Quiz No.1 (Skill Test)	
Course Code: CPE 201L	**Program:** BS Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 8/30/2025
Section: 2-A	**Date Submitted:** 8/30/2025
Name: Delinia, Filjohn B.	**Instructor:** Engr. Maria Rizette H. Sayo

1.Objectives

1. Choose only one (1) Data Structure (Array, Linked-List (Singly, Double), Stack, Queue)
2. Create a python program that appends each character of your Full name and traverse each character.
3. Save your python program as Skill-Test in your Colab and GitHub.

2. Discussion

I implemented a queue in Python, which is a linear data structure that follows FIFO (First In, First Out) rule. This means that elements are removed in the same order they were added, using two main operations: enqueue (adding) and dequeue (removing). Writing the queue logic manually as opposed to relying on built-in tools, shows clearly how this structure works in practice

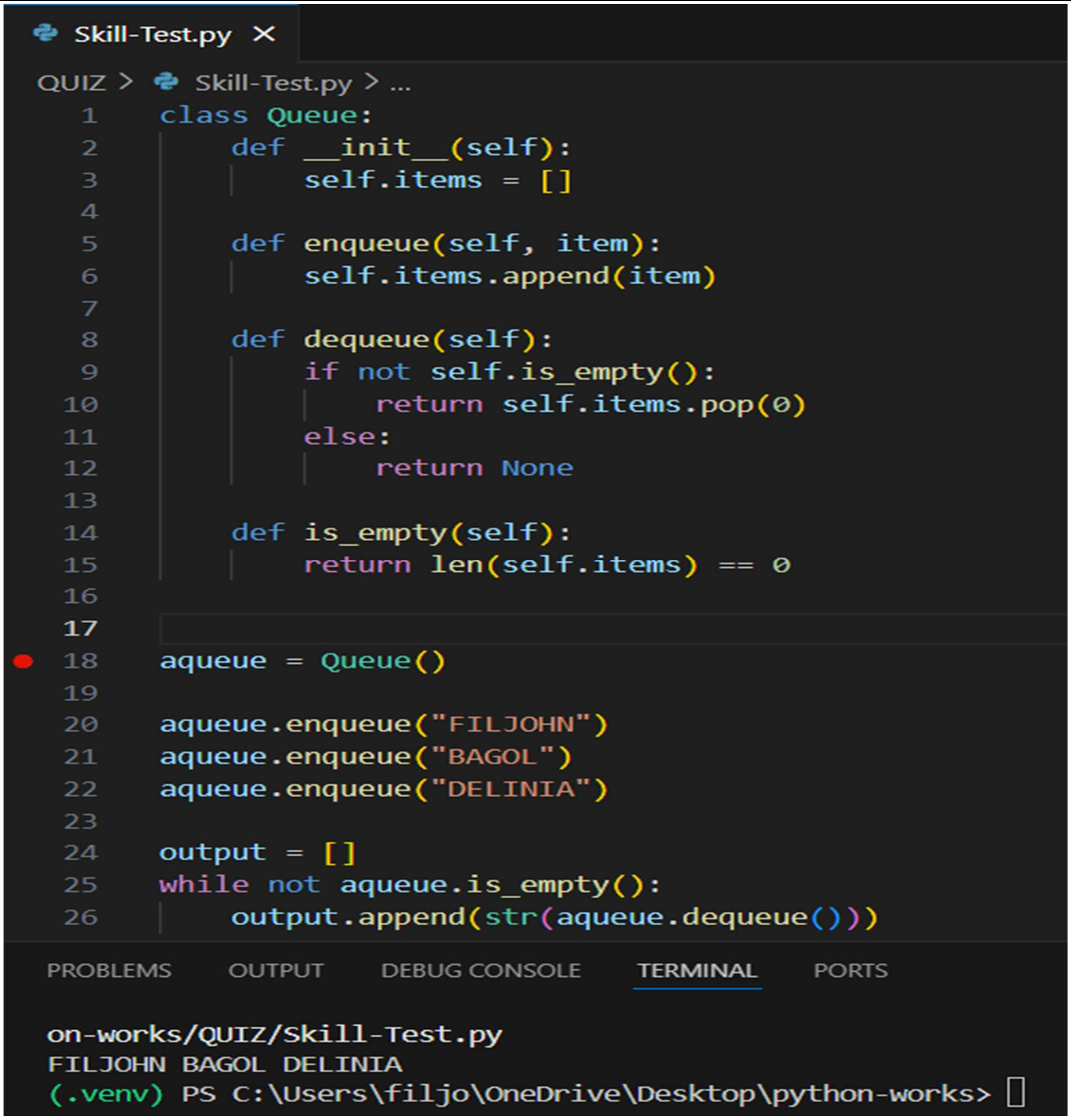
3. Materials and Equipment

1. Github
2. Colab
3. MS Word

4. Procedure

- Define a Queue class with methods for enqueue, dequeue, and is_empty.
- Create a queue object (aqueue = Queue()).
- Add elements into the queue using enqueue().
- Use a loop with dequeue() to traverse elements while the queue is not empty.
- Print the output in one line to demonstrate the FIFO order.

5. Output



```python
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.items.pop(0)
        else:
            return None

    def is_empty(self):
        return len(self.items) == 0


aqueue = Queue()

aqueue.enqueue("FILJOHN")
aqueue.enqueue("BAGOL")
aqueue.enqueue("DELINIA")

output = []
while not aqueue.is_empty():
    output.append(str(aqueue.dequeue()))
```

```
on-works/QUIZ/Skill-Test.py
FILJOHN BAGOL DELINIA
(.venv) PS C:\Users\filjo\OneDrive\Desktop\python-works>
```

7. Conclusion

In this activity, I successfully implemented a queue in Python to demonstrate the FIFO principle. By using enqueue and dequeue operations, we were able to store and retrieve data in the correct sequence. This activity reinforced the importance of queues as a fundamental data structure and showed how they can be implemented and traversed effectively using basic programming concepts.

8. References

1. GeeksforGeeks. Introduction to Queue Data Stucture and Algorithm Tutorials. In GeeksforGeeks, https://www.geeksforgeeks.org/dsa/introduction-to-queue-data-structure-and-algorithm-tutorials/
2. Wikipedia contributors. Queue. https://en.wikipedia.org/wiki/Queue_(abstract_data_type)

9. Source Code

```
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.items.pop(0)
        else:
            return None

    def is_empty(self):
        return len(self.items) == 0


aqueue = Queue()

aqueue.enqueue("FILJOHN")
aqueue.enqueue("BAGOL")
aqueue.enqueue("DELINIA")

output = []
while not aqueue.is_empty():
    output.append(str(aqueue.dequeue()))

print(" ".join(output))
```