AI Lab\BFS.py

```python
from collections import deque

class Graph:
    def __init__(self, directed=False):
        self.graph = {}
        self.directed = directed

    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        self.graph[u].append(v)

        if not self.directed:
            if v not in self.graph:
                self.graph[v] = []
            self.graph[v].append(u)

    def bfs(self, start_vertex):
        visited = set()
        queue = deque([start_vertex])
        visited.add(start_vertex)

        traversal = []

        while queue:
            vertex = queue.popleft()
            traversal.append(vertex)

            for neighbor in self.graph.get(vertex, []):
                if neighbor not in visited:
                    visited.add(neighbor)
                    queue.append(neighbor)

        return traversal

if __name__ == "__main__":
    g = Graph()
    g.add_edge(0, 1)
    g.add_edge(0, 2)
    g.add_edge(1, 2)
    g.add_edge(2, 0)
    g.add_edge(2, 3)
    g.add_edge(3, 3)

    print("BFS traversal starting from vertex 2:")
    print(g.bfs(2))

```


>_pwsh
Python_LocalVC
master ≡ ?1
2ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\BFS.py"
BFS traversal starting from vertex 2:
[2, 0, 1, 3]

AI Lab\Water_Jug_Problem.py

```python
from collections import deque

x = int(input("Enter capacity of Jug 1: "))
y = int(input("Enter capacity of Jug 2: "))
target = int(input("Enter Target to achieve: "))

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

if target > max(x, y):
    print("No solution possible: Target exceeds capacity of both jugs")
elif target % gcd(x, y) != 0:
    print("No solution possible: Target cannot be measured with these jug sizes")
else:

    visited = set()
    queue = deque([(0, 0, [])])
    solution_found = False

    while queue and not solution_found:
        jug1, jug2, steps = queue.popleft()

        if jug1 == target or jug2 == target:
            print("Solution found:")
            for i, step in enumerate(steps, 1):
                print(f"{i}. {step}")
            solution_found = True

        if (jug1, jug2) in visited:
            continue

        visited.add((jug1, jug2))

        if jug1 < x:
            queue.append((x, jug2, steps + [f"Fill jug 1: ({x}, {jug2})"]))

        if jug2 < y:
            queue.append((jug1, y, steps + [f"Fill jug 2: ({jug1}, {y})"]))

        if jug1 > 0:
            queue.append((0, jug2, steps + [f"Empty jug 1: (0, {jug2})"]))

        if jug2 > 0:
            queue.append((jug1, 0, steps + [f"Empty jug 2: ({jug1}, 0)"]))

        if jug1 > 0 and jug2 < y:
            pour = min(jug1, y - jug2)
            queue.append((jug1 - pour, jug2 + pour, steps + [f"Pour jug 1 to jug 2: ({jug1 - pour}, {jug2 + pour})"]))

```

```
        if jug2 > 0 and jug1 < x:
            pour = min(jug2, x - jug1)
            queue.append((jug1 + pour, jug2 - pour, steps + [f"Pour jug 2 to jug 1: ({jug1 + pour}, {jug2 - pour})"]))

    if not solution_found:
        print("No solution found")
```

```
>_pwsh  Python_LocalVC  master ≡ ?1  56s 844ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Water_Jug_Problem.py"
Enter capacity of Jug 1: 5
Enter capacity of Jug 2: 3
Enter Target to achieve: 4
Solution found:
1. Fill jug 1: (5, 0)
2. Pour jug 1 to jug 2: (2, 3)
3. Empty jug 2: (2, 0)
4. Pour jug 1 to jug 2: (0, 2)
5. Fill jug 1: (5, 2)
6. Pour jug 1 to jug 2: (4, 3)
```

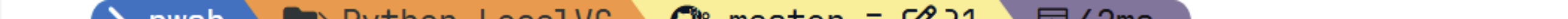
AI Lab\PunctuationRemoval.py

```python
import string

def remove_punctuation(input_string):
    translator = str.maketrans('', '', string.punctuation)
    return input_string.translate(translator)


sample_text = "Hi, I am Under the water! Here its too much raining :("
clean_text = remove_punctuation(sample_text)
print(f"Original: {sample_text}")
print(f"Cleaned: {clean_text}")

```




```
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\PunctuationRemoval.py"
Original: Hi, I am Under the water! Here its too much raining :(
Cleaned: Hi I am Under the water Here its too much raining
```

AI Lab\sentenceOrder.py

```python
def sort_sentence(sentence):
    words = sentence.split()
    words.sort(key=str.lower)
    sorted_sentence = ' '.join(words)
    return sorted_sentence

input_sentence = input("Enter a sentence: ")
result = sort_sentence(input_sentence)
print("Sorted sentence:", result)
```


>_pwsh
Python_LocalVC
master ≡ ?1
49ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\sentenceOrder.py"
Enter a sentence: The Fool that doesn't belong to this era;
Sorted sentence: belong doesn't era; Fool that The this to