ULB Digital and IT essentials VUB

COURSE: FOUNDATIONS

TASK 2: Social Network

SUBMITTED BY:

DEEPA MARY VARGHESE

18/04/2025

INTRODUCTION:

The goal of this project is to have you write a collection of Python functions to manipulate a network. I had to build a Social media network step-by-step using functions.

METHODOLOGY:

Step 1: I had to understand what does each function do.

Step 2. I started writing the programme step-by-step using the functions mentioned in the assignment

Step 3: Formulated each step in the given function.

Step 4: Run the programme.

Step 5: Debugging the programme.

THE NETWORK:

1. **A function add_friend(network, person1, person2) that takes a dictionary network representing a social network and two strings person1 and person2 as parameters, and modifies the network so that the two persons become friends. If one of the two persons, or both, do not already belong to the network, the function adds her/them. If the two persons are already friends, the function does not do anything. In all cases, the function does not return anything.**
   First I, defined the function and wrote the network in the form of a dictionary where: Keys are names of people (strings) and Values are lists of friends (strings) for each person. I divided the code into 3 steps. First, ensuring both the persons exist in the network. I did that by adding two people in the network. Then I added mutual friendship between person1 and person2.

Python code:

```
network = {'Alice': ['Bob', 'Eva'], 'Eva': ['Alice'], 'Bob': ['Alice', 'Charles'], 'Charles': ['Bob']}
def add_friend(network, person1, person2):
    # Step 1: Ensure both persons exist
    if person1 not in network:
        network[person1] = []
    if person2 not in network:
        network[person2] = []

    # Step 2: Do nothing if they're already friends
    if person2 in network[person1]:
        return

    # Step 3: Add mutual friendship
    network[person1].append(person2)
    network[person2].append(person1)
```

2. **A boolean function is_friend(network, person1, person2) with the same parameters that returns True if the two persons are friends in the network, and False otherwise.**

Here, I Checked if both people exist in the network. I used if and return functions here to check if both people exist in the network. If either person1 or person2 is not in network, they can't be friends → return False. If they are friends, return True and if both are not friends, return False. I also tried to eliminate possibilility of similarity,ie: person1 and person2 should not be same.

Python code:

```
def is_friend(network, person1, person2):
    if person1 not in network or person2 not in network:
        return False
    if person1 == person2:
        return False
    if person1 in network[person2] or person2 in network[person1]:
        return True
    else:
        return False
```

3. **A function common_friend(network, person1, person2) that returns the name of a common friend of person1 and person2, or None if no such person exists. Note that the common friend must be a third person, so that the function may return None even if the two input persons are friends.**

   Here, also I used the if, for and return function. Here I also introduced a common friend. I had to ensure that the common friend is in the network and also friend of minimum two people.

Python code:

```
def common_friend(network, person1, person2):
    # Check if either person is missing from the network
    if person1 not in network or person2 not in network:
        return None

    # Iterate through each friend of person1
    for friend in network[person1]:
        # Check if this friend is also a friend of person2 (and not person1 or person2
themselves)
        if friend in network[person2] and friend != person1 and friend != person2:
            return friend  # Return the first common friend found

    # If no common friend found, return None
    return None
```

4. **A function distance(network, person1, person2) that computes the distance between the two input persons in the network. Two friends are at distance one, two persons with a common friend are at distance at most two, and so on. The function should return -1 whenever the two persons are not connected in the network.**

This part of the code was the most challenging for me. I needed to find the shortest path between person1 and person2 in the network. The assumptions I made were that each person is a node, each friendship is an edge between nodes. So I chose an algorithm. For this step I had to consult AI. My steps:

1. Start at person1 (distance = 0).
2. Explore all direct friends (distance = 1).
3. Then explore their friends (distance = 2), and so on.
4. Stop when we reach person2 or exhaust all possibilities.
5. Use a queue to track people to explore.
6. Use a visited set to avoid cycles.
7. Track distances in a dictionary ({person: distance}).

Python code:

```
def distance(network, person1, person2):
    if person1 not in network or person2 not in network:
        return -1
    if person1 == person2:
        return 0

    queue = [person1]          # People to explore
    visited = {person1}        # Track visited people
    distances = {person1: 0}    # Track distances

    while queue:
        current = queue.pop(0)   # Get the next person
        for friend in network[current]:
            if friend == person2:
                return distances[current] + 1
            if friend not in visited:
                visited.add(friend)
                distances[friend] = distances[current] + 1
                queue.append(friend)
    return -1
```

5. A function diameter(network) that returns the maximum distance between any two persons in the network, or -1 whenever the network is not connected.

Here I was expected to find the shortest path connecting two people and also calculate the diameter ie; the maximum distance between any two persons

Python code:

```
    def diameter(network):
    max_distance = -1
    for person1 in network:
        for person2 in network:
            if person1 != person2:
                dist = distance(network, person1, person2)
                if dist > max_distance:
                    max_distance = dist
```

```
    return max_distance
```

CHALLENGES:

1. Detailed understanding the process of each function
2. Indentation
3. This assignment easy for me to code when I understood the process.

APPENDIX:

THE CODE:

```
network = {'Alice': ['Bob', 'Eva'],'Eva': ['Alice'],'Bob': ['Alice', 'Charles'],'Charles': ['Bob']}
def add_friend(network, person1, person2):

    if person1 not in network:
        network[person1] = []
    if person2 not in network:
        network[person2] = []



    if person2 in network[person1]:
        return



    network[person1].append(person2)
    network[person2].append(person1)

def is_friend(network, person1, person2):
    if person1 not in network or person2 not in network:
        return False
    if person1 == person2:
        return False
```

```
    if person1 in network[person2] or person2 in network[person1]:
        return True
    else:
        return False
def common_friend(network, person1, person2):

    if person1 not in network or person2 not in network:
        return None
    for friend in network[person1]:
         if friend in network[person2] and friend != person1 and friend != person2:
           return friend

    return None



def distance(network, person1, person2):
    if person1 not in network or person2 not in network:
        return -1
    if person1 == person2:
        return 0

    queue = [person1]
    visited = {person1}
    distances = {person1: 0}
    while queue:
        current = queue.pop(0)
        for friend in network[current]:
            if friend == person2:
                return distances[current] + 1
            if friend not in visited:
                visited.add(friend)
```

```
                distances[friend] = distances[current] + 1
                queue.append(friend)
    return -1
def diameter(network):
    max_distance = -1
    for person1 in network:
        for person2 in network:
            if person1 != person2:
                dist = distance(network, person1, person2)
                if dist > max_distance:
                    max_distance = dist
    return max_distance


add_friend(network, 'Deepa', 'Eva')
print(f"Is Alice friends with Charles? {is_friend(network, 'Alice', 'Charles')}")
print(f"Common friend between Eva and Deepa: {common_friend(network, 'Eva', 'Deepa')}")
print(f"Distance between Alice and Charles: {distance(network, 'Alice', 'Charles')}")
print(f"Diameter of the network: {diameter(network)}")
print("Updated Social Network:", network)
```

Output

Is Alice friends with Charles? False

Common friend between Eva and Deepa: None

Distance between Alice and Charles: 2

Diameter of the network: 4

Updated Social Network: {'Alice': ['Bob', 'Eva'], 'Eva': ['Alice', 'Deepa'], 'Bob': ['Alice', 'Charles'], 'Charles': ['Bob'], 'Deepa': ['Eva']}