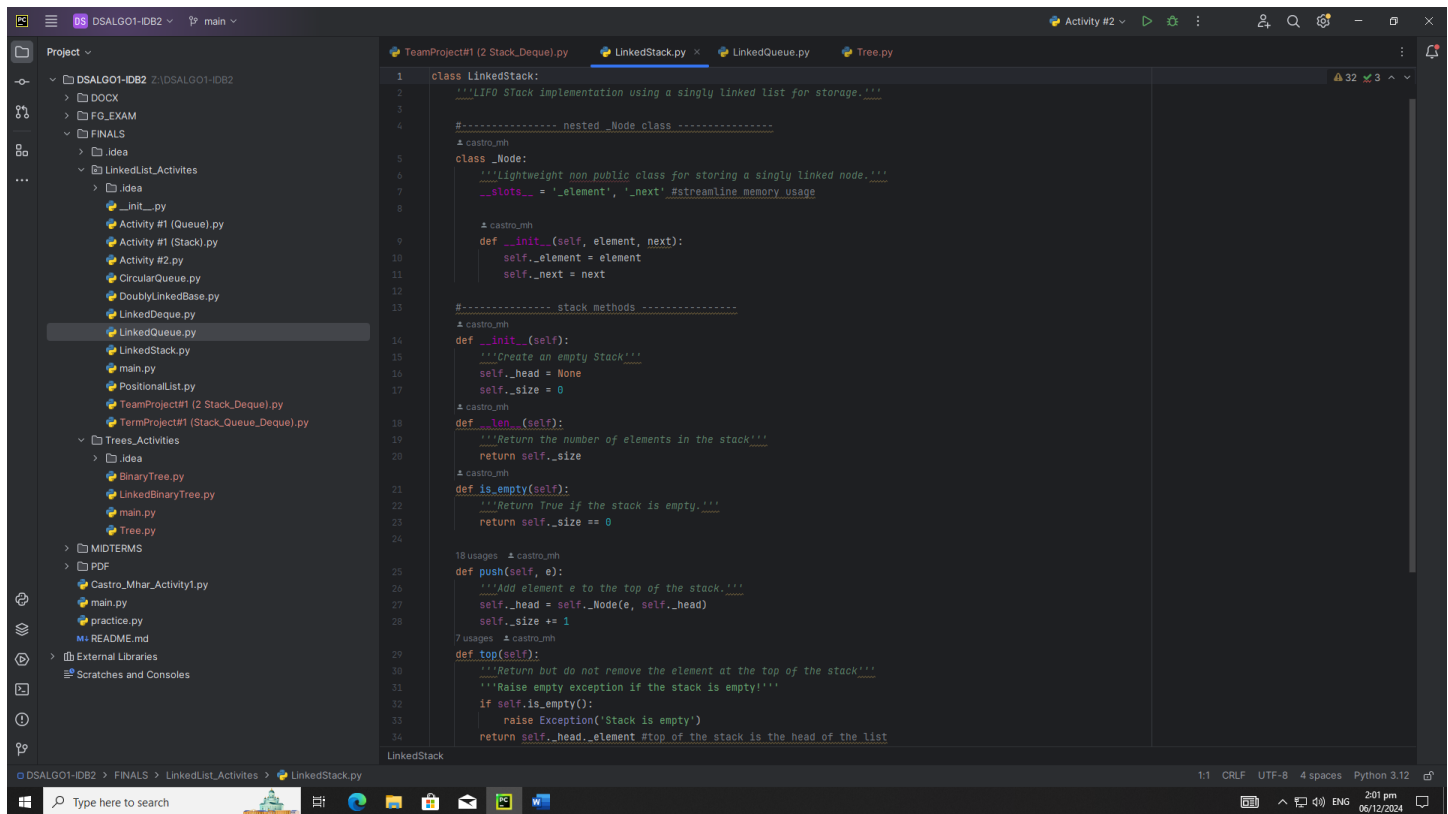
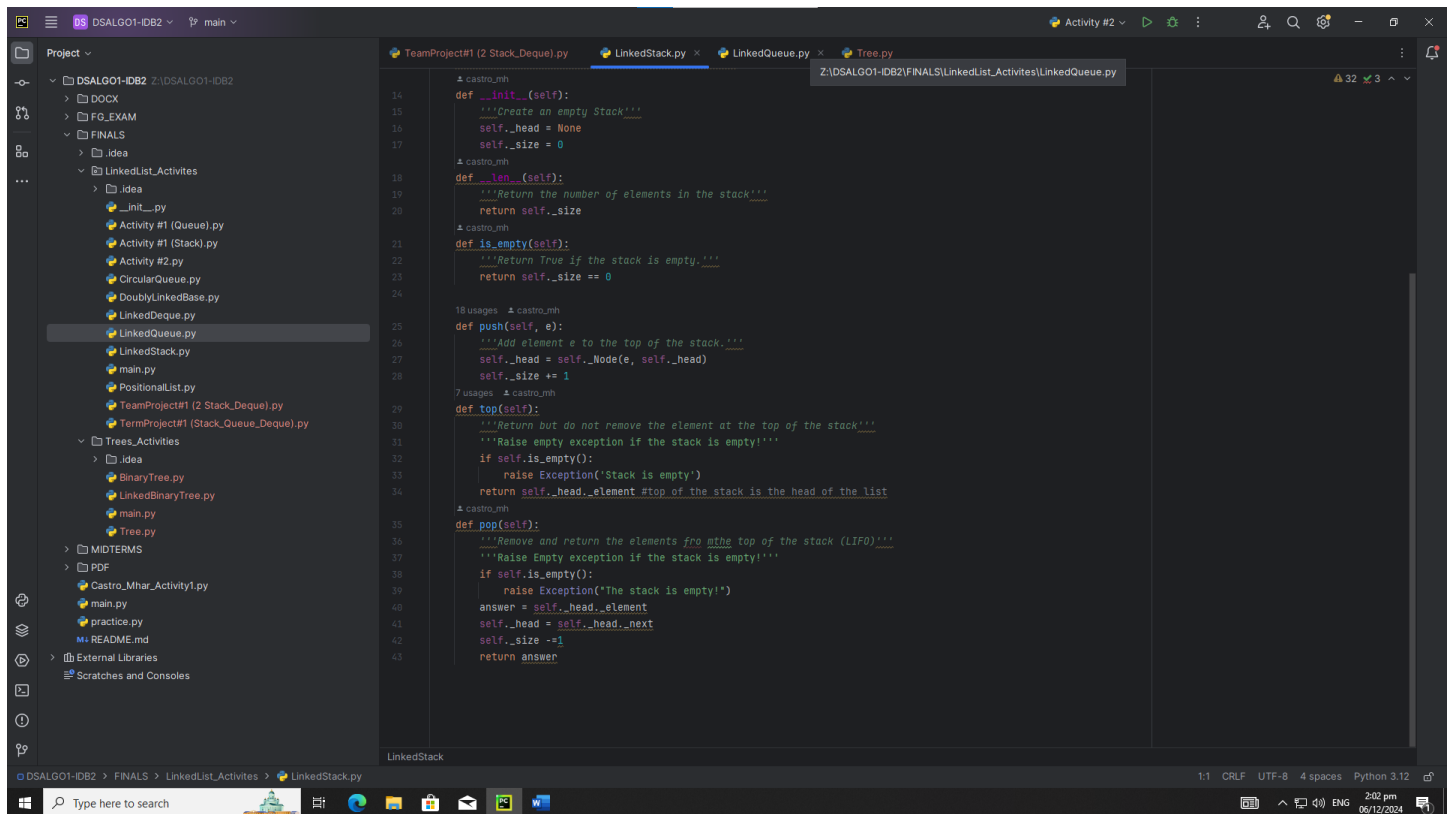


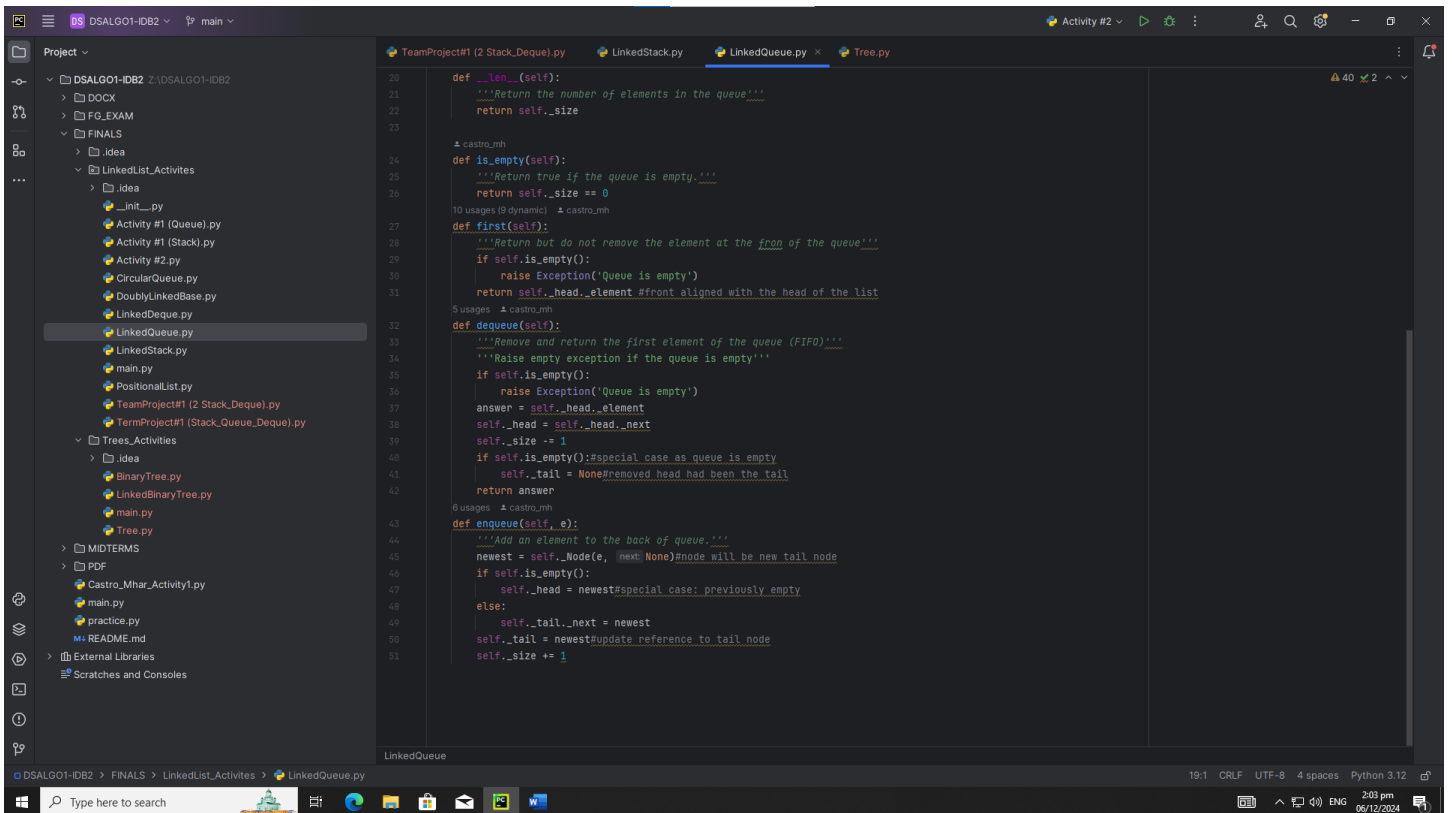
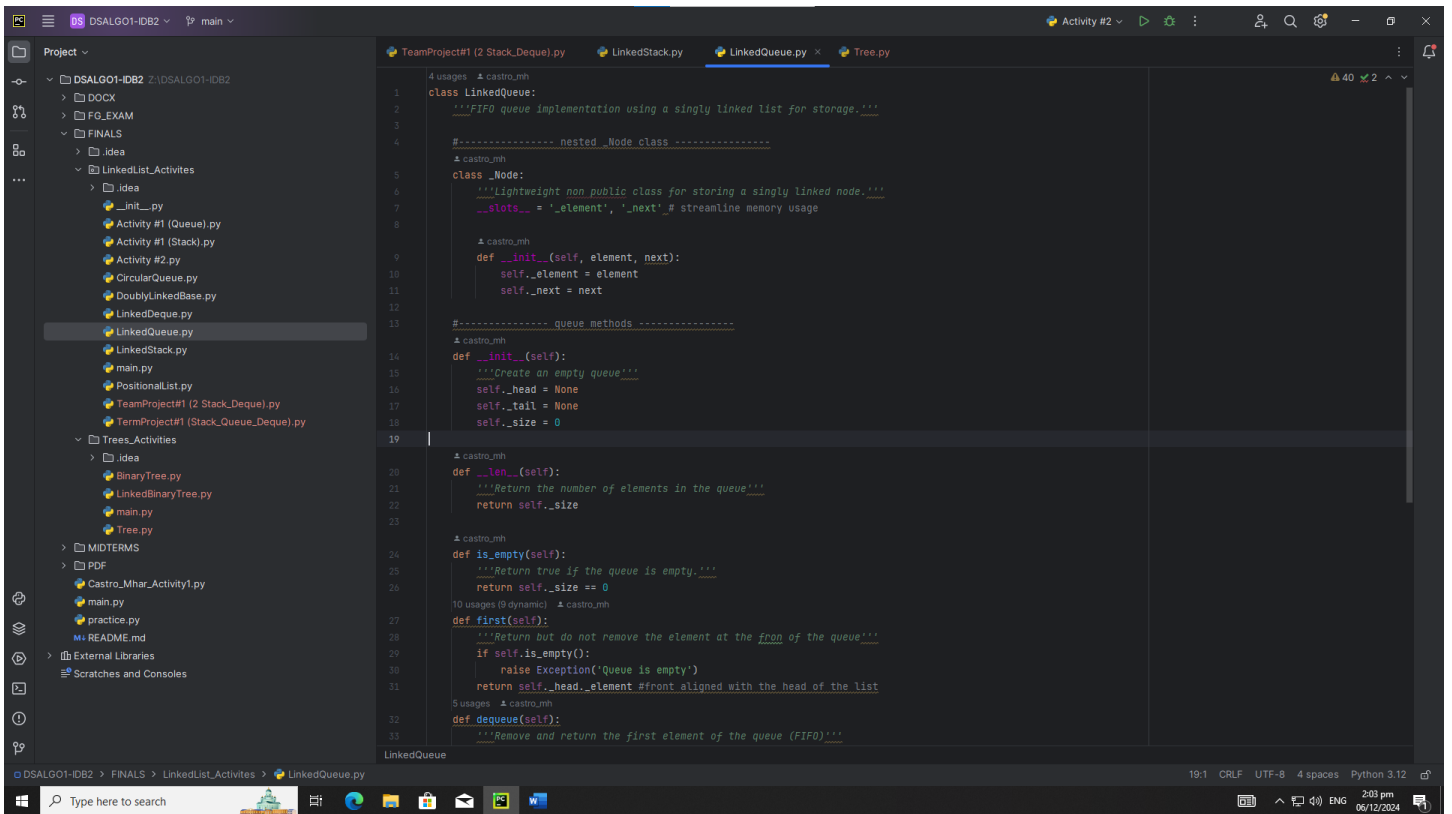
SOURCE CODE:



```
1 class LinkedStack:
2     """LIFO Stack implementation using a singly linked list for storage."""
3
4     #----- nested _Node class -----
5     class _Node:
6         """Lightweight non public class for storing a singly linked node."""
7         __slots__ = '_element', '_next' #streamline memory usage
8
9         @castro_mh
10        def __init__(self, element, next):
11            self._element = element
12            self._next = next
13
14    #----- stack methods -----
15    @castro_mh
16    def __init__(self):
17        """Create an empty Stack"""
18        self._head = None
19        self._size = 0
20
21    @castro_mh
22    def __len__(self):
23        """Return the number of elements in the stack"""
24        return self._size
25
26    @castro_mh
27    def is_empty(self):
28        """Return True if the stack is empty."""
29        return self._size == 0
30
31    18 usages @castro_mh
32    def push(self, e):
33        """Add element e to the top of the stack."""
34        self._head = self._Node(e, self._head)
35        self._size += 1
36
37    7 usages @castro_mh
38    def top(self):
39        """Return but do not remove the element at the top of the stack"""
40        """Raise empty exception if the stack is empty!"""
41        if self.is_empty():
42            raise Exception('Stack is empty')
43        return self._head._element #top of the stack is the head of the list
44
45 LinkedStack
```



```
14 @castro_mh
15 def __init__(self):
16     """Create an empty Stack"""
17     self._head = None
18     self._size = 0
19
20 @castro_mh
21 def __len__(self):
22     """Return the number of elements in the stack"""
23     return self._size
24
25 @castro_mh
26 def is_empty(self):
27     """Return True if the stack is empty."""
28     return self._size == 0
29
30 18 usages @castro_mh
31 def push(self, e):
32     """Add element e to the top of the stack."""
33     self._head = self._Node(e, self._head)
34     self._size += 1
35
36 7 usages @castro_mh
37 def top(self):
38     """Return but do not remove the element at the top of the stack"""
39     """Raise empty exception if the stack is empty!"""
40     if self.is_empty():
41         raise Exception('Stack is empty')
42     return self._head._element #top of the stack is the head of the list
43
44 @castro_mh
45 def pop(self):
46     """Remove and return the elements fro mthe top of the stack (LIFO)"""
47     """Raise Empty exception if the stack is empty!"""
48     if self.is_empty():
49         raise Exception('The stack is empty!')
50     answer = self._head._element
51     self._head = self._head._next
52     self._size -= 1
53     return answer
```



DSALGO1-IDB2 main

Project

- DSALGO1-IDB2 Z:\DSALGO1-IDB2
 - DOXX
 - FG_EXAM
 - FINALS
 - .idea
 - LinkedList_Activities
 - .idea
 - __init__.py
 - Activity #1 (Queue).py
 - Activity #1 (Stack).py
 - Activity #2.py
 - CircularQueue.py
 - DoublyLinkedListBase.py
 - LinkedDeque.py
 - LinkedQueue.py
 - LinkedStack.py
 - main.py
 - PositionalList.py
 - TeamProject#1 (2 Stack_Deque).py
 - TermProject#1 (Stack_Queue_Deque).py
 - Trees_Activities
 - .idea
 - BinaryTree.py
 - LinkedBinaryTree.py
 - main.py
 - Tree.py
 - MIDTERMS
 - PDF
 - Castro_Mhar_Activity1.py
 - main.py
 - practice.py
 - README.md
 - External Libraries
 - Scratches and Consoles

TeamProject#1 (2 Stack_Deque).py

```
1 from LinkedStack import LinkedStack
2
3
4 class LinkedDeque:
5     def __init__(self):
6         self.front_stack = LinkedStack()
7         self.back_stack = LinkedStack()
8
9     def len(self):
10        return len(self.front_stack) + len(self.back_stack)
11
12    def is_empty(self):
13        return self.len() == 0
14
15    3 usages
16    def add_last(self, e):
17        self.back_stack.push(e)
18
19    1 usage
20    def add_first(self, e):
21        self.front_stack.push(e)
22
23    1 usage
24    def delete_last(self):
25        if self.is_empty():
26            raise Exception('Deque is empty')
27
28        if self.back_stack.is_empty():
29            while not self.front_stack.is_empty():
30                self.back_stack.push(self.front_stack.pop())
31
32        return self.back_stack.pop()
33
34    1 usage
35    def delete_first(self):
36        if self.is_empty():
37            raise Exception('Deque is empty')
38
39        if self.front_stack.is_empty():
40            while not self.back_stack.is_empty():
41                self.front_stack.push(self.back_stack.pop())
42
43        return self.front_stack.pop()
```

1:1 CRLF UTF-8 4 spaces Python 3.12 2:03 pm 06/12/2024

DSALGO1-IDB2 main

Project

- DSALGO1-IDB2 Z:\DSALGO1-IDB2
 - DOXX
 - FG_EXAM
 - FINALS
 - .idea
 - LinkedList_Activities
 - .idea
 - __init__.py
 - Activity #1 (Queue).py
 - Activity #1 (Stack).py
 - Activity #2.py
 - CircularQueue.py
 - DoublyLinkedListBase.py
 - LinkedDeque.py
 - LinkedQueue.py
 - LinkedStack.py
 - main.py
 - PositionalList.py
 - TeamProject#1 (2 Stack_Deque).py
 - TermProject#1 (Stack_Queue_Deque).py
 - Trees_Activities
 - .idea
 - BinaryTree.py
 - LinkedBinaryTree.py
 - main.py
 - Tree.py
 - MIDTERMS
 - PDF
 - Castro_Mhar_Activity1.py
 - main.py
 - practice.py
 - README.md
 - External Libraries
 - Scratches and Consoles

TeamProject#1 (2 Stack_Deque).py

```
35 if self.front_stack.is_empty():
36     while not self.back_stack.is_empty():
37         self.front_stack.push(self.back_stack.pop())
38
39     return self.front_stack.pop()
40
41 10 usages (9 dynamic)
42 def first(self):
43     if self.is_empty():
44         raise Exception('Deque is empty')
45
46     if self.front_stack.is_empty():
47         while not self.back_stack.is_empty():
48             self.front_stack.push(self.back_stack.pop())
49
50     return self.front_stack.top()
51
52 5 usages (4 dynamic)
53 def last(self):
54     if self.is_empty():
55         raise Exception('Deque is empty')
56
57     if self.back_stack.is_empty():
58         while not self.front_stack.is_empty():
59             self.back_stack.push(self.front_stack.pop())
60
61     return self.back_stack.top()
62
63 dq = LinkedDeque()
64 dq.add_last(1)
65 dq.add_last(2)
66 dq.add_last(3)
67 dq.add_first(0)
68 print(dq.delete_last())
69 print(dq.delete_first())
70 print(dq.first())
71 print(dq.last())
72 print(dq.is_empty())
73 print(dq.len())
```

1:1 CRLF UTF-8 4 spaces Python 3.12 2:04 pm 06/12/2024

DSALGO1-IDB2 main

Project

- DSALGO1-IDB2 Z:\DSALGO1-IDB2
 - DOXX
 - FG_EXAM
 - FINALS
 - .idea
 - LinkedList_Activites
 - .idea
 - __init__.py
 - Activity #1 (Queue).py
 - Activity #1 (Stack).py
 - Activity #2.py
 - CircularQueue.py
 - DoublyLinkedListBase.py
 - LinkedDeque.py
 - LinkedQueue.py
 - LinkedStack.py
 - main.py
 - PositionalList.py
 - TeamProject#1 (2 Stack_Deque).py
 - TermProject#1 (Stack_Queue_Deque).py
 - Trees_Activites
 - .idea
 - BinaryTree.py
 - LinkedBinaryTree.py
 - main.py
 - Tree.py
 - MIDTERMS
 - PDF
 - Castro_Mhar_Activity1.py
 - main.py
 - practice.py
 - README.md
 - External Libraries
 - Scratches and Consoles

TeamProject#1 (2 Stack_Deque).py TermProject#1 (Stack_Queue_Deque).py LinkedStack.py LinkedQueue.py

```
1 from LinkedStack import LinkedStack
2 from LinkedQueue import LinkedQueue
3
4
5 1 usage
6 class LinkedDeque:
7
8     def __init__(self):
9         self.front_queue = LinkedQueue()
10        self.back_stack = LinkedStack()
11
12    def len(self):
13        return len(self.front_queue) + len(self.back_stack)
14
15    def is_empty(self):
16        return self.len() == 0
17
18    1 usage
19    def add_first(self, e):
20        self.front_queue.enqueue(e)
21
22    3 usages
23    def add_last(self, e):
24        self.back_stack.push(e)
25
26    1 usage
27    def delete_first(self):
28        if self.is_empty():
29            raise Exception('Deque is empty')
30
31        if self.front_queue.is_empty():
32            while not self.back_stack.is_empty():
33                self.front_queue.enqueue(self.back_stack.pop())
34
35        return self.front_queue.dequeue()
36
37    1 usage
38    def delete_last(self):
39        if self.is_empty():
40            raise Exception('Deque is empty')
41
42        if self.back_stack.is_empty():
43            while not self.front_queue.is_empty():
44                self.back_stack.push(self.front_queue.dequeue())
45
46        return self.back_stack.pop()
47
48    10 usages (9 dynamic)
49    def first(self):
50        if self.is_empty():
51            raise Exception('Deque is empty')
52
53        if self.front_queue.is_empty():
54            while not self.back_stack.is_empty():
55                self.front_queue.enqueue(self.back_stack.pop())
56
57        return self.front_queue.first()
58
59    5 usages (4 dynamic)
60    def last(self):
61        if self.is_empty():
62            raise Exception('Deque is empty')
63
64        if self.back_stack.is_empty():
65            while not self.front_queue.is_empty():
66                self.back_stack.push(self.front_queue.dequeue())
67
68        return self.back_stack.top()
69
70    dq = LinkedDeque()
71    dq.add_last(1)
72    dq.add_last(2)
73    dq.add_last(3)
74    dq.add_first(0)
75    print(dq.delete_last())
76    print(dq.delete_first())
77    print(dq.first())
78    print(dq.last())
79    print(dq.is_empty())
```

1:1 CRLF UTF-8 4 spaces Python 3.12

Type here to search

2:04 pm 06/12/2024

DSALGO1-IDB2 main

Project

- DSALGO1-IDB2 Z:\DSALGO1-IDB2
 - DOXX
 - FG_EXAM
 - FINALS
 - .idea
 - LinkedList_Activites
 - .idea
 - __init__.py
 - Activity #1 (Queue).py
 - Activity #1 (Stack).py
 - Activity #2.py
 - CircularQueue.py
 - DoublyLinkedListBase.py
 - LinkedDeque.py
 - LinkedQueue.py
 - LinkedStack.py
 - main.py
 - PositionalList.py
 - TeamProject#1 (2 Stack_Deque).py
 - TermProject#1 (Stack_Queue_Deque).py
 - Trees_Activites
 - .idea
 - BinaryTree.py
 - LinkedBinaryTree.py
 - main.py
 - Tree.py
 - MIDTERMS
 - PDF
 - Castro_Mhar_Activity1.py
 - main.py
 - practice.py
 - README.md
 - External Libraries
 - Scratches and Consoles

TeamProject#1 (2 Stack_Deque).py TermProject#1 (Stack_Queue_Deque).py LinkedStack.py LinkedQueue.py

```
35 raise Exception('Deque is empty')
36
37 if self.back_stack.is_empty():
38     while not self.front_queue.is_empty():
39         self.back_stack.push(self.front_queue.dequeue())
40
41 return self.back_stack.pop()
42
43 10 usages (9 dynamic)
44 def first(self):
45     if self.is_empty():
46         raise Exception('Deque is empty')
47
48     if self.front_queue.is_empty():
49         while not self.back_stack.is_empty():
50             self.front_queue.enqueue(self.back_stack.pop())
51
52     return self.front_queue.first()
53
54 5 usages (4 dynamic)
55 def last(self):
56     if self.is_empty():
57         raise Exception('Deque is empty')
58
59     if self.back_stack.is_empty():
60         while not self.front_queue.is_empty():
61             self.back_stack.push(self.front_queue.dequeue())
62
63     return self.back_stack.top()
64
65 dq = LinkedDeque()
66 dq.add_last(1)
67 dq.add_last(2)
68 dq.add_last(3)
69 dq.add_first(0)
70 print(dq.delete_last())
71 print(dq.delete_first())
72 print(dq.first())
73 print(dq.last())
74 print(dq.is_empty())
```

1:1 CRLF UTF-8 4 spaces Python 3.12

Type here to search

2:04 pm 06/12/2024

OUTPUT:

```
"C:\Program Files\Python312\python.exe" "Z:\DSAL601-IDB2\FINALS\LinkedList_Activites\TeamProject#1 (2 Stack_Deque).py"  
3  
0  
1  
2  
False  
2  
  
Process finished with exit code 0
```