

1.

Code:

```
main.py ×  
  
1 usage  
1 class Stack:  
2     def __init__(self):  
3         self.item = []  
4  
5     4 usages  
6     def is_empty(self):  
7         return len(self.item) == 0  
8  
9     7 usages  
10    def push(self, item):  
11        self.item.append(item)  
12        print(f"Pushed item:", item)  
13  
14    5 usages  
15    def pop(self):  
16        if self.is_empty():  
17            raise IndexError("Cannot pop from an empty stack")  
18        return self.item.pop()  
19  
20    1 usage  
21    def peek(self):  
22        if self.is_empty():  
23            raise IndexError("Stack is empty")  
24        return self.item[-1]  
25  
26    2 usages  
27    def len(self):  
28        return len(self.item)
```

```

26 S = Stack()
27
28 S.push(5)
29 S.push(3)
30 print("The current top of Stack: ", S.len())
31 print("Popped item: ", S.pop())
32 print("Is the stack empty? ", S.is_empty())
33 print("Popped item: ", S.pop())
34 print("Is the stack empty? ", S.is_empty())
35 try:
36     S.pop()
37 except IndexError as e:
38     print(e)
39 S.push(7)
40 S.push(9)
41 print("Top item: ", S.peek())
42 S.push(4)
43 print("The current top of Stack: ", S.len())
44 S.pop()
45 S.push(6)
46 S.push(8)
47 S.pop()

```

Output:

```

Z:\Midterms\.venv\Scripts\python.exe Z:\Midterms\main.py
Pushed item: 5
Pushed item: 3
The current top of Stack: 2
Popped item: 3
Is the stack empty? False
Popped item: 5
Is the stack empty? True
Cannot pop from an empty stack
Pushed item: 7
Pushed item: 9
Top item: 9
Pushed item: 4
The current top of Stack: 3
Pushed item: 6
Pushed item: 8

Process finished with exit code 0

```

2.

Code:

```
51 class Stack:
52     def __init__(self):
53         self.item = []
54
55     2 usages
56     def is_empty(self):
57         return len(self.item) == 0
58     9 usages
59     def push(self, item):
60         self.item.append(item)
61         print(f"Pushed item:", item)
62
63     8 usages
64     def pop(self):
65         if self.is_empty():
66             raise IndexError("Cannot pop from an empty stack")
67         return self.item.pop()
68
69     def peek(self):
70         if self.is_empty():
71             raise IndexError("Stack is empty")
72         return self.item[-1]
73
74     def len(self):
75         return len(self.item)
```

```

73
74     S = Stack()
75
76     S.push(5)
77     S.push(3)
78     print("Popped: ", S.pop())
79     S.push(2)
80     S.push(8)
81     print("Popped: ", S.pop())
82     print("Popped: ", S.pop())
83     S.push(9)
84     S.push(1)
85     print("Popped: ", S.pop())
86     S.push(7)
87     S.push(6)
88     print("Popped: ", S.pop())
89     print("Popped: ", S.pop())
90     S.push(4)
91     print("Popped: ", S.pop())
92     print("Popped: ", S.pop())

```

Output:

```

Z:\Midterms\.venv\Scripts\python.exe Z:\Midterms\main.py
Pushed item: 5
Pushed item: 3
Popped: 3
Pushed item: 2
Pushed item: 8
Popped: 8
Popped: 2
Pushed item: 9
Pushed item: 1
Popped: 1
Pushed item: 7
Pushed item: 6
Popped: 6
Popped: 7
Pushed item: 4
Popped: 4
Popped: 9

```

Process finished with exit code 0