IT314 - Software Engineering
Lab 5 - Static Analysis

Name : Parth Prajapati

ID : 202001177

Repository Link used for static analysis :

https://github.com/geekcomputers/Python

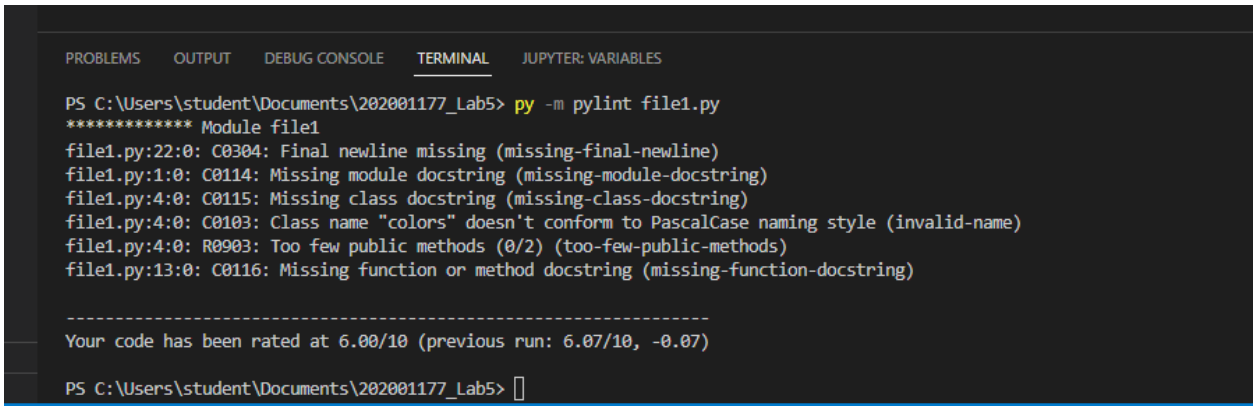
Tool Used for Static Analysis :

- ***Pylint***

Code file 1:

```
import sys



class colors:
    CYAN = "\033[36m"
    GREEN = "\033[32m"
    YELLOW = "\033[33m"
    BLUE = "\033[34m"
    RED = "\033[31m"
    ENDC = "\033[0m"


def printc(color, message):
    print(color + message + colors.ENDC)


# color which we print or import
printc(colors.CYAN, sys.argv[1])
printc(colors.GREEN, sys.argv[1])
printc(colors.YELLOW, sys.argv[1])
printc(colors.BLUE, sys.argv[1])
printc(colors.RED, sys.argv[1])
```

Static Analysis:



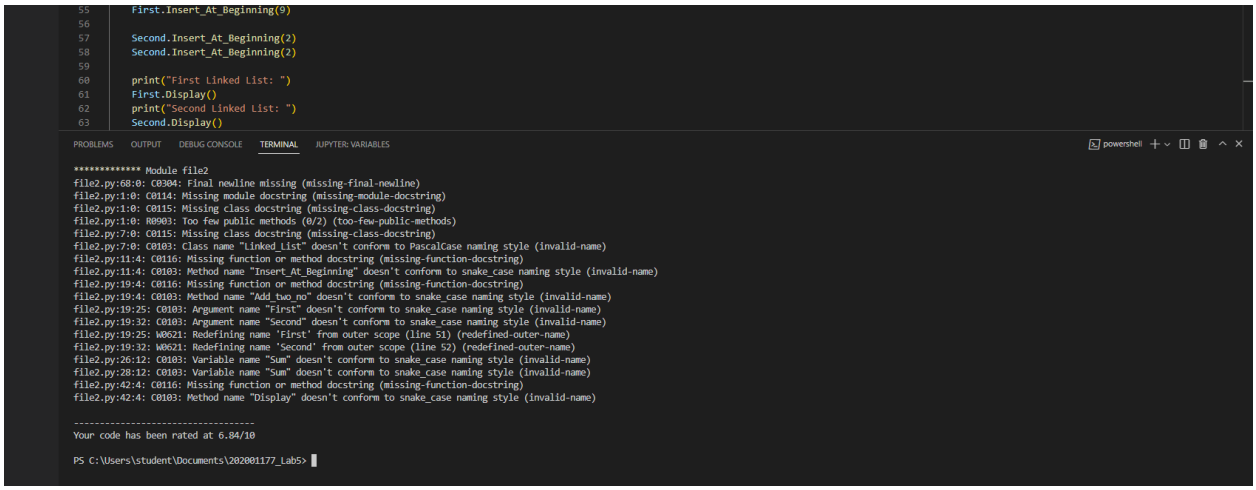
Code File 2:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


class Linked_List:
    def __init__(self):
        self.head = None

    def Insert_At_Beginning(self, new_data):
        new_node = Node(new_data)
        if self.head is None:
            self.head = new_node
            return
        new_node.next = self.head
        self.head = new_node

    def Add_two_no(self, First, Second):
        prev = None
        temp = None
```

```
        carry = 0
        while First is not None or Second is not None:
            first_data = 0 if First is None else First.data
            second_data = 0 if Second is None else Second.data
            Sum = carry + first_data + second_data
            carry = 1 if Sum >= 10 else 0
            Sum = Sum if Sum < 10 else Sum % 10
            temp = Node(Sum)
            if self.head is None:
                self.head = temp
            else:
                prev.next = temp
            prev = temp
            if First is not None:
                First = First.next
            if Second is not None:
                Second = Second.next
        if carry > 0:
            temp.next = Node(carry)

    def Display(self):
        temp = self.head
        while temp:
            print(temp.data, "->", end=" ")
            temp = temp.next
        print("None")


if __name__ == "__main__":
    First = Linked_List()
    Second = Linked_List()
    First.Insert_At_Beginning(6)
    First.Insert_At_Beginning(4)
    First.Insert_At_Beginning(9)

    Second.Insert_At_Beginning(2)
    Second.Insert_At_Beginning(2)

    print("First Linked List: ")
    First.Display()
```

```
print("Second Linked List: ")
Second.Display()

Result = Linked_List()
Result.Add_two_no(First.head, Second.head)
print("Final Result: ")
Result.Display()
```

Static Analysis :

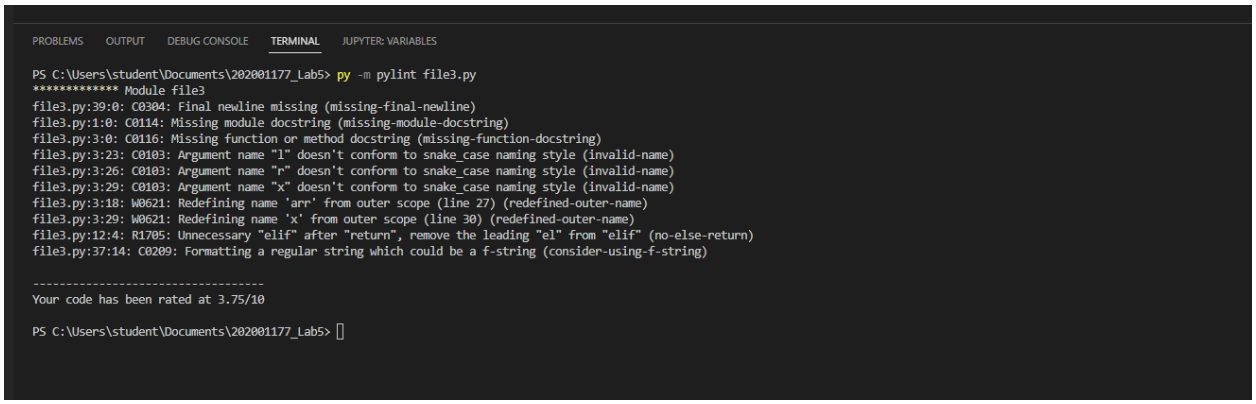


Code File 3:

```
# It returns location of x in given array arr
# if present, else returns -1
def binary_search(arr, l, r, x):
    # Base case: if left index is greater than right index, element is not
present
    if l > r:
        return -1

    # Calculate the mid index
    mid = (l + r) // 2

    # If element is present at the middle itself
    if arr[mid] == x:
        return mid

    # If element is smaller than mid, then it can only be present in left
subarray
    elif arr[mid] > x:
        return binary_search(arr, l, mid - 1, x)

    # Else the element can only be present in right subarray
    else:
        return binary_search(arr, mid + 1, r, x)


# Main Function
if __name__ == "__main__":
    # User input array
    arr = [int(x) for x in input("Enter the array with elements separated
by commas: ").split(",")]

    # User input element to search for
    x = int(input("Enter the element you want to search for: "))

    # Function call
    result = binary_search(arr, 0, len(arr) - 1, x)
```

```
# printing the output
if result != -1:
    print("Element is present at index {}".format(result))
else:
    print("Element is not present in array")
```

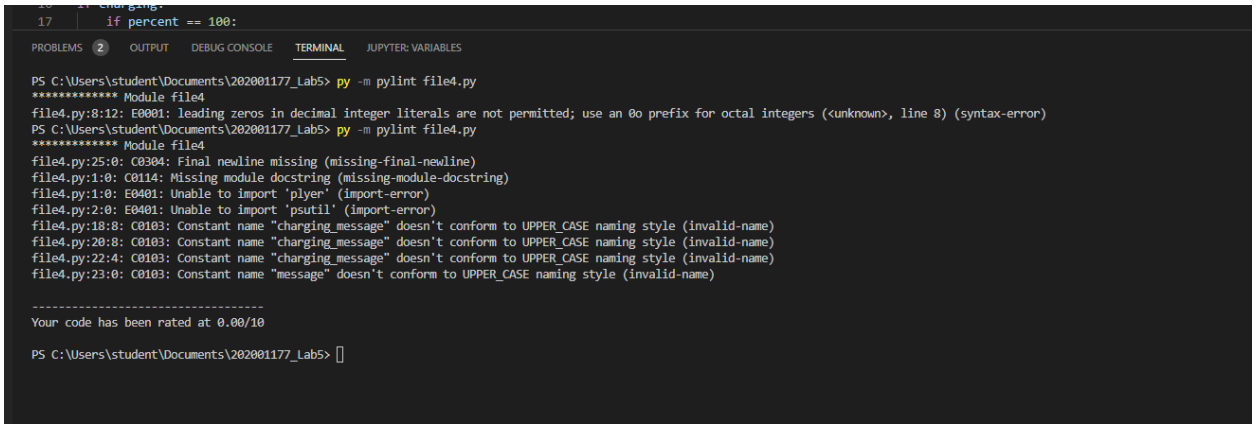
Static Analysis :



Code File 4:

```
from plyer import notification  # pip install plyer
import psutil  # pip install psutil

# psutil.sensors_battery() will return the information related to battery
battery = psutil.sensors_battery()

# battery percent will return the current battery prcentage
percent = battery.percent
charging = (
    battery.power_plugged
)

# Notification(title, description, duration)--to send
# notification to desktop
# help(Notification)
if charging:
    if percent == 100:
        charging_message = "Unplug your Charger"
    else:
        charging_message = "Charging"
else:
    charging_message = "Not Charging"
message = str(percent) + "% Charged\n" + charging_message

notification.notify("Battery Information", message, timeout=10)
```

Static Analysis :

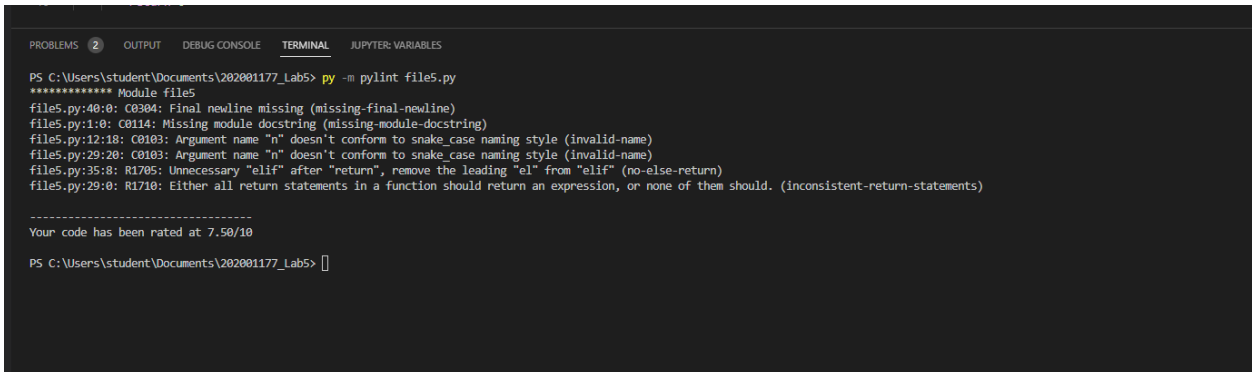


Code file 5:

```
def is_square_free(factors):
    """
    This functions takes a list of prime factors as input.
    returns True if the factors are square free.
    """
    for i in factors:
        if factors.count(i) > 1:
            return False
    return True


def prime_factors(n):
    """
    Returns prime factors of n as a list.
    """
    i = 2
    factors = []
    while i * i <= n:
        if n % i:
            i += 1
        else:
            n //= i
            factors.append(i)
    if n > 1:
        factors.append(n)
    return factors


def mobius_function(n):
    """
    Defines Mobius function
    """
    factors = prime_factors(n)
    if is_square_free(factors):
        if len(factors) % 2 == 0:
            return 1
        elif len(factors) % 2 != 0:
```

```
            return -1
    else:
        return 0
```

Static Analysis :



Analysis of Errors :

S.No	Message Object	Expansion	Explanation
1.	C	Convention	It is displayed when the program is not following the standard rules.
2.	R	Refactor	It is displayed for bad code smell
3.	W	Warning	It is displayed for python specific problems
4.	E	Error	It is displayed when that particular line execution results some error
5.	F	Fatal	It is displayed when pylint has no access to further process that line.