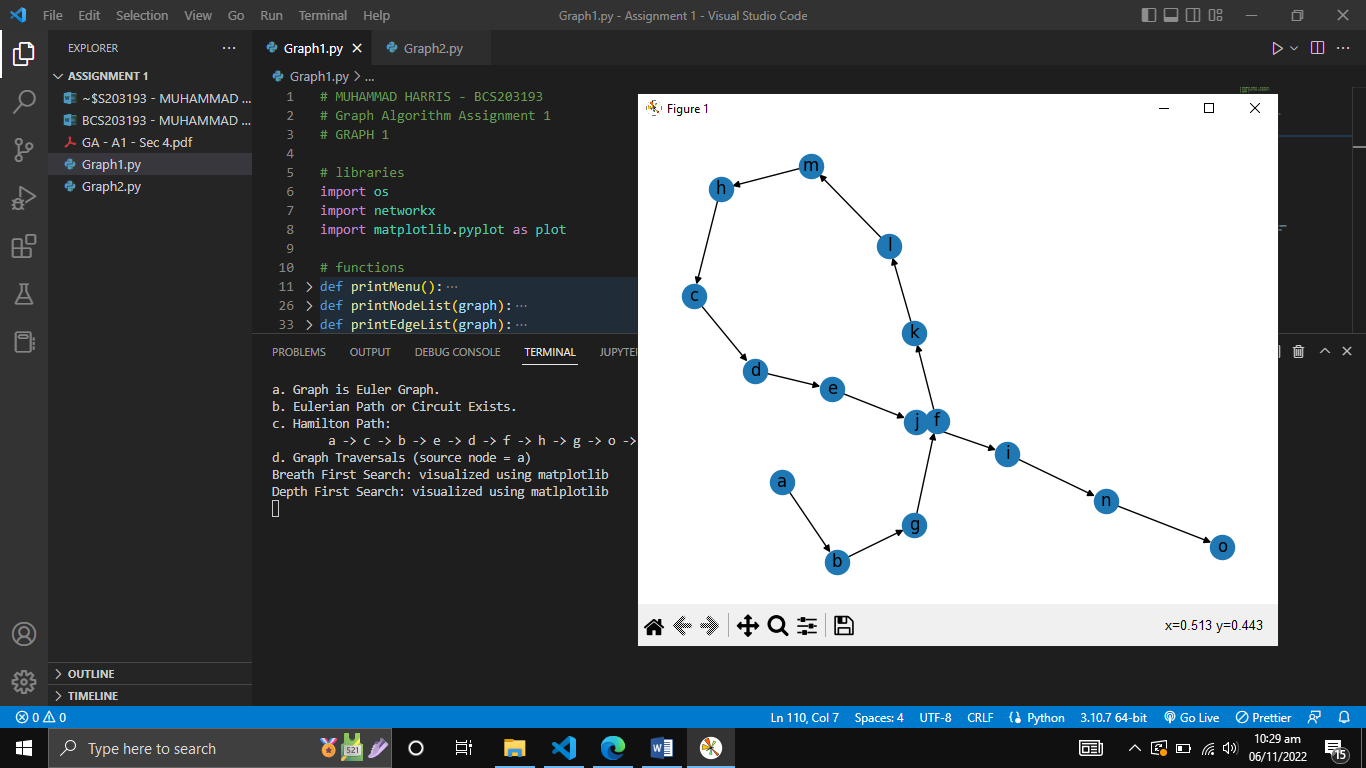
Output 8: number of edges in graph

Output 7: node degrees



Output 10: eulerian & hamilton path

Output 9: visualization of graph



Output 12: DFS traversal

Output 11: BFS traversal

## Graph 2:

### Code of Graph2.py

# MUHAMMAD HARRIS - BCS203193

# Graph Algorithm Assignment 1

# GRAPH 2

# libraries

import os

import networkx

import matplotlib.pyplot as plot

# functions

def printMenu():

    print("""|-------------------------------- MENU --------------------------------|

1. Display Nodes List of the Graph

2. Display Edge List of the Graph

3. Count Connected Components of the Graph

4. Print Connected Components of the Graph

5. Display Incidence Matrix of a Graph

6. Display the Nodes degrees

7. Count Number of Edges

8. Visualize the graphs

9. Check if:

    a. Euler circuit exist or not and Graph is Eulerian

    b. Euler path exists or not

    c. Hamilton Path

    d. Perform Depth First and Breadth First Traversal on graph""")

def printNodeList(graph):

    os.system('cls')

    nodeList = graph.nodes() # get node list of graph

    print('Node List of Graph:')

    for x in nodeList:

        print(x, end=', ') # print each node

    input('\n\npress enter to return...')

def printEdgeList(graph):

    os.system('cls')

    edgeList = graph.edges() # get edge list of graph

    print('Edge List of Graph:')

    for x in edgeList:

        print(x[0], '->', x[1], end=', ') # print each edge

    input('\n\npress enter to return...')

def printCountConnectedComponents(graph):

    os.system('cls')

    count = networkx.number\_strongly\_connected\_components(graph) # get number of connected components

    print('Number of Connected Components in Graph: ', count) # print count

    input('\npress enter to return...')

def printConnectedComponents(graph):

    os.system('cls')

    connectedComponents = [len(c) for c in sorted(networkx.strongly\_connected\_components(graph), key=len, reverse=True)] # get connected components

    print('Length of Connected Components in Sorted Order:')

    for x in connectedComponents:

        print(x) # print length of each connected component

    input('\npress enter to return...')

def printIncidenceMatrix(graph):

    os.system('cls')

    incidenceMatrix = networkx.to\_numpy\_matrix(graph) # get incidence matix

    incidenceMatrixString = str(incidenceMatrix).replace('[','').replace(']','').replace('.','')

    print('Incidence Matrix of Graph: ')

    print('',incidenceMatrixString) # print incidence matrix

    input('\npress enter to return...')

def printNodeDegrees(graph):

    os.system('cls')

    print('in-Degree of all Nodes in Graph:')

    nodeN = 'a'

    for x in range(6):

        print(nodeN, ' = ', graph.in\_degree(nodeN)) # print degree of each node

        nodeN = chr(ord(nodeN)+1)

    print('out-Degree of all Nodes in Graph:')

    nodeN = 'a'

    for x in range(6):

        print(nodeN, ' = ', graph.out\_degree(nodeN)) # print degree of each node

        nodeN = chr(ord(nodeN)+1)

    input('\npress enter to return...')

def printCountEdges(graph):

    os.system('cls')

    print('Number of Edges in Graph (parallel edges ignored): ', networkx.number\_of\_edges(graph)) # print number of edges in graph

    input('\npress enter to return...')

def visualizeGraph(graph):

    os.system('cls')

    networkx.draw(graph, with\_labels = True) # draw graph with labels

    plot.show()

    input('press enter to return...')

def printIsEulerian(graph):

    os.system('cls')

    if networkx.is\_eulerian(graph): # check if graph is euler or not

        print('a. Graph is Euler Graph.')

    else:

        print('a. Graph is not Euler.')

def printHasEulerPath(graph):

    if networkx.has\_eulerian\_path(graph): # check if graph has euler path or not

        print('b. Eulerian Path or Circuit Exists.')

    else:

        print('b. Eulerian Path or Circuit does not Exist.')

def printHamiltonPath(graph):

    try:

        hamiltonPath = networkx.algorithms.tournament.hamiltonian\_path(graph) # get hamilton path

        print('c. Hamilton Path: ', )

        print('\t', end='')

        for x in hamiltonPath:

            if x != 'a':

                print(' ->', end=' ')

            print(x, end='') # print hamilton path if exists

    except:

        print('c. Hamilton Path does not exist.') # else print exception

def printGraphTraversals(graph):

    print('\nd. Graph Traversals (source node = a)')

    print('Breath First Search: visualized using matplotlib')

    BFStree = networkx.bfs\_tree(graph, 'a') # create BFS tree

    networkx.draw(BFStree, with\_labels=True)

    plot.show() # visualize BFS tree

    print('Depth First Search: visualized using matlplotlib')

    DFStree = networkx.dfs\_tree(graph, 'a') # create DFS tree

    networkx.draw(DFStree, with\_labels=True)

    plot.show() # visualize DFS tree

    input('\npress enter key to return...')

# main

# initialize graph

graphTwo = networkx.DiGraph()

# creating nodes a - f

nodeName = 'a'

for x in range(6):

    graphTwo.add\_node(nodeName)

    nodeName = chr(ord(nodeName)+1)

# creating edges

graphTwo.add\_edge('a','b')

graphTwo.add\_edge('a','f')

graphTwo.add\_edge('b','c')

graphTwo.add\_edge('b','d')

graphTwo.add\_edge('b','f')

graphTwo.add\_edge('c','b')

graphTwo.add\_edge('c','b')

graphTwo.add\_edge('c','e')

graphTwo.add\_edge('d','c')

graphTwo.add\_edge('d','e')

graphTwo.add\_edge('e','a')

graphTwo.add\_edge('e','b')

graphTwo.add\_edge('e','f')

graphTwo.add\_edge('f','a')

graphTwo.add\_edge('f','d')

graphTwo.add\_edge('f','e')

# menu system

while True:

    os.system('cls')

    printMenu()

    option = input('\nOption: ')

    if option == '1':

        printNodeList(graphTwo)

    elif option == '2':

        printEdgeList(graphTwo)

    elif option == '3':

        printCountConnectedComponents(graphTwo)

    elif option == '4':

        printConnectedComponents(graphTwo)

    elif option == '5':

        printIncidenceMatrix(graphTwo)

    elif option == '6':

        printNodeDegrees(graphTwo)

    elif option == '7':

        printCountEdges(graphTwo)

    elif option == '8':

        visualizeGraph(graphTwo)

    elif option == '9':

        printIsEulerian(graphTwo)

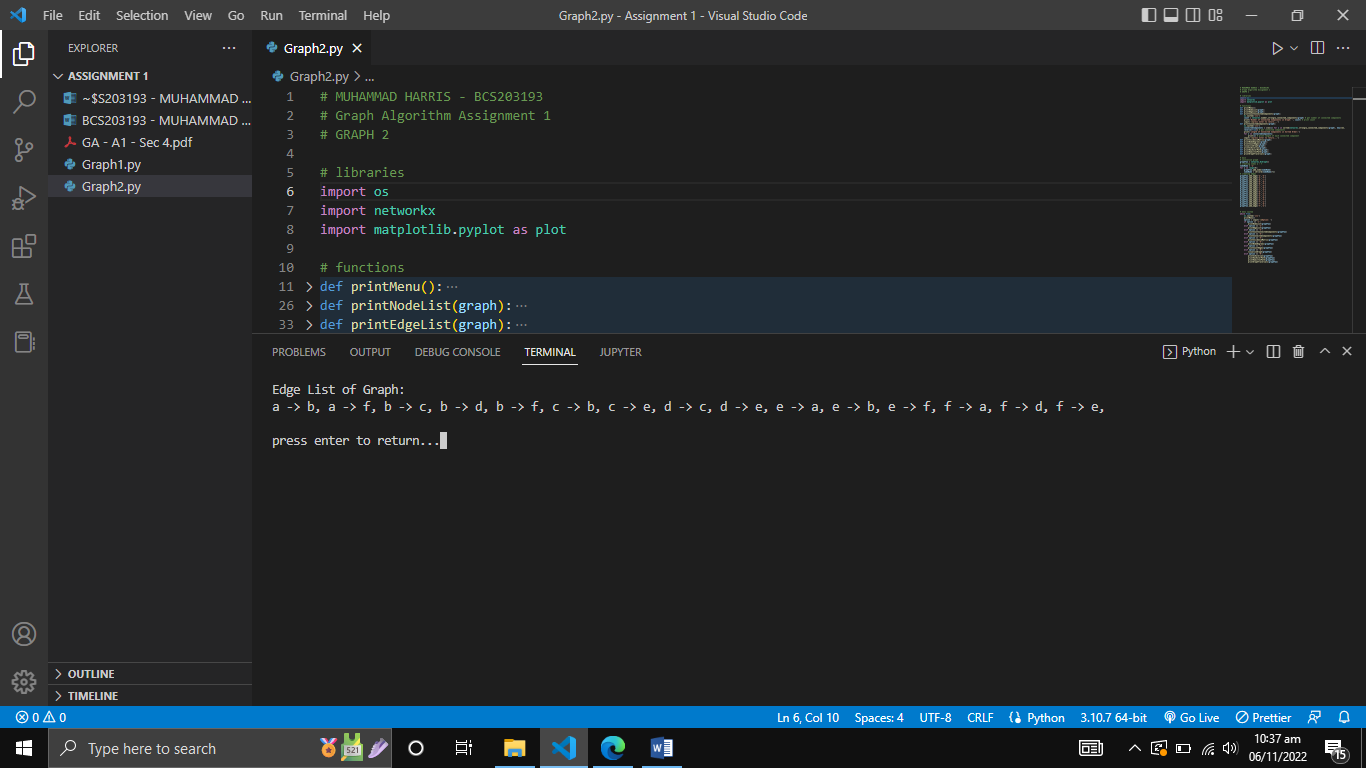
        printHasEulerPath(graphTwo)

        printHamiltonPath(graphTwo)

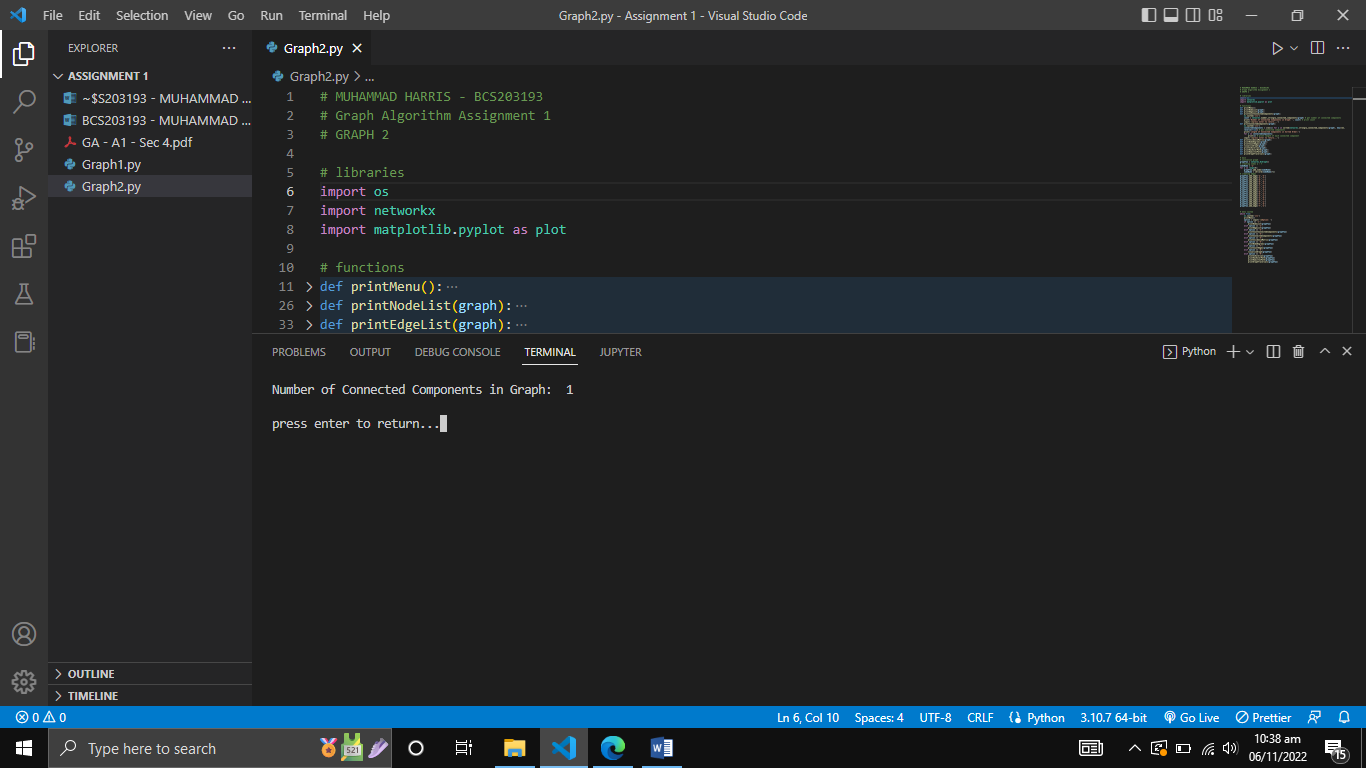
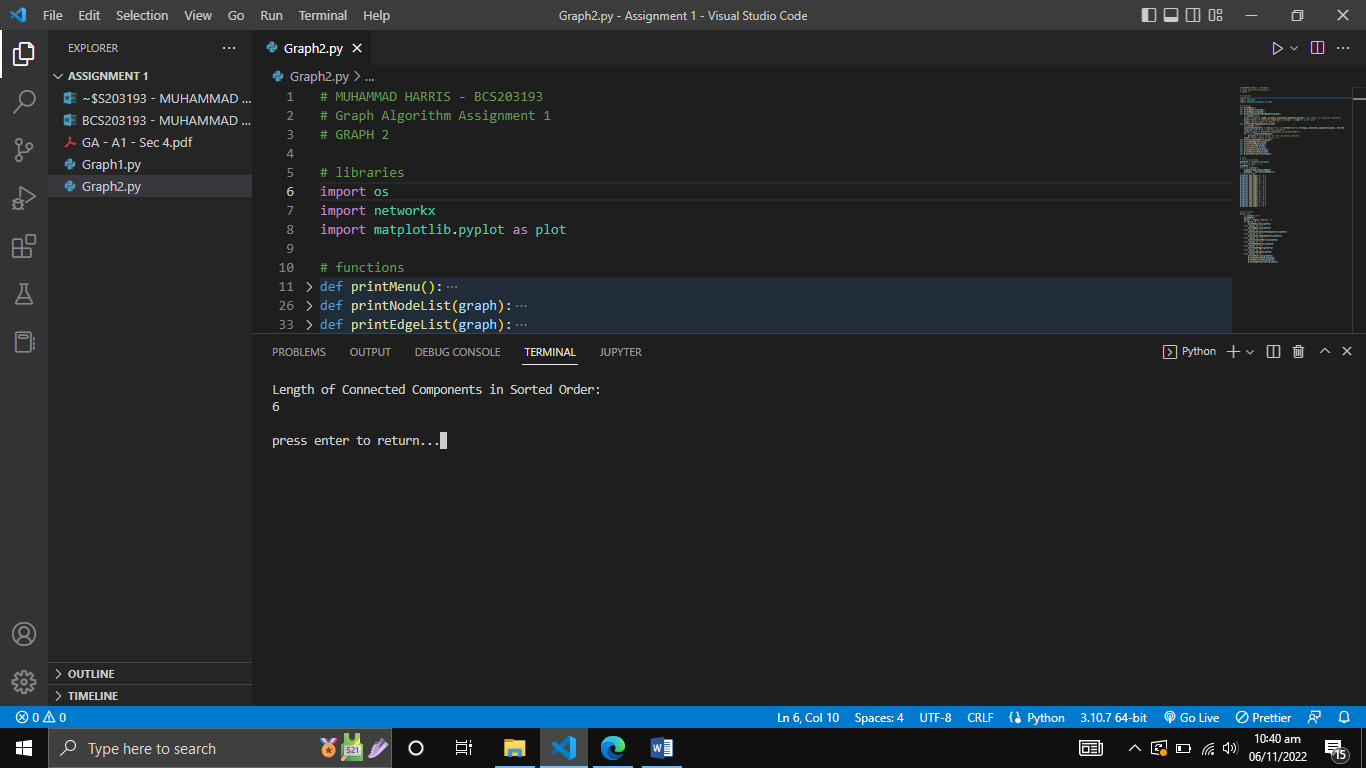
        printGraphTraversals(graphTwo)

### Outputs of Graph2.py

Output 1: node list of graph

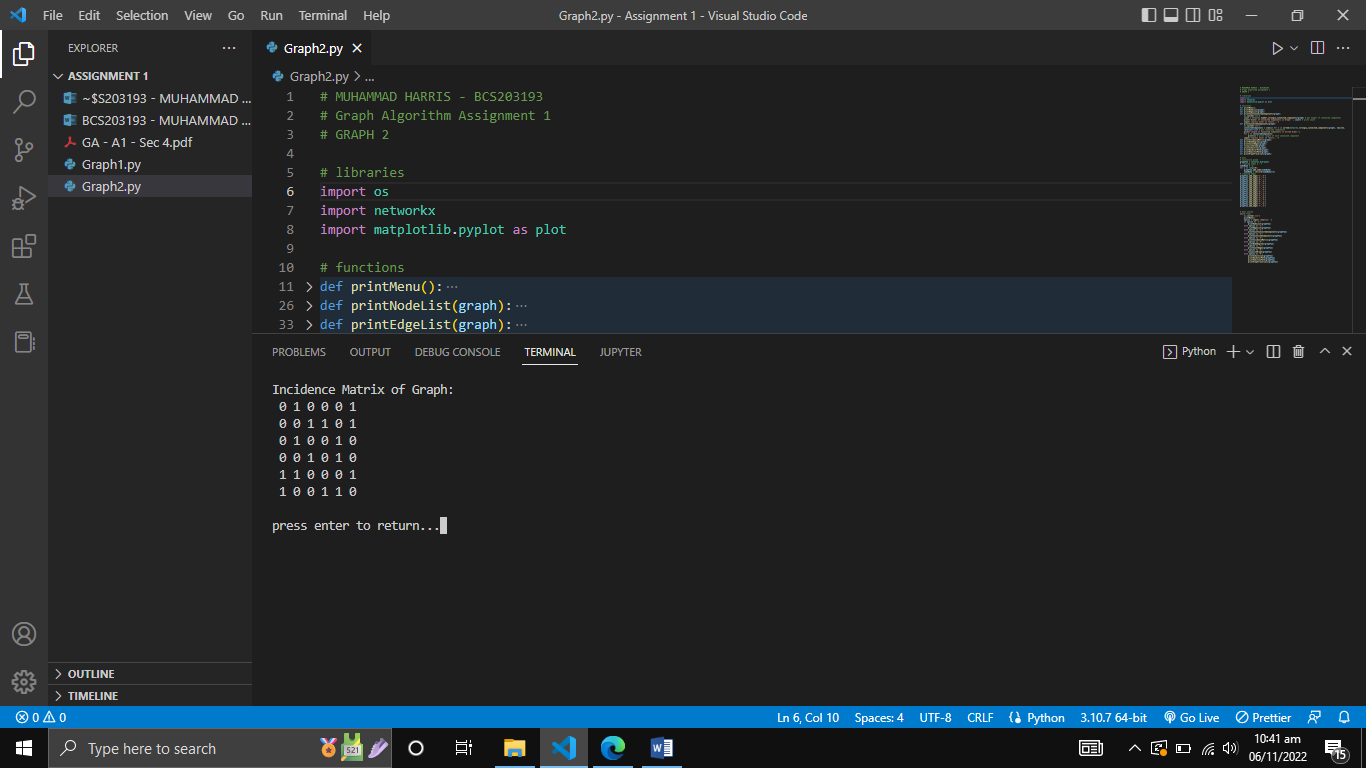
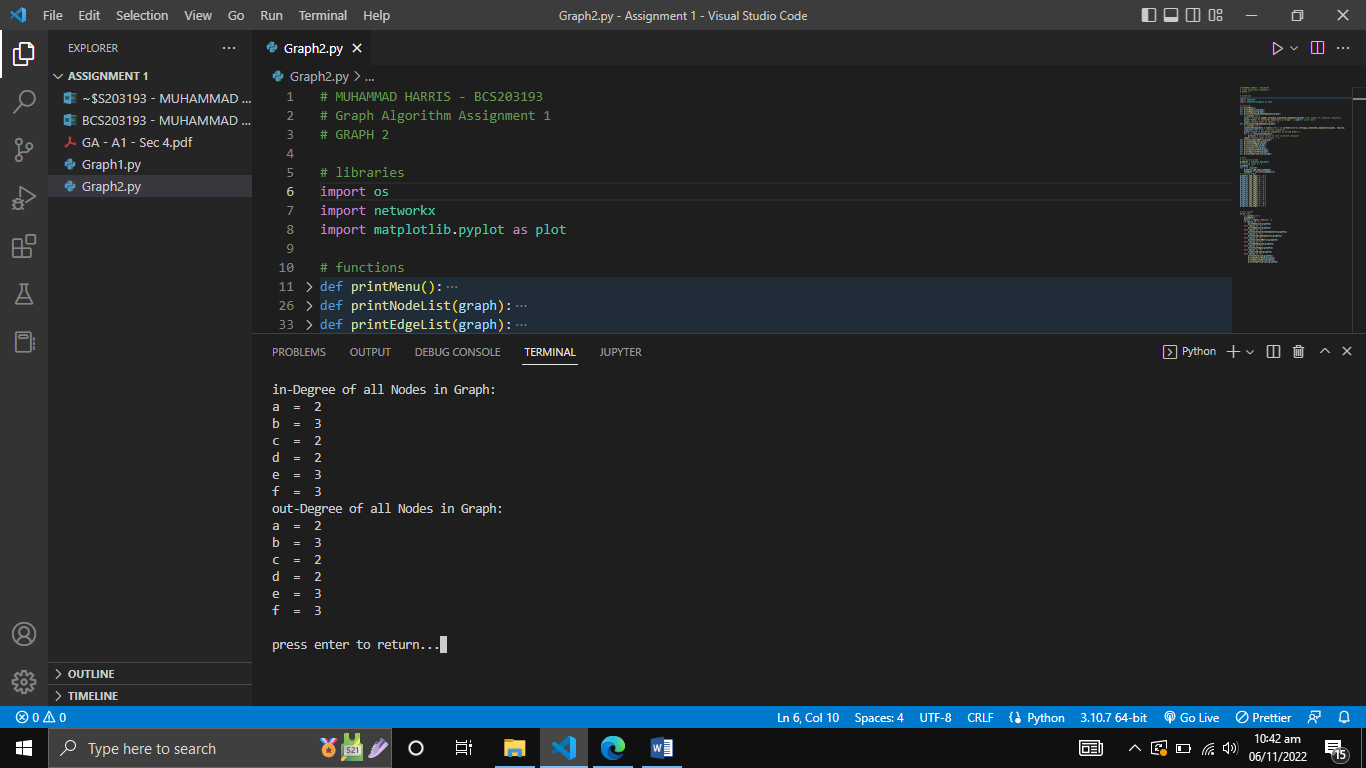


Output 2: node list of graph



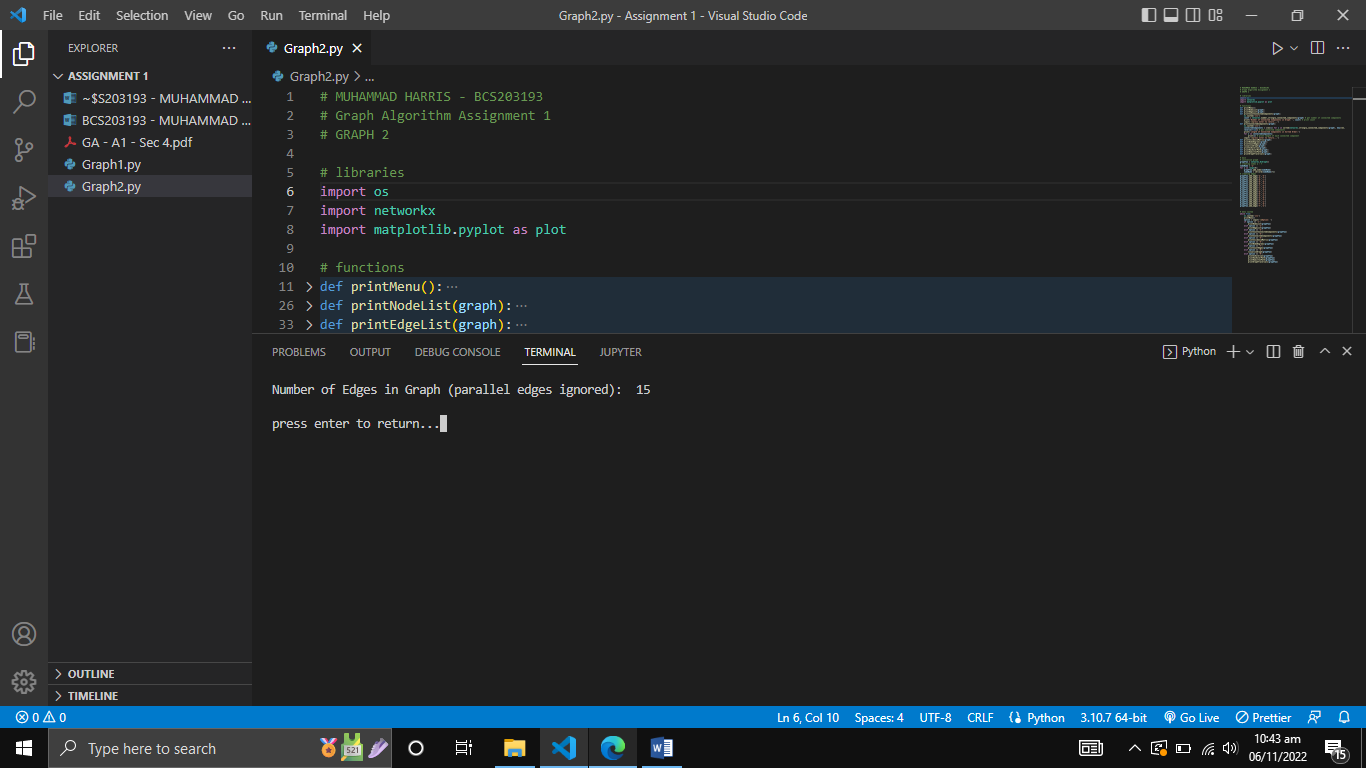
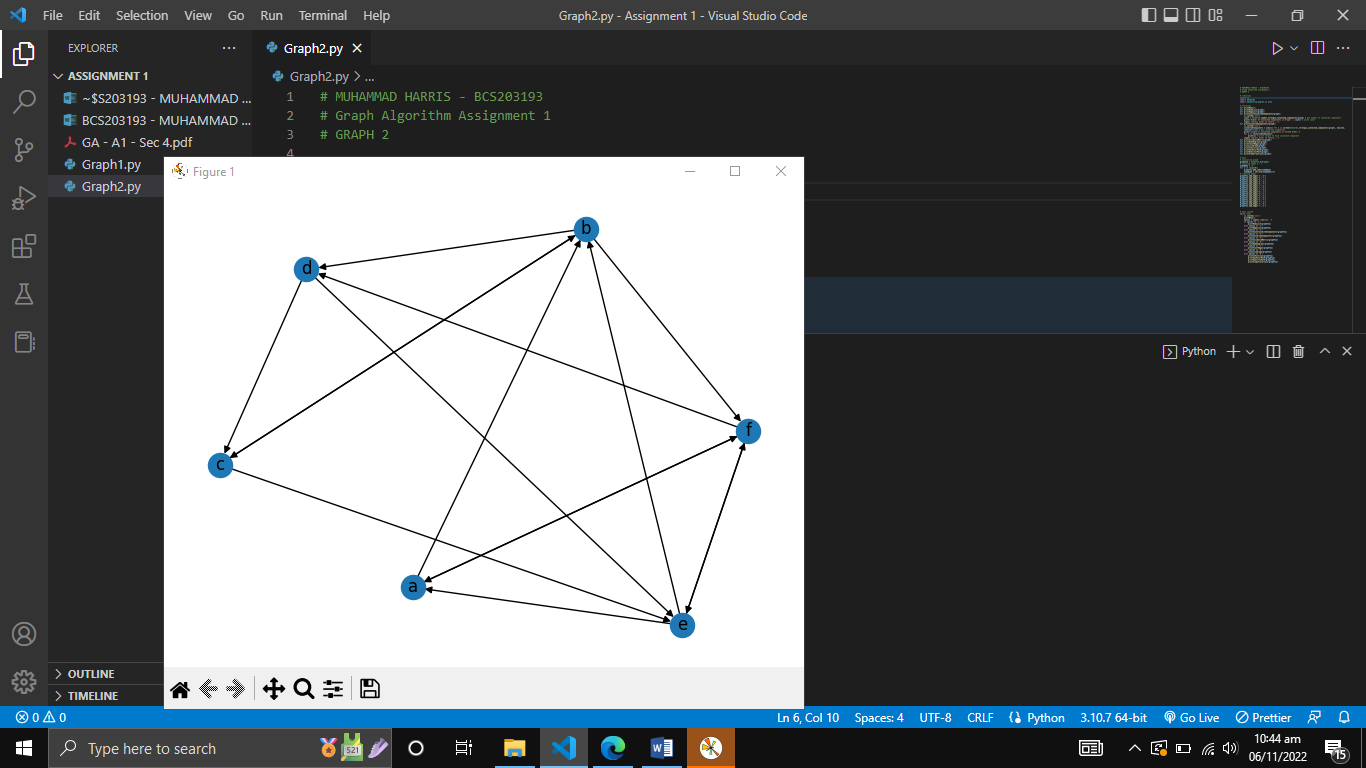
Output 4: connected components of graph

Output 3: number of connected components



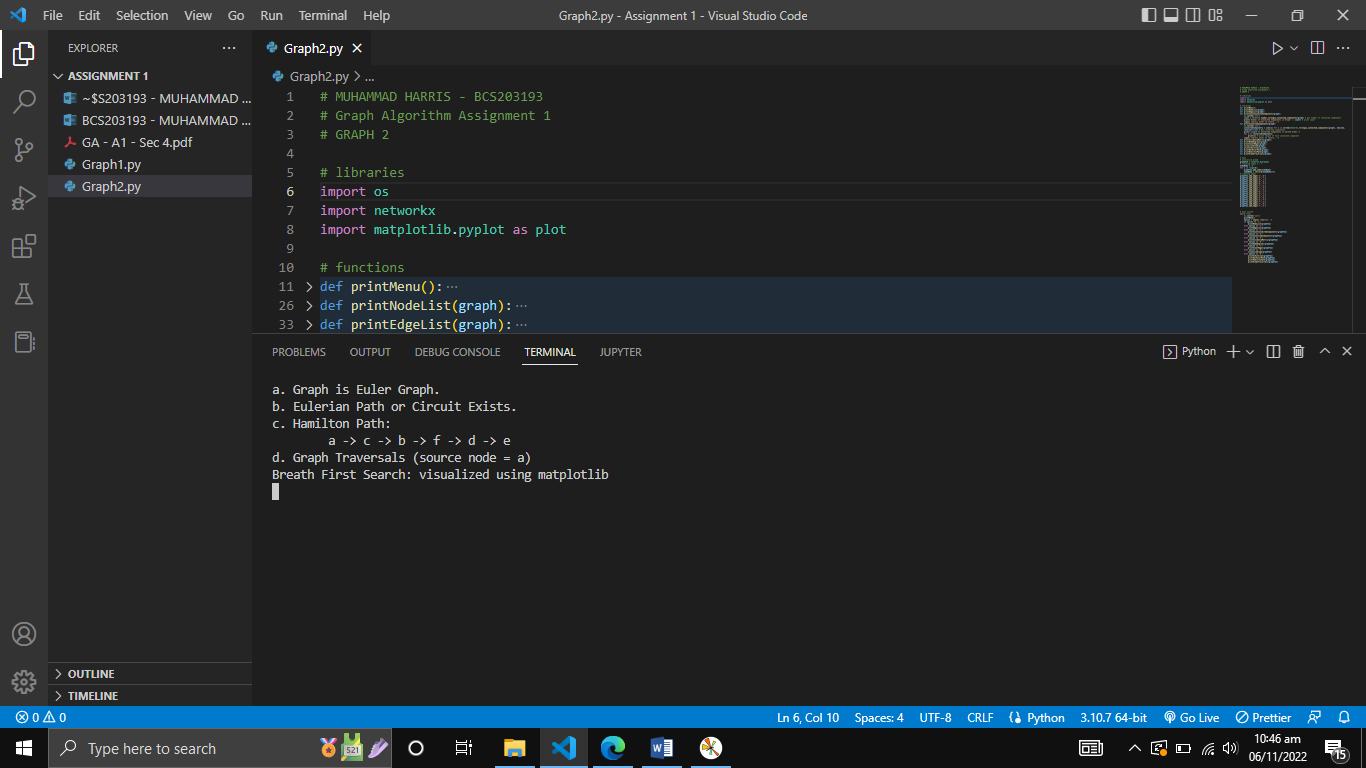
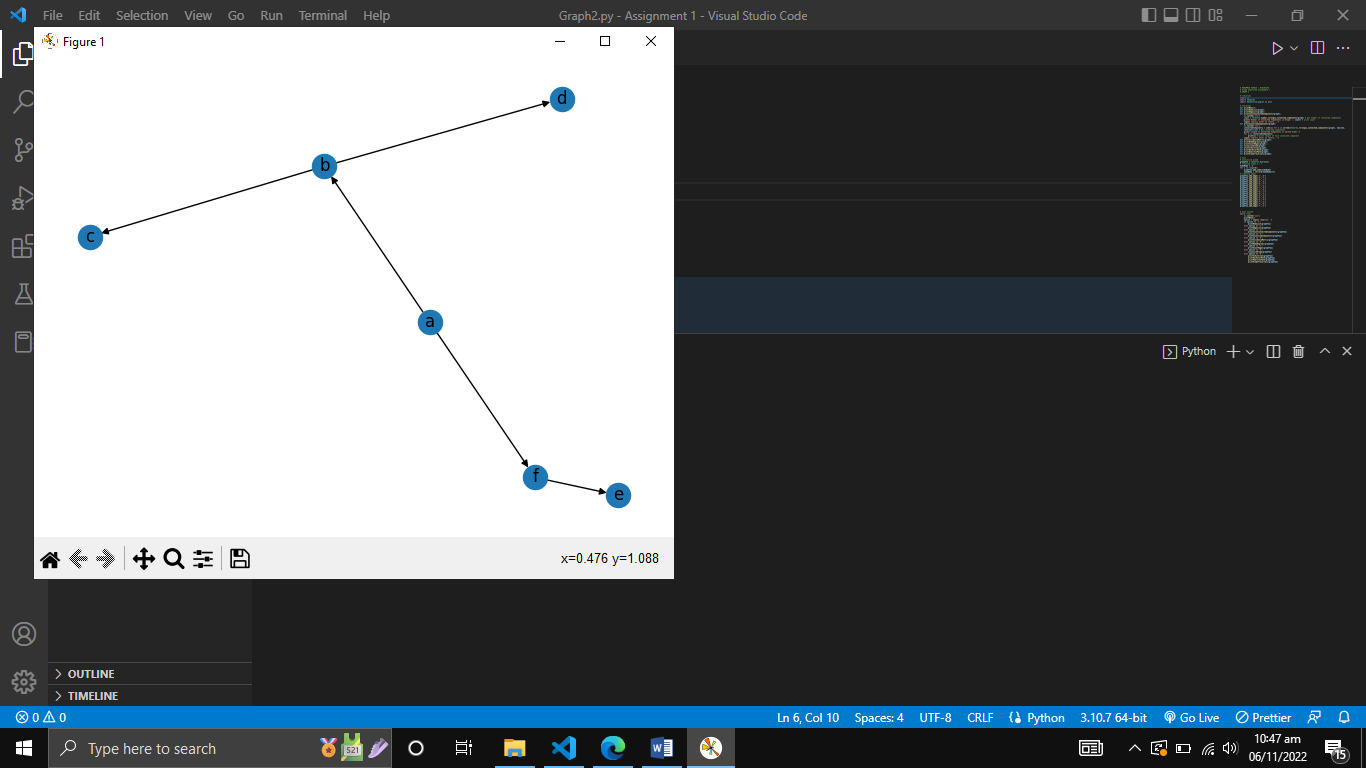
Output 6: in-degree & out-degree of nodes

Output 5: incidence matrix of graph



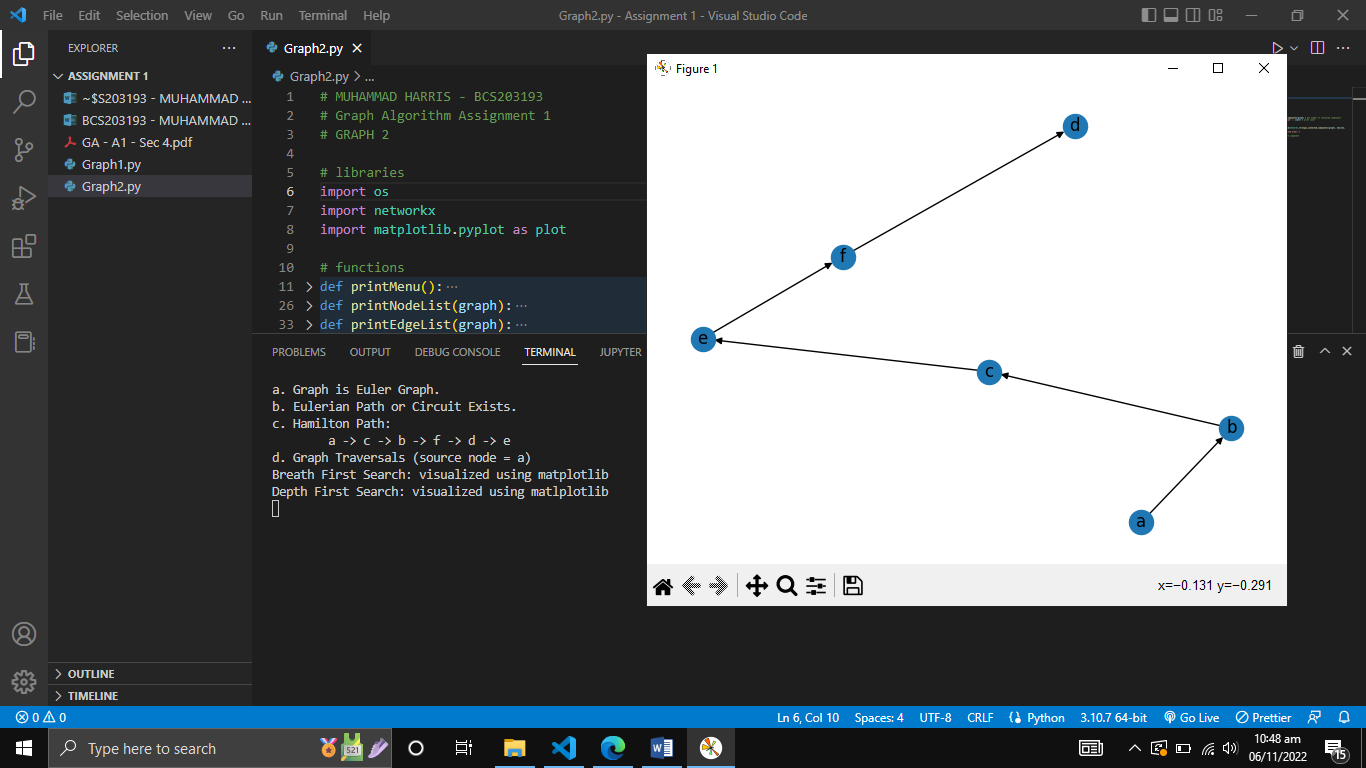
Output 8: graph visualization

Output 7: number of edges in graph



Output 10: BFS traversal

Output 9: eulerian & hamilton path



Output 11: DFS traversal