Practical No.2

Data Structures

-Abin Pillai S103

AIM: Stack with insertion, deletion, traversal operations

(Code Using Command Line)

```
import os
import time
from termcolor import colored, cprint
class Stack:
    def __init__(self):
        self.items = []
    def is_empty(self):
        return len(self.items) == 0
    def insert(self, item, position):
        if position < 0 or position > len(self.items):
            raise IndexError("Invalid position")
        self.items.insert(position, item)
        print(colored(f"'{item}' has been inserted at position {position}.", "green"))
        self.animate_insert(item)
    def delete(self, position):
        if position < 0 or position >= len(self.items):
            raise IndexError("Invalid position")
        item = self.items.pop(position)
        print(colored(f"'{item}' has been deleted from position {position}.", "red"))
        self.animate_delete(item)
        return item
    def peek(self):
        if self.is_empty():
```

Practical No.2
Data Structures

-Abin Pillai S103

```
            raise IndexError("Peek from an empty stack")
        return self.items[-1]
    def size(self):
        return len(self.items)
    def traverse(self):
        if self.is_empty():
            raise IndexError("Cannot traverse an empty stack")
        return " <- ".join(self.items)
    def __str__(self):
        return " <- ".join(reversed(self.items)) if self.items else "Stack is empty"
    def animate_insert(self, item):
        for _ in range(3):
            print(colored(f"Inserting {item}...", "yellow"))
            time.sleep(0.3)
            self.clear_screen()
    def animate_delete(self, item):
        for _ in range(3):
            print(colored(f"Deleting {item}...", "magenta"))
            time.sleep(0.3)
            self.clear_screen()
    @staticmethod
    def clear_screen():
        os.system('cls' if os.name == 'nt' else 'clear')
def stack_operations():
    stack = Stack()
```

Practical No.2
Data Structures

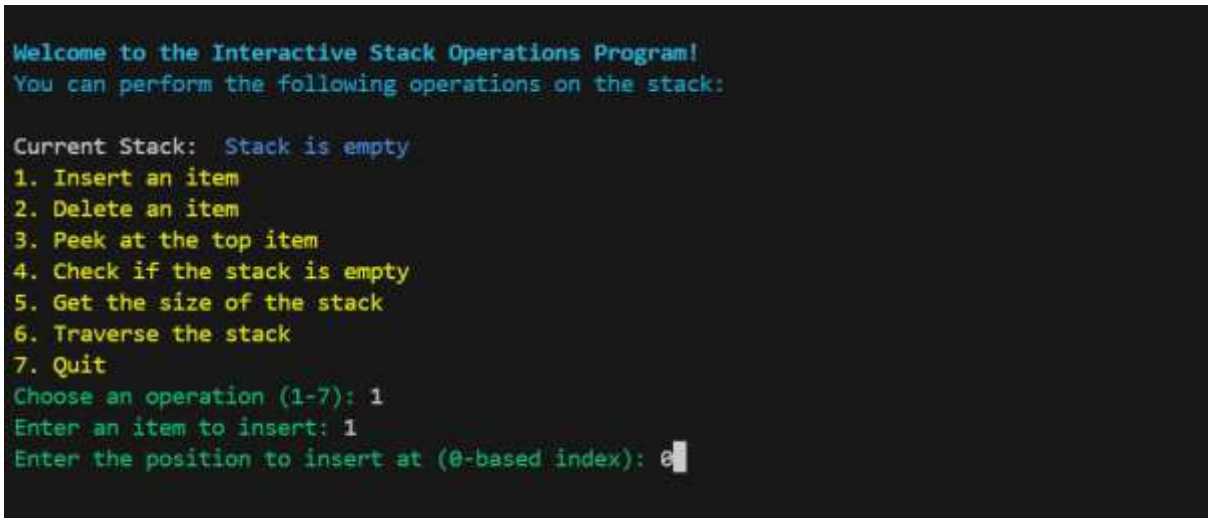
-Abin Pillai S103

```
    cprint("Welcome to the Interactive Stack Operations Program!", "cyan", attrs=["bold"])
    cprint("You can perform the following operations on the stack:", "cyan")
    while True:
        print("\nCurrent Stack: ", colored(str(stack), "blue"))
        print(colored("1. Insert an item", "yellow"))
        print(colored("2. Delete an item", "yellow"))
        print(colored("3. Peek at the top item", "yellow"))
        print(colored("4. Check if the stack is empty", "yellow"))
        print(colored("5. Get the size of the stack", "yellow"))
        print(colored("6. Traverse the stack", "yellow"))
        print(colored("7. Quit", "yellow"))
        try:
            choice = int(input(colored("Choose an operation (1-7): ", "green")))
        except ValueError:
            cprint("Invalid input. Please enter a number between 1 and 7.", "red")
            continue
        if choice == 1:
            item = input(colored("Enter an item to insert: ", "green"))
            try:
                position = int(input(colored("Enter the position to insert at (0-based index): ", "green")))
                stack.insert(item, position)
            except ValueError:
                cprint("Invalid input. Position must be an integer.", "red")
            except IndexError as e:
```

```
            cprint(e, "red")
        elif choice == 2:
            try:
                position = int(input(colored("Enter the position to delete from (0-based index): ", "green")))
                stack.delete(position)
            except ValueError:
                cprint("Invalid input. Position must be an integer.", "red")
            except IndexError as e:
                cprint(e, "red")
        elif choice == 3:
            try:
                cprint("Top item: " + stack.peek(), "blue")
            except IndexError as e:
                cprint(e, "red")
        elif choice == 4:
            cprint("Is the stack empty? " + ("Yes" if stack.is_empty() else "No"), "blue")
        elif choice == 5:
            cprint("Size of the stack: " + str(stack.size()), "blue")
        elif choice == 6:
            try:
                cprint("Stack contents: " + stack.traverse(), "blue")
            except IndexError as e:
                cprint(e, "red")
        elif choice == 7:
```

```
            cprint("Exiting the program. Goodbye!", "cyan", attrs=["bold"])
            break
        else:
            cprint("Invalid choice. Please select a number between 1 and 7.", "red")
# Call the function to start stack operations
if __name__ == "__main__":
    stack_operations()
```

Ouput:



Practical No.2

Data Structures

-Abin Pillai S103

```
Current Stack:  3 <- 1
1. Insert an item
2. Delete an item
3. Peek at the top item
4. Check if the stack is empty
5. Get the size of the stack
6. Traverse the stack
7. Quit
Choose an operation (1-7): 3
Top item: 3

Current Stack:  3 <- 1
1. Insert an item
2. Delete an item
3. Peek at the top item
4. Check if the stack is empty
5. Get the size of the stack
6. Traverse the stack
7. Quit
Choose an operation (1-7): 4
Is the stack empty? No
```

```
Current Stack:  3 <- 1
1. Insert an item
2. Delete an item
3. Peek at the top item
4. Check if the stack is empty
5. Get the size of the stack
6. Traverse the stack
7. Quit
Choose an operation (1-7): 5
Size of the stack: 2

Current Stack:  3 <- 1
1. Insert an item
2. Delete an item
3. Peek at the top item
4. Check if the stack is empty
5. Get the size of the stack
6. Traverse the stack
7. Quit
Choose an operation (1-7): 6
Stack contents: 1 <- 3
```