

Python Programming Lab Task Sets

Set 5

1. Q1. Generate Python code for the Fibonacci sequence with Cursor AI. Refine your prompts to handle only positive integers and record the changes and outcomes.

(Each question carries 5 marks)

2. Q2. Write a Python queue class (`enqueue`, `dequeue`, `peek`). Use an AI assistant and test the class with multiple scenarios. Describe how AI-generated code manages edge cases and how you improved it.

(Each question carries 5 marks)

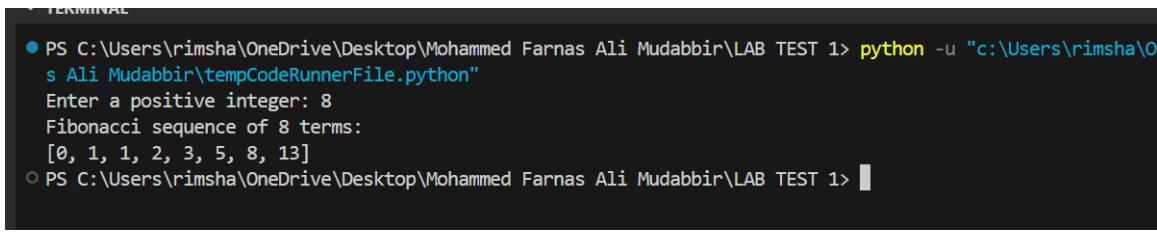
ANSWERS:

Q1. Generate Python code for the Fibonacci sequence with Cursor AI. Refine your prompts to handle only positive integers and record the changes and outcomes.

PROMPT: Write Python code that takes a positive integer as user input and prints the Fibonacci sequence with input validation. Include example output.

```
(Task 1.py) C:\> Users > rimsha > OneDrive > Desktop > AIPP TEST 1 > (Task 1.py) main
1 # ...existing code...
2 def fibonacci(n):
3     sequence = []
4     a, b = 0, 1
5     for _ in range(n):
6         sequence.append(a)
7         a, b = b, a + b
8     return sequence
9
10 def main():
11     while True:
12         s = input("Enter a positive integer: ").strip()
13         if not s:
14             print("Input cannot be empty. Try again.")
15             continue
16         try:
17             num = int(s)
18             if num <= 0:
19                 print("Please enter a positive integer greater than 0.")
20                 continue
21             break
22         except ValueError:
23             print("Invalid input! Please enter a valid positive integer.")
24
(Task 1.py) C:\> Users > rimsha > OneDrive > Desktop > AIPP TEST 1 > (Task 1.py) main
10 def main():
11     try:
12         num = int(s)
13         if num <= 0:
14             print("Please enter a positive integer greater than 0.")
15             continue
16         break
17     except ValueError:
18         print("Invalid input! Please enter a valid positive integer.")
19
20     seq = fibonacci(num)
21     print(f"Fibonacci sequence of {num} term{'s' if num != 1 else ''}:")
22     print(seq)
23
24     if __name__ == "__main__":
25         main()
26     # ...existing code...
```

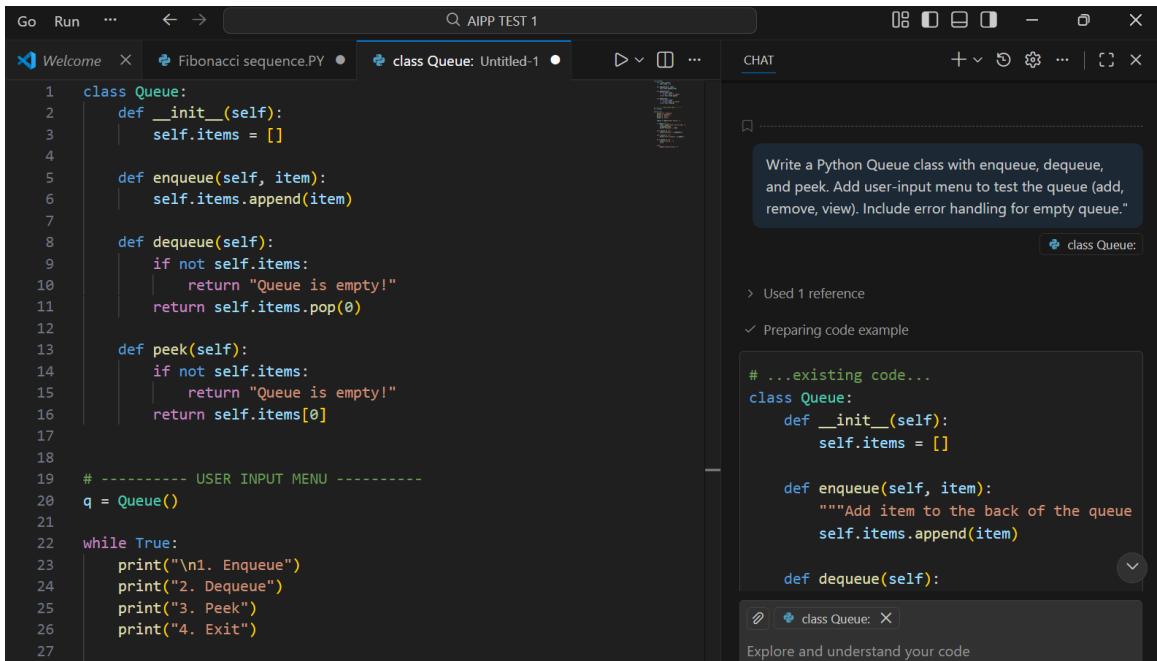
PRACTICAL OUTPUT:



```
TERMINAL
● PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB TEST 1> python -u "c:\Users\rimsha\0
s Ali Mudabbir\tempCodeRunnerFile.py"
Enter a positive integer: 8
Fibonacci sequence of 8 terms:
[0, 1, 1, 2, 3, 5, 8, 13]
○ PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB TEST 1>
```

Q2. Write a Python queue class (`enqueue`, `dequeue`, `peek`). Use an AI assistant and test the class with multiple scenarios. Describe how AI-generated code manages edge cases and how you improved it.

PROMPT: Write a Python Queue class with enqueue, dequeue, and peek. Add user-input menu to test the queue (add, remove, view). Include error handling for empty queue."



```
Go Run ... ← → AIPP TEST 1 CHAT + ...
Welcome ✎ Fibonacci sequence.PY • class Queue: Untitled-1 •
1 class Queue:
2     def __init__(self):
3         self.items = []
4
5     def enqueue(self, item):
6         self.items.append(item)
7
8     def dequeue(self):
9         if not self.items:
10             return "Queue is empty!"
11         return self.items.pop(0)
12
13     def peek(self):
14         if not self.items:
15             return "Queue is empty!"
16         return self.items[0]
17
18
19 # ----- USER INPUT MENU -----
20 q = Queue()
21
22 while True:
23     print("\n1. Enqueue")
24     print("2. Dequeue")
25     print("3. Peek")
26     print("4. Exit")
27
```

Write a Python Queue class with enqueue, dequeue, and peek. Add user-input menu to test the queue (add, remove, view). Include error handling for empty queue."

Used 1 reference

Preparing code example

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

    def enqueue(self, item):
        """Add item to the back of the queue
        self.items.append(item)

    def dequeue(self):
```

Explore and understand your code

The screenshot shows a code editor interface with several tabs open. The active tab is 'class Queue: Untitled-1' containing the following Python code:

```
21 while True:
22     print("1. Enqueue")
23     print("2. Dequeue")
24     print("3. Peek")
25     print("4. Exit")
26
27     choice = input("Enter choice: ")
28
29     if choice == "1":
30         item = input("Enter item to add: ")
31         q.enqueue(item)
32         print("Enqueued:", item)
33
34     elif choice == "2":
35         print("Dequeued:", q.dequeue())
36
37     elif choice == "3":
38         print("Front element:", q.peek())
39
40     elif choice == "4":
41         print("Exiting...")
42         break
43
44     else:
45         print("Invalid choice !")
46
47
```

To the right of the code editor is a 'CHAT' pane with a message from 'class Queue' asking for a Python Queue class implementation. Below the code editor is a terminal window showing the execution of the program. The terminal output is as follows:

```
PS C:\Users\rimsha\OneDrive\Desktop\AIPP TEST 1> python -u "c:\Users\rimsha\OneDrive\Desktop\queueRunnerFile.python"
1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 1
Enter item to add: A
Enqueued: A

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 1
Enter item to add: B
Enqueued: B

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 2
Dequeued: A
```

PRACTICAL OUT:

The terminal window shows the execution of the 'queueRunnerFile.python' script. The user interacts with the program, choosing to enqueue items 'A' and 'B', and then dequeuing item 'A'.

```
PS C:\Users\rimsha\OneDrive\Desktop\AIPP TEST 1> python -u "c:\Users\rimsha\OneDrive\Desktop\queueRunnerFile.python"
1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 1
Enter item to add: A
Enqueued: A

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 1
Enter item to add: B
Enqueued: B

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 2
Dequeued: A
```

```
PS C:\Users\rimsha\OneDrive\Desktop\AIPP TEST 1> python -u "c:\Users\rimsha\OneDrive\AIPP\queue.py"
Enter choice: 2
Dequeued: A

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 2
Dequeued: B

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 2
Dequeued: Queue is empty!

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 4
Exiting...
PS C:\Users\rimsha\OneDrive\Desktop\AIPP TEST 1>
```