Activity #1 (Midterm) Laboratory Activity

Create an implementation of the **Stack Data Structure:**

As a guide, you can use the following description of the **Stack Data Structure**.

Stacks are the simplest of all data structures, yet they are also among the most important. They are used in a host of different applications, and as a tool for many more sophisticated data structures and algorithms. Formally, a stack is an abstract data type (ADT) such that an instance S supports the following two methods:

S.push(e): Add element e to the top of stack S.

S.pop( ): Remove and return the top element from the stack S; an error occurs if the stack is empty.

Additionally, let us define the following accessor methods for convenience:

S.top( ): Return a reference to the top element of stack S, without removing it; an error occurs if the stack is empty.

S.is_empty( ): Return True if stack S does not contain any elements.

len(S): Return the number of elements in stack S; in Python, we implement this with the special method __len__.

By convention, we assume that a newly created stack is empty, and that there is no a priori bound on the capacity of the stack. Elements added to the stack can have arbitrary type.

Next, simulate the **Stack Data Structure** using the table below:

Operation
S.push(5)
S.push(3)
len(S)
S.pop( )
S.is_empty( )
S.pop( )
S.is_empty( )
S.pop( )
S.push(7)
S.push(9)
S.top( )
S.push(4)
len(S)
S.pop( )
S.push(6)
S.push(8)
S.pop( )

Lastly, simulate the following operations to answer the question listed.

What values are returned during the following series of stack operations, if executed upon an initially empty stack? **push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop()**.

CODE:

Activity1Midterms_Jacobivan.py

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

    def push(self, itemfromOBJ):
        self.items.append(itemfromOBJ)
        print(f"Pushed element {itemfromOBJ}; After the push: {self.items}")

    def pop(self):
        if not len(self.items) == 0:
            popped = self.items.pop()
            print(f"Popped element: [{popped}]; After the pop: {self.items}")
            print()
        else:
            print("Stack is empty [:(] Can no longer pop")
            print()

    def top(self):
        if not self.is_empty():
            print(f"Rightmost/Top element: [{self.items[-1]}]")
        else:
            print("Stack is empty")

    def len(self):
        print()
        print(f"Current stack length: {len(self.items)}")

    def is_empty(self):
        walanglaman = len(self.items) == 0
        print(f"Is the stack empty? {walanglaman}")
        return walanglaman

S = Stack()
print("Simulation 1")
```

Stack › pop()

Activity1Midterms_Jacobivan.py

```python
print("Simulation 1")
S.push(5)
S.push(3)
S.len()
S.pop()
S.is_empty()
S.pop()
S.is_empty()
S.pop()
S.push(7)
S.push(9)
S.top()
S.push(4)
S.len()
S.pop()
S.push(6)
S.push(8)
S.pop()
print()
print("Simulation 2")
X = Stack()
X.push(5)
X.push(3)
X.pop()
X.push(2)
X.push(8)
X.pop()
X.pop()
X.push(9)
X.push(1)
X.pop()
X.push(7)
X.push(6)
X.pop()
X.pop()
X.push(4)
X.pop()
X.pop()
```

Stack › pop()

OUTPUT:

Run Activity1Midterms_Jacobivan

```
Z:\DSALG01-IDB2\MIDTERMS\.venv\Scripts\python.exe Z:\DSALG01-IDB2\MIDTERMS\Activity1Midterms_JacobIvan.py
Simulation 1
Pushed element 5; After the push: [5]
Pushed element 3; After the push: [5, 3]

Current stack length: 2
Popped element: [3]; After the pop: [5]

Is the stack empty? False
Popped element: [5]; After the pop: []

Is the stack empty? True
Stack is empty [:(] Can no longer pop

Pushed element 7; After the push: [7]
Pushed element 9; After the push: [7, 9]
Is the stack empty? False
Rightmost/Top element: [9]
Pushed element 4; After the push: [7, 9, 4]

Current stack length: 3
Popped element: [4]; After the pop: [7, 9]

Pushed element 6; After the push: [7, 9, 6]
Pushed element 8; After the push: [7, 9, 6, 8]
Popped element: [8]; After the pop: [7, 9, 6]


Simulation 2
Pushed element 5; After the push: [5]
Pushed element 3; After the push: [5, 3]
Popped element: [3]; After the pop: [5]

Pushed element 2; After the push: [5, 2]
Pushed element 8; After the push: [5, 2, 8]
Popped element: [8]; After the pop: [5, 2]

Popped element: [2]; After the pop: [5]
```

```
Popped element: [8]; After the pop: [5, 2]

Popped element: [2]; After the pop: [5]

Pushed element 9; After the push: [5, 9]
Pushed element 1; After the push: [5, 9, 1]
Popped element: [1]; After the pop: [5, 9]

Pushed element 7; After the push: [5, 9, 7]
Pushed element 6; After the push: [5, 9, 7, 6]
Popped element: [6]; After the pop: [5, 9, 7]

Popped element: [7]; After the pop: [5, 9]

Pushed element 4; After the push: [5, 9, 4]
Popped element: [4]; After the pop: [5, 9]

Popped element: [9]; After the pop: [5]


Process finished with exit code 0
```