class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.next = None

class Stack:

    def \_\_init\_\_(self):

        self.count = 0

        self.top = None

    def push(self,data):

        new\_node = Node(data)

        new\_node.next = self.top

        self.top = new\_node

        self.count += 1

        print(f"push the value to stack: {data}")

    def pop(self) -> int:

        # Stack is empty

        if self.top == None:

            print("Stack is empty can't perform pop operation")

            return -100

        data = self.top.data

        self.top = self.top.next

        self.count -= 1

        print(f"Popped item from stack: {data}")

        return data

    def peek(self) -> int:

        # Stack is empty

        if self.top == None:

            print("Stack is empty")

            return -100

        print(f"Top data: {self.top.data}")

        return self.top.data

    def get\_count(self) -> int:

        print(f"there are {self.count} items in stack")

        return self.count

    def print\_all\_values(self):

        # Stack is empty

        if self.top == None:

            print("Stack is empty ")

            return

        print("Elements in Stack")

        current\_node = self.top

        while current\_node is not None:

            print(f" {current\_node.data}")

            current\_node = current\_node.next

def Stack\_test\_code(stack: Stack):

    stack.pop()

    stack.get\_count()

    stack.peek()

    stack.print\_all\_values()

    stack.push(1)

    stack.get\_count()

    stack.peek()

    stack.print\_all\_values()

    stack.pop()

    stack.push(1)

    stack.push(2)

    stack.push(3)

    stack.push(4)

    stack.push(5)

    stack.print\_all\_values()

    stack.pop()

    stack.peek()

    stack.print\_all\_values()

    stack.get\_count()