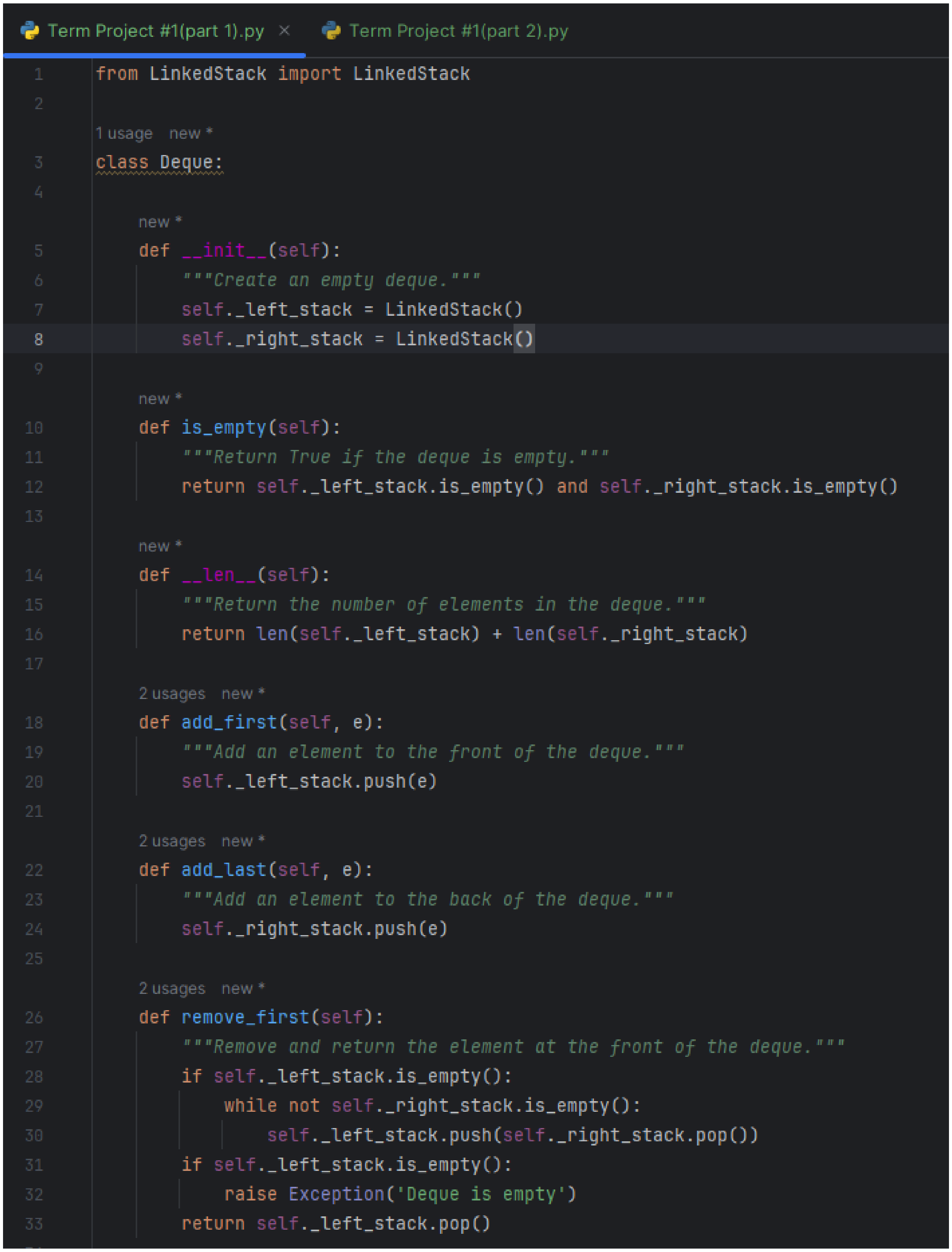
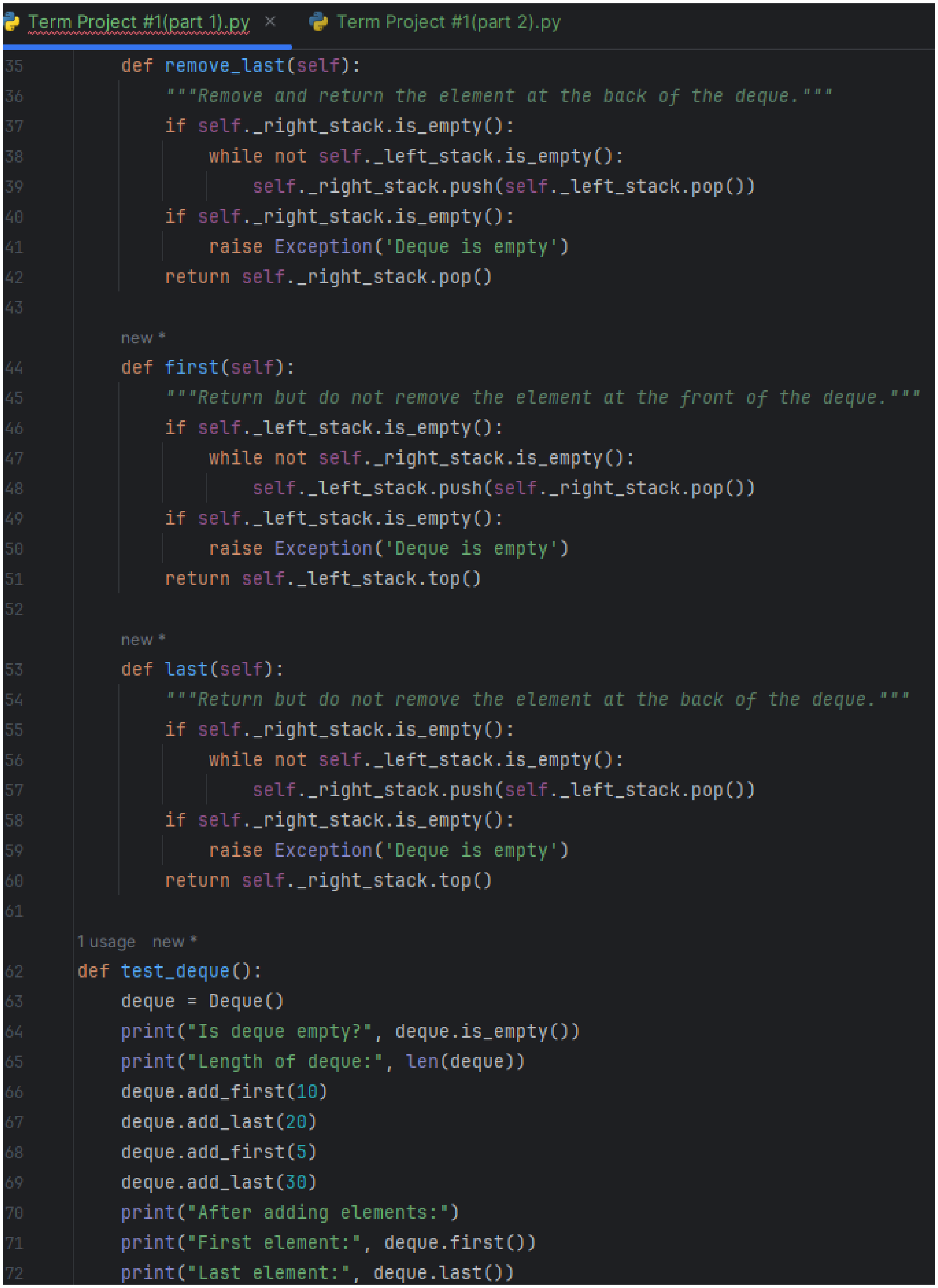


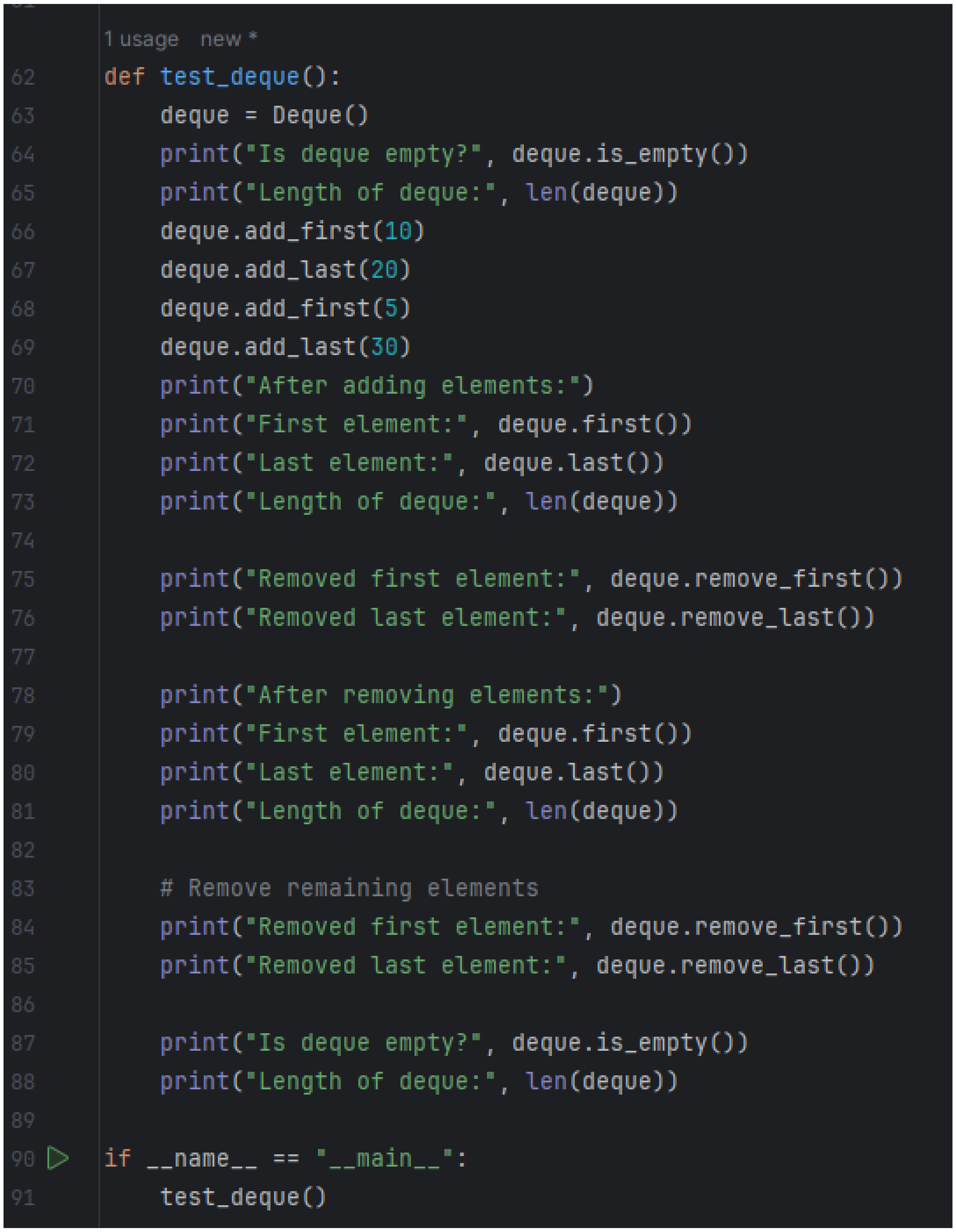
Term Project #1(part 1).py ×   Term Project #1(part 2).py

```python
from LinkedStack import LinkedStack

1 usage  new *
class Deque:

    new *
    def __init__(self):
        """Create an empty deque."""
        self._left_stack = LinkedStack()
        self._right_stack = LinkedStack()

    new *
    def is_empty(self):
        """Return True if the deque is empty."""
        return self._left_stack.is_empty() and self._right_stack.is_empty()

    new *
    def __len__(self):
        """Return the number of elements in the deque."""
        return len(self._left_stack) + len(self._right_stack)

    2 usages  new *
    def add_first(self, e):
        """Add an element to the front of the deque."""
        self._left_stack.push(e)

    2 usages  new *
    def add_last(self, e):
        """Add an element to the back of the deque."""
        self._right_stack.push(e)

    2 usages  new *
    def remove_first(self):
        """Remove and return the element at the front of the deque."""
        if self._left_stack.is_empty():
            while not self._right_stack.is_empty():
                self._left_stack.push(self._right_stack.pop())
        if self._left_stack.is_empty():
            raise Exception('Deque is empty')
        return self._left_stack.pop()
```

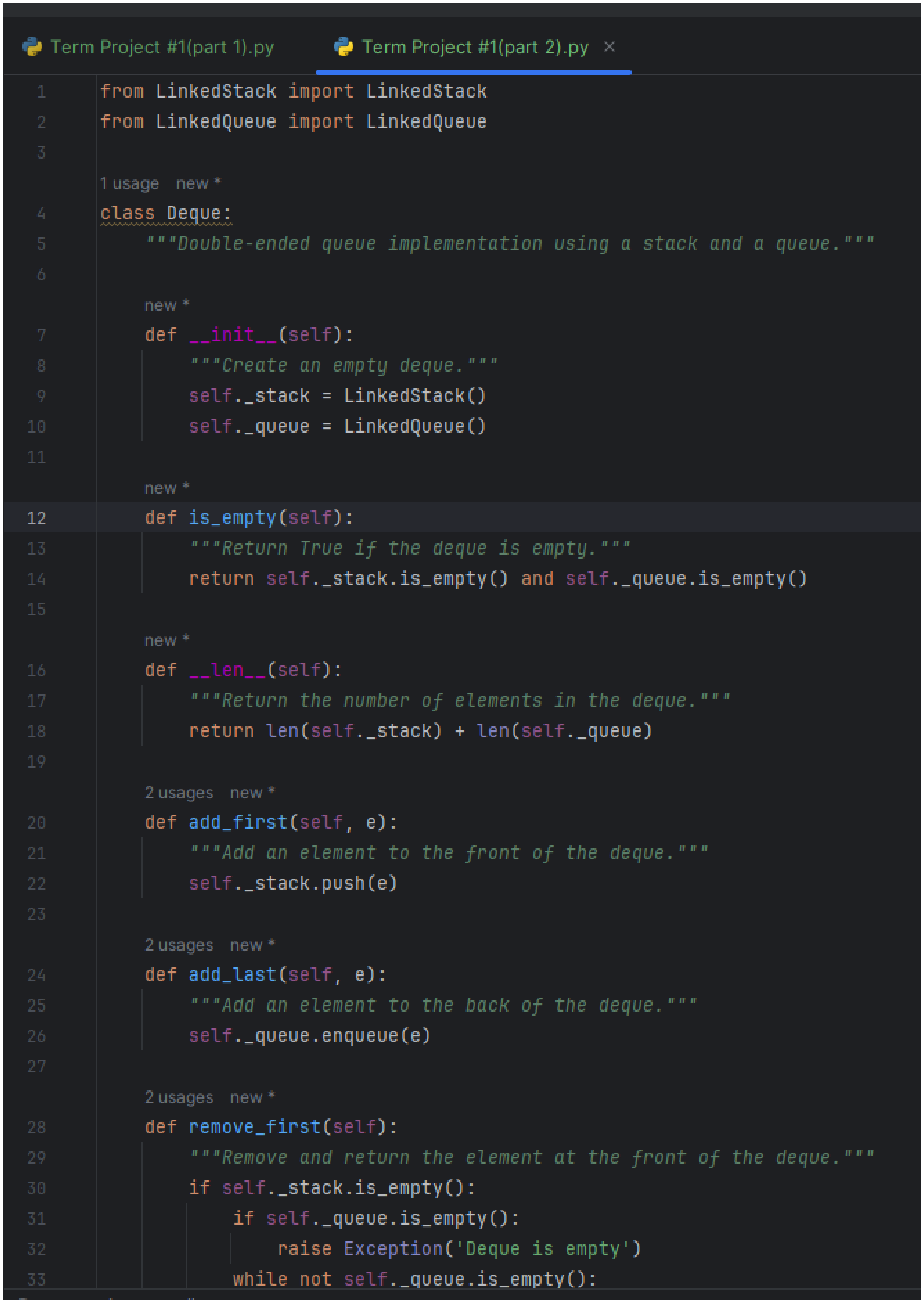


Term Project #1(part 1).py × Term Project #1(part 2).py

```
    def remove_last(self):
        """Remove and return the element at the back of the deque."""
        if self._right_stack.is_empty():
            while not self._left_stack.is_empty():
                self._right_stack.push(self._left_stack.pop())
        if self._right_stack.is_empty():
            raise Exception('Deque is empty')
        return self._right_stack.pop()

    new *
    def first(self):
        """Return but do not remove the element at the front of the deque."""
        if self._left_stack.is_empty():
            while not self._right_stack.is_empty():
                self._left_stack.push(self._right_stack.pop())
        if self._left_stack.is_empty():
            raise Exception('Deque is empty')
        return self._left_stack.top()

    new *
    def last(self):
        """Return but do not remove the element at the back of the deque."""
        if self._right_stack.is_empty():
            while not self._left_stack.is_empty():
                self._right_stack.push(self._left_stack.pop())
        if self._right_stack.is_empty():
            raise Exception('Deque is empty')
        return self._right_stack.top()

1 usage   new *
def test_deque():
    deque = Deque()
    print("Is deque empty?", deque.is_empty())
    print("Length of deque:", len(deque))
    deque.add_first(10)
    deque.add_last(20)
    deque.add_first(5)
    deque.add_last(30)
    print("After adding elements:")
    print("First element:", deque.first())
    print("Last element:", deque.last())
```

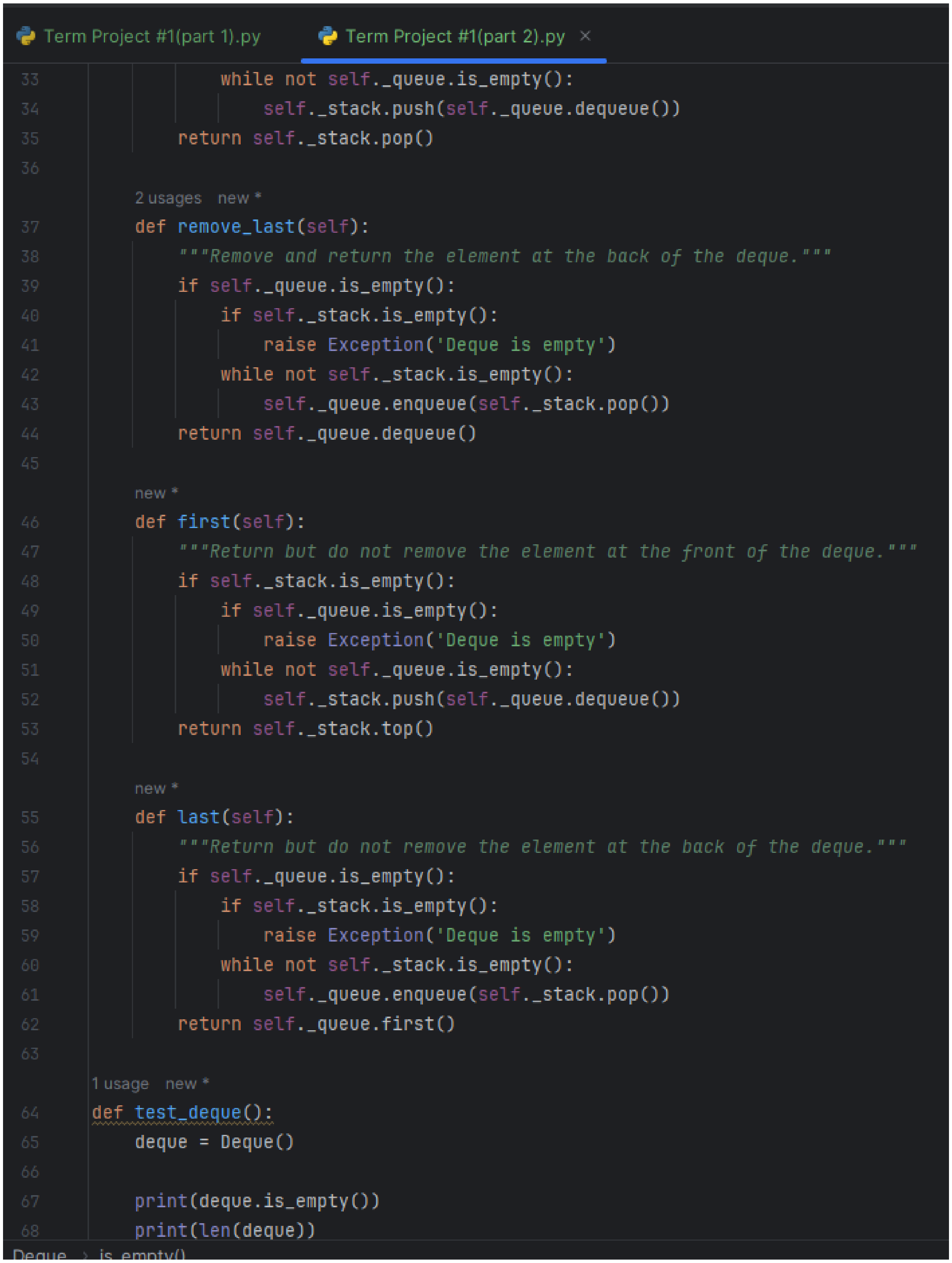


```python
def test_deque():
    deque = Deque()
    print("Is deque empty?", deque.is_empty())
    print("Length of deque:", len(deque))
    deque.add_first(10)
    deque.add_last(20)
    deque.add_first(5)
    deque.add_last(30)
    print("After adding elements:")
    print("First element:", deque.first())
    print("Last element:", deque.last())
    print("Length of deque:", len(deque))

    print("Removed first element:", deque.remove_first())
    print("Removed last element:", deque.remove_last())

    print("After removing elements:")
    print("First element:", deque.first())
    print("Last element:", deque.last())
    print("Length of deque:", len(deque))

    # Remove remaining elements
    print("Removed first element:", deque.remove_first())
    print("Removed last element:", deque.remove_last())

    print("Is deque empty?", deque.is_empty())
    print("Length of deque:", len(deque))

if __name__ == "__main__":
    test_deque()
```



Term Project #1(part 1).py   Term Project #1(part 2).py

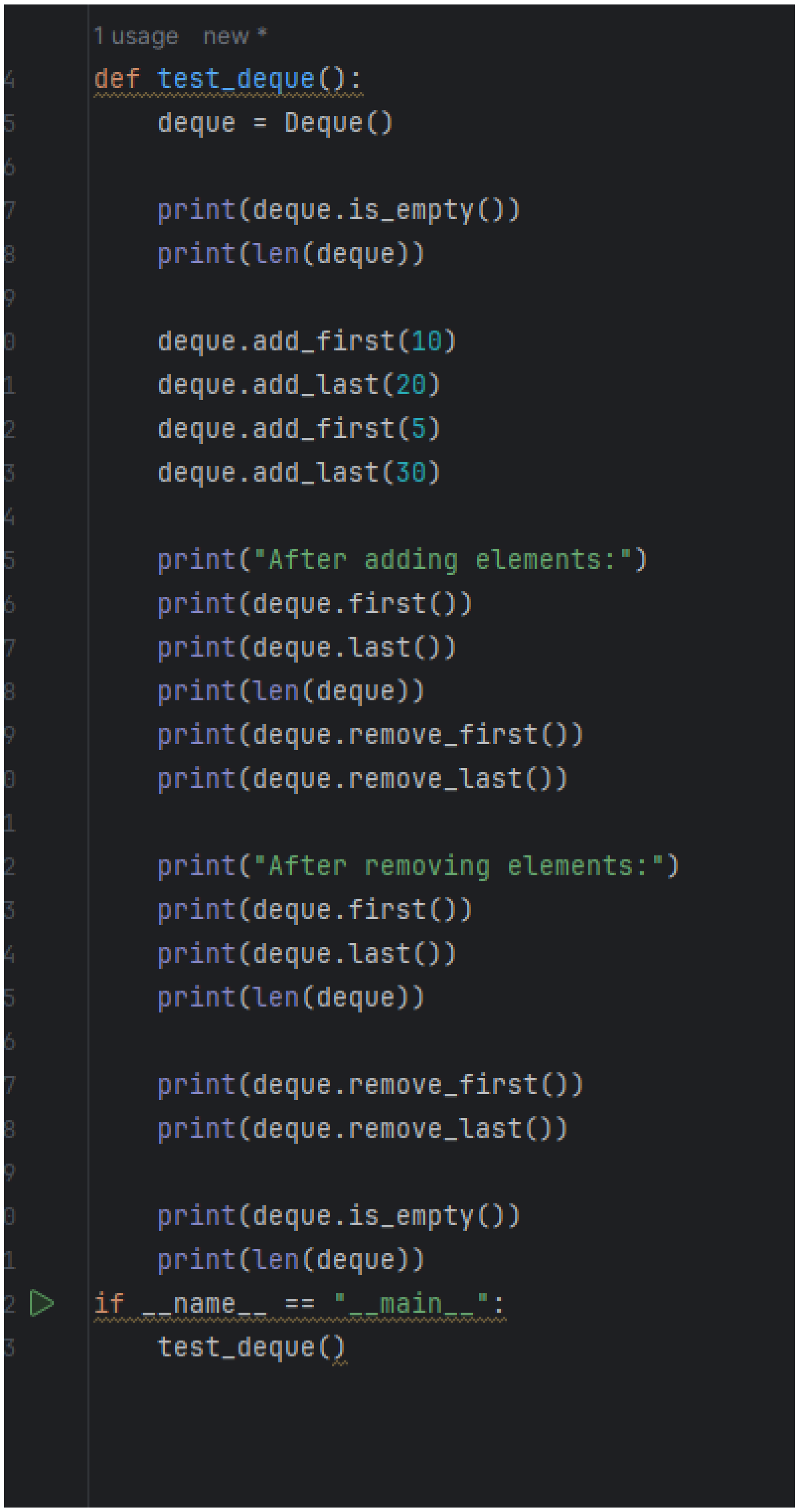
```python
from LinkedStack import LinkedStack
from LinkedQueue import LinkedQueue

1 usage  new *
class Deque:
    """Double-ended queue implementation using a stack and a queue."""

    new *
    def __init__(self):
        """Create an empty deque."""
        self._stack = LinkedStack()
        self._queue = LinkedQueue()

    new *
    def is_empty(self):
        """Return True if the deque is empty."""
        return self._stack.is_empty() and self._queue.is_empty()

    new *
    def __len__(self):
        """Return the number of elements in the deque."""
        return len(self._stack) + len(self._queue)

    2 usages  new *
    def add_first(self, e):
        """Add an element to the front of the deque."""
        self._stack.push(e)

    2 usages  new *
    def add_last(self, e):
        """Add an element to the back of the deque."""
        self._queue.enqueue(e)

    2 usages  new *
    def remove_first(self):
        """Remove and return the element at the front of the deque."""
        if self._stack.is_empty():
            if self._queue.is_empty():
                raise Exception('Deque is empty')
            while not self._queue.is_empty():
```



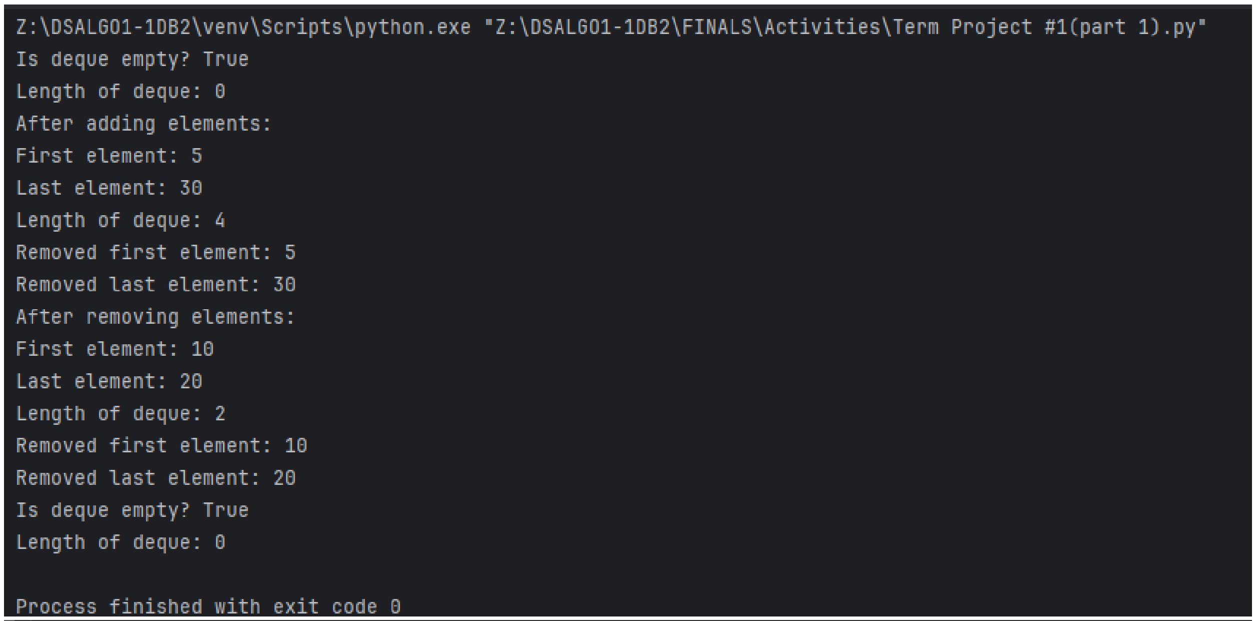
Term Project #1(part 1).py   Term Project #1(part 2).py

```
            while not self._queue.is_empty():
                self._stack.push(self._queue.dequeue())
        return self._stack.pop()

    2 usages   new *
    def remove_last(self):
        """Remove and return the element at the back of the deque."""
        if self._queue.is_empty():
            if self._stack.is_empty():
                raise Exception('Deque is empty')
            while not self._stack.is_empty():
                self._queue.enqueue(self._stack.pop())
        return self._queue.dequeue()

    new *
    def first(self):
        """Return but do not remove the element at the front of the deque."""
        if self._stack.is_empty():
            if self._queue.is_empty():
                raise Exception('Deque is empty')
            while not self._queue.is_empty():
                self._stack.push(self._queue.dequeue())
        return self._stack.top()

    new *
    def last(self):
        """Return but do not remove the element at the back of the deque."""
        if self._queue.is_empty():
            if self._stack.is_empty():
                raise Exception('Deque is empty')
            while not self._stack.is_empty():
                self._queue.enqueue(self._stack.pop())
        return self._queue.first()

1 usage   new *
def test_deque():
    deque = Deque()

    print(deque.is_empty())
    print(len(deque))
```

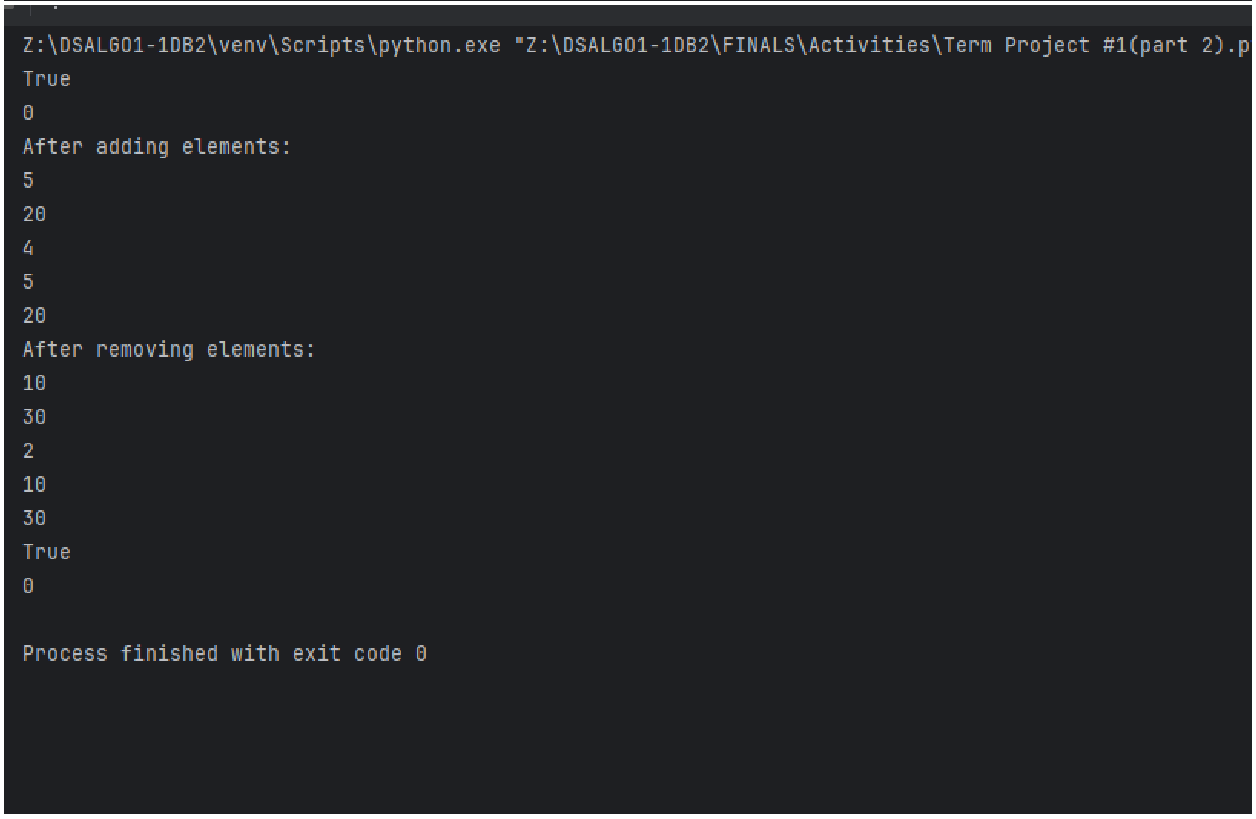
Deque 〉 is_empty()



```
1 usage   new *
def test_deque():
    deque = Deque()

    print(deque.is_empty())
    print(len(deque))

    deque.add_first(10)
    deque.add_last(20)
    deque.add_first(5)
    deque.add_last(30)

    print("After adding elements:")
    print(deque.first())
    print(deque.last())
    print(len(deque))
    print(deque.remove_first())
    print(deque.remove_last())

    print("After removing elements:")
    print(deque.first())
    print(deque.last())
    print(len(deque))

    print(deque.remove_first())
    print(deque.remove_last())

    print(deque.is_empty())
    print(len(deque))
if __name__ == "__main__":
    test_deque()
```



```
Z:\DSALGO1-1DB2\venv\Scripts\python.exe "Z:\DSALGO1-1DB2\FINALS\Activities\Term Project #1(part 1).py"
Is deque empty? True
Length of deque: 0
After adding elements:
First element: 5
Last element: 30
Length of deque: 4
Removed first element: 5
Removed last element: 30
After removing elements:
First element: 10
Last element: 20
Length of deque: 2
Removed first element: 10
Removed last element: 20
Is deque empty? True
Length of deque: 0

Process finished with exit code 0
```



```
Z:\DSALGO1-1DB2\venv\Scripts\python.exe "Z:\DSALGO1-1DB2\FINALS\Activities\Term Project #1(part 2).p
True
0
After adding elements:
5
20
4
5
20
After removing elements:
10
30
2
10
30
True
0

Process finished with exit code 0
```