Practical No.1

Data Structures

-Abin Pillai S103

AIM: Write a program to implement Abstract Data Types (ADT)

(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 push(self, item):
        self.items.append(item)
        print(colored(f"'{item}' has been pushed onto the stack.", "green"))
        self.animate_push(item)
    def pop(self):
        if self.is_empty():
            raise IndexError("Pop from an empty stack")
        item = self.items.pop()
        print(colored(f"'{item}' has been popped from the stack.", "red"))
        self.animate_pop(item)
        return item
    def peek(self):
        if self.is_empty():
            raise IndexError("Peek from an empty stack")
        return self.items[-1]
```

Practical No.1

Data Structures

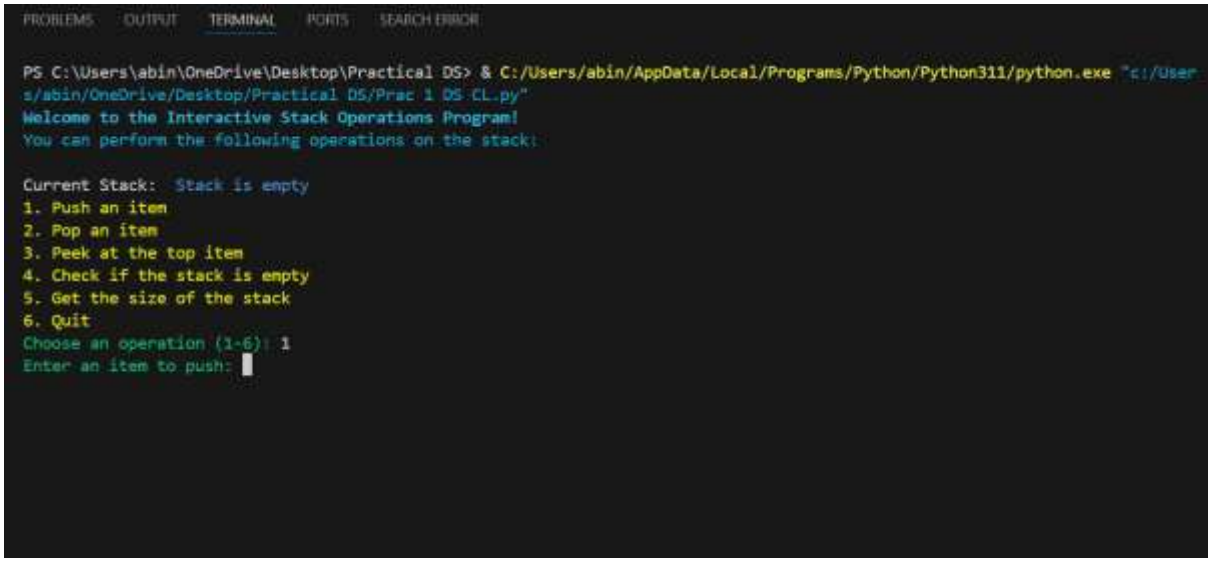
-Abin Pillai S103

```
    def size(self):
        return len(self.items)
    def __str__(self):
        return " <- ".join(reversed(self.items)) if self.items else "Stack is empty"
    def animate_push(self, item):
        for _ in range(3):
            print(colored(f"Pushing {item}...", "yellow"))
            time.sleep(0.3)
            self.clear_screen()
    def animate_pop(self, item):
        for _ in range(3):
            print(colored(f"Popping {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()
    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. Push an item", "yellow"))
```

```
print(colored("2. Pop 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. Quit", "yellow"))
try:
    choice = int(input(colored("Choose an operation (1-6): ", "green")))
except ValueError:
    cprint("Invalid input. Please enter a number between 1 and 6.", "red")
    continue
if choice == 1:
    item = input(colored("Enter an item to push: ", "green"))
    stack.push(item)
elif choice == 2:
    try:
        stack.pop()
    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:
        cprint("Exiting the program. Goodbye!", "cyan", attrs=["bold"])
        break
    else:
        cprint("Invalid choice. Please select a number between 1 and 6.", "red")
# Call the function to start stack operations
if __name__ == "__main__":
  stack_operations()
```

Output:



Practical No.1

Data Structures

-Abin Pillai S103

```
5. Get the size of the stack
6. Quit
Choose an operation (1-6): 3
Top item: 3

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

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

```
Current Stack:  3 <- 2 <- 1
1. Push an item
2. Pop an item
3. Peek at the top item
4. Check if the stack is empty
5. Get the size of the stack
6. Quit
Choose an operation (1-6): 6
Exiting the program. Goodbye!
```