Assignment-8

Ht. No: 2303A51923

Name: V.Sravani

Batch: 23

Task Description #1 (Username Validator – Apply AI in Authentication Context)

• Task: Use AI to generate at least 3 assert test cases for a function is_valid_username(username) and then implement the function using Test-Driven Development principles.

• Requirements:

o Username length must be between 5 and 15 characters.

o Must contain only alphabets and digits.

o Must not start with a digit.

o No spaces allowed. Example Assert Test Cases:

assert is_valid_username("User123") ==

True assert is_valid_username("12User")

== False assert is_valid_username("Us er")

== False

Expected Output #1:

• Username validation logic successfully passing all AI-generated test cases.

Task Description #2 (Even–Odd & Type Classification – Apply AI for Robust Input Handling)

- Task: Use AI to generate at least 3 assert test cases for a function classify_value(x) and implement it using conditional logic and loops.
- Requirements:

o If input is an integer, classify as "Even" or "Odd".

o If input is 0, return "Zero".

o If input is non-numeric, return "Invalid Input".

Example Assert Test Cases:

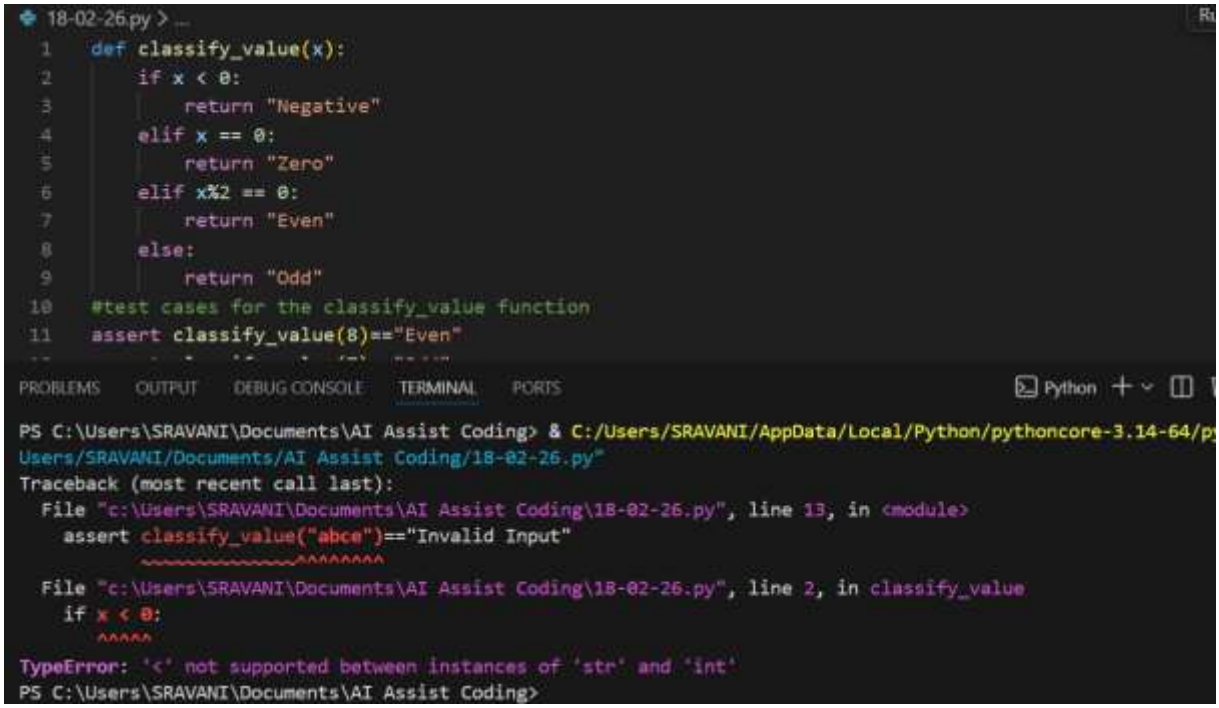
assert classify_value(8) == "Even" assert

classify_value(7) == "Odd" assert

classify_value("abc") == "Invalid Input"

Expected Output #2:

- Function correctly classifying values and passing all test cases.



Task Description #3 (Palindrome Checker – Apply AI for String Normalization)

- Task: Use AI to generate at least 3 assert test cases for a function is_palindrome(text) and implement the function.
- Requirements:

o Ignore case, spaces, and punctuation. o Handle edge cases such

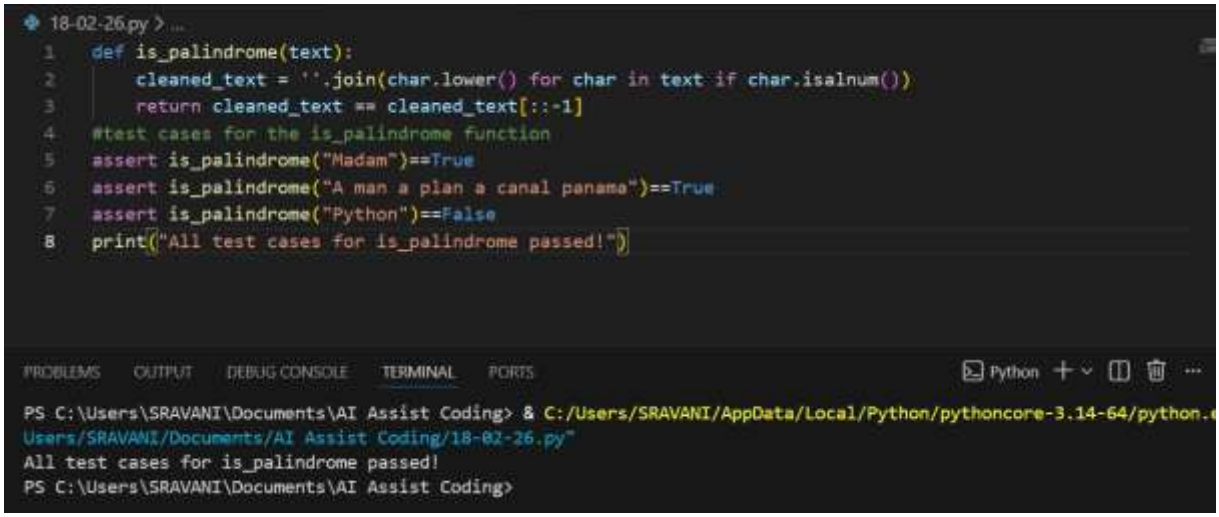
as empty strings and single characters.

Example Assert Test Cases:

assert is_palindrome("Madam") == True assert

is_palindrome("A man a plan a canal Panama") ==True

assert is_palindrome("Python") == False



Task Description #4 (Email ID Validation – Apply AI for Data Validation)

- Task: Use AI to generate at least 3 assert test cases for a function validate_email(email) and implement the function.
- Requirements:

o Must contain @ and . o Must not start or end

with special characters. o Should handle invalid

formats gracefully.

Example Assert Test Cases:

assert validate_email("user@example.com") == True

assert validate_email("userexample.com") == False assert

validate_email("@gmail.com") == False

Expected Output #5:

- Email validation function passing all AI-generated test cases and handling edge cases correctly.

```python
def validate_email(email):
    if "@" not in email or "." not in email:
        return False
    at_index=email.index('@')
    dot_index=email.rindex('.')
    if at_index < 1 or dot_index < at_index + 2 or dot_index >= len(email) - 1:
        return False
    return True
#test cases for the validate_email function
assert validate_email("user@example.com")==True
assert validate_email("userexample.com")==False
assert validate_email("@gmail.com")==False
assert validate_email("user@com")==False
print("All test cases for validate_email passed!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAVANI
ng/18-02-26.py"
All test cases for validate_email passed!
PS C:\Users\SRAVANI\Documents\AI Assist Coding>
```

Task 5 (Perfect Number Checker – Test Case Design)

• Function: Check if a number is a perfect number (sum of divisors = number).

• Test Cases to Design:

o Normal case: 6 → True,

10 → False. o Edge case: 1.

o Negative number case. o

Larger case: 28.

• Requirement: Validate correctness with assertions.

```python
def is_perfect_number(num):
    if num < 1:
        return False
    divisors_sum=sum(i for i in range(1,num) if num % i == 0)
    return divisors_sum == num
#test cases for the is_perfect_number function
assert is_perfect_number(6)==True
assert is_perfect_number(10)==False
assert is_perfect_number(28)==True
assert is_perfect_number(1)==False
print("All test cases for is_perfect_number passed!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/
ng/18-02-26.py"
All test cases for is_perfect_number passed!
PS C:\Users\SRAVANI\Documents\AI Assist Coding>
```

Task 6 (Abundant Number Checker – Test Case Design)

• Function: Check if a number is abundant (sum of divisors >number).

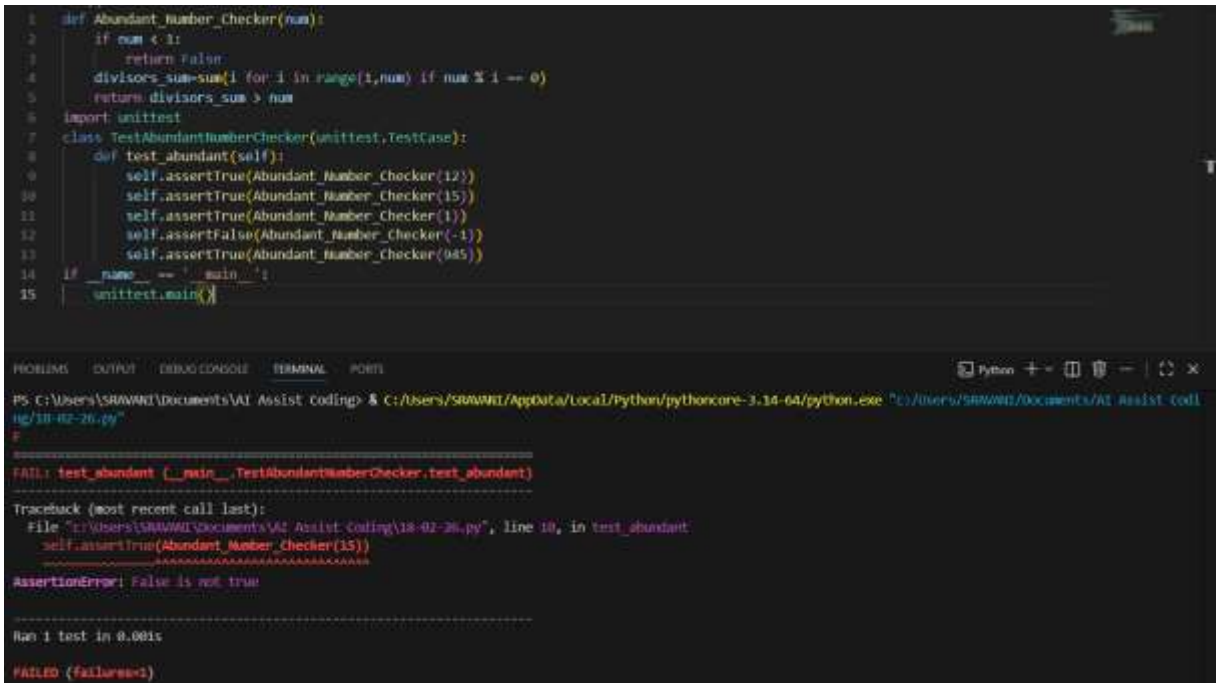
• Test Cases to Design:

o Normal case: 12 → True, 15 → False. o Edge case: 1. o

Negative number case.

o Large case: 945.

Requirement: Validate correctness with unittest



```
def Abundant_Number_Checker(num):
    if num < 1:
        return False
    divisors_sum=sum(i for i in range(1,num) if num % i == 0)
    return divisors_sum > num
import unittest
class TestAbundantNumberChecker(unittest.TestCase):
    def test_abundant(self):
        self.assertTrue(Abundant_Number_Checker(12))
        self.assertTrue(Abundant_Number_Checker(15))
        self.assertTrue(Abundant_Number_Checker(1))
        self.assertFalse(Abundant_Number_Checker(-1))
        self.assertTrue(Abundant_Number_Checker(945))
if __name__ == '__main__':
    unittest.main()
```

```
PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAVANI/Documents/AI Assist Codi
ng/18-02-26.py"
F
======================================================================
FAIL: test_abundant (__main__.TestAbundantNumberChecker.test_abundant)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\SRAVANI\Documents\AI Assist Coding\18-02-26.py", line 10, in test_abundant
    self.assertTrue(Abundant_Number_Checker(15))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: False is not true

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

Task 7 (Deficient Number Checker – Test Case Design)

• Function: Check if a number is deficient (sum of divisors <number).

• Test Cases to Design:

o Normal case: 8 → True,

12 → False. o Edge case: 1.

o Negative number case. o

Large case: 546.

Requirement: Validate correctness with pytest

```python
def deficient_number_checker(num):
    if num < 1:
        return False
    divisors_sum=sum(i for i in range(1,num) if num % i == 0)
    return divisors_sum < num
def test_deficient_number_checker():
    assert deficient_number_checker(8)==True
    assert deficient_number_checker(12)==False
    assert deficient_number_checker(1)==True
    assert deficient_number_checker(546)==False
    print("All test cases for deficient_number_checker passed!")
```

```
PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAVANI/Documents/AI Assist Coding/18-02-26.py"
PS C:\Users\SRAVANI\Documents\AI Assist Coding> python -m pytest 18-02-26.py
========== test session starts ==========
platform win32 -- Python 3.14.2, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\SRAVANI\Documents\AI Assist Coding
collected 1 item

18-02-26.py .                                                    [100%]

========== 1 passed in 0.01s ==========
PS C:\Users\SRAVANI\Documents\AI Assist Coding>
```

Task 8 :

Write a function LeapYearChecker and validate its implementation using 10 pytest test cases

```python
def LeapYearChecker(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    return False
def test_LeapYearChecker():
    assert LeapYearChecker(2020)==True
    assert LeapYearChecker(1900)==False
    assert LeapYearChecker(2000)==True
    assert LeapYearChecker(2021)==False
    print("All test cases for LeapYearChecker passed!")
```

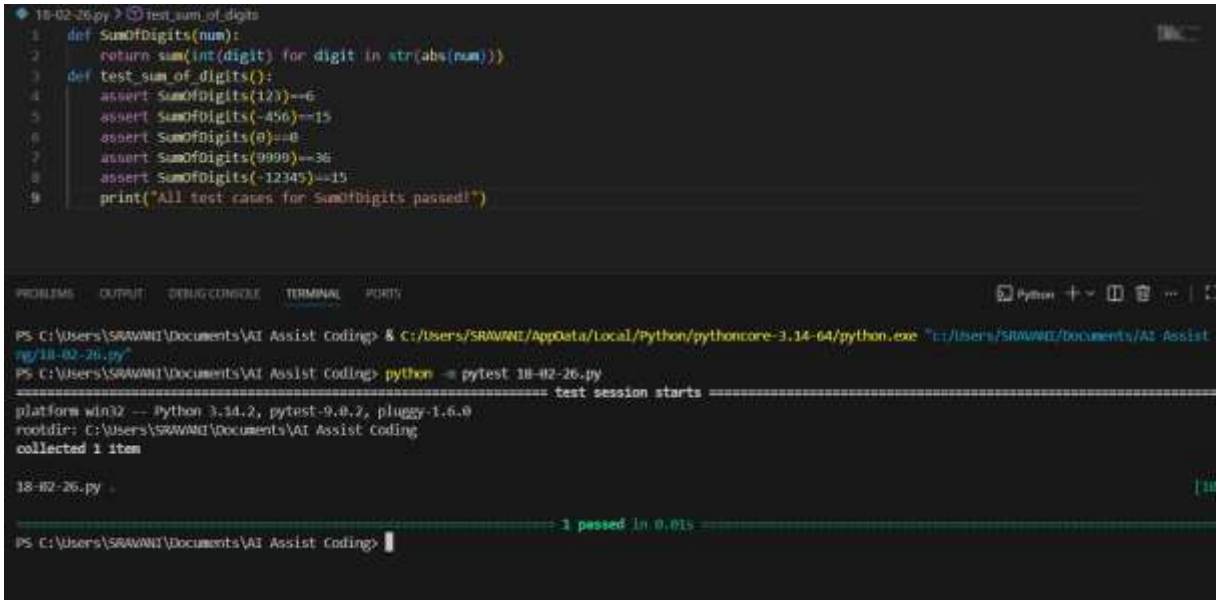
```
PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAVANI/Documents/AI Assist Coding/18-02-26.py"
PS C:\Users\SRAVANI\Documents\AI Assist Coding> python -m pytest 18-02-26.py
========== test session starts ==========
platform win32 -- Python 3.14.2, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\SRAVANI\Documents\AI Assist Coding
collected 1 item

18-02-26.py .                                                    [100%]

========== 1 passed in 0.01s ==========
PS C:\Users\SRAVANI\Documents\AI Assist Coding>
```

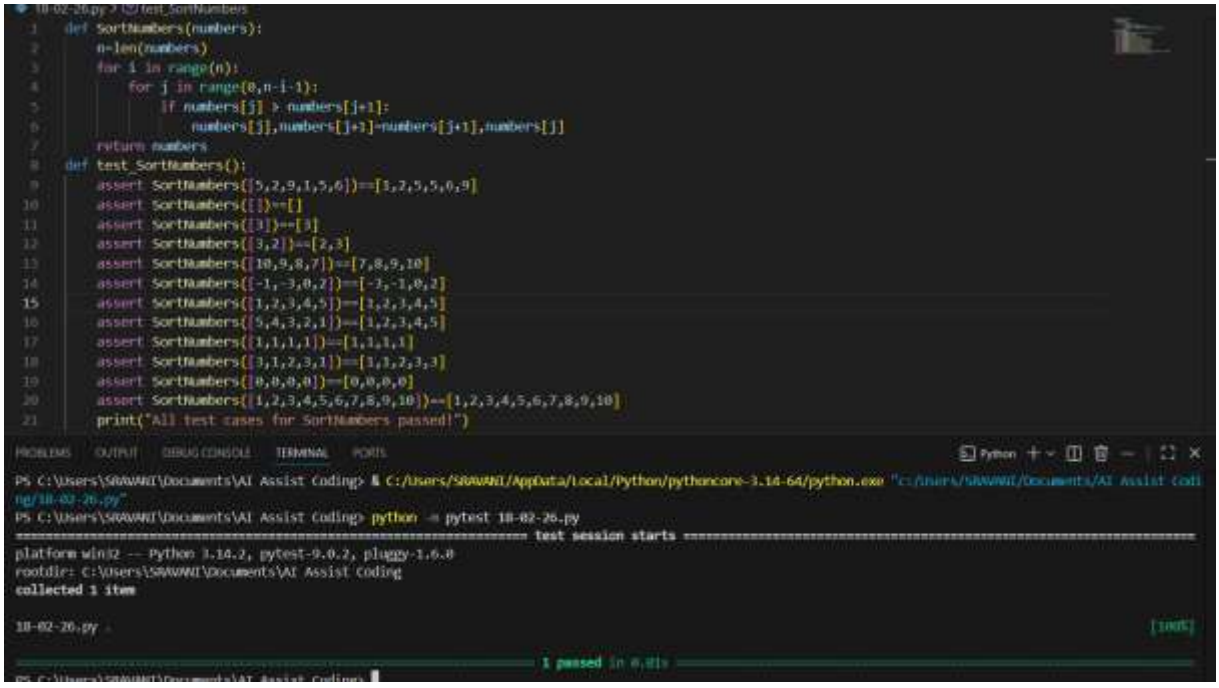
Task 9 :

Write a function SumOfDigits and validate its implementation using 7 pytest test cases.



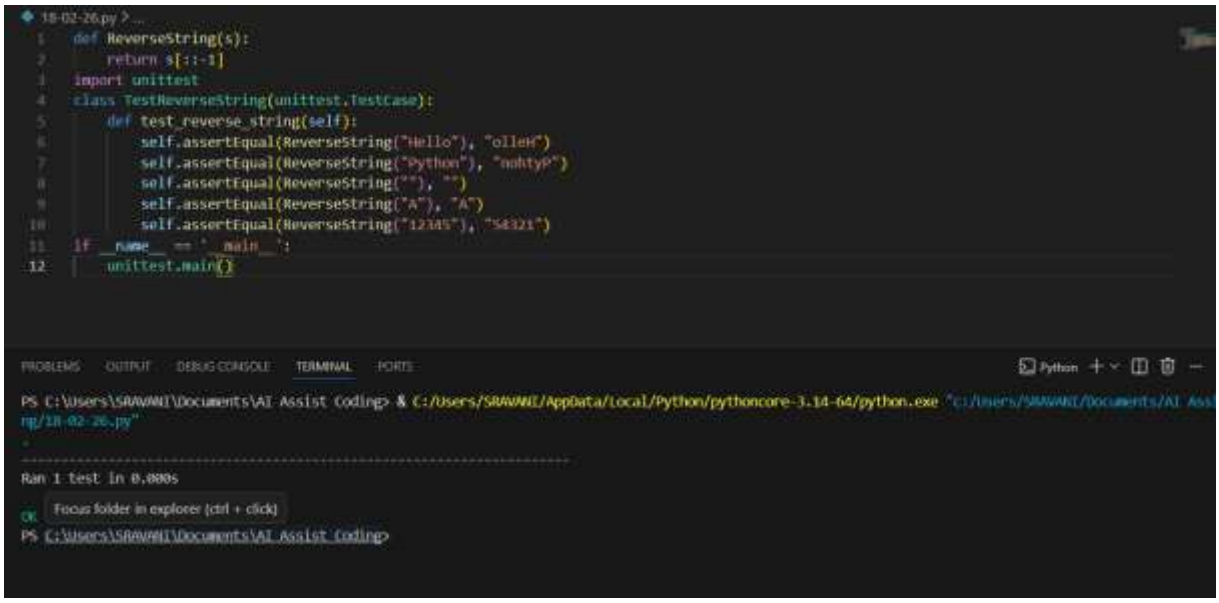
Task 10 :

Write a function SortNumbers (implement bubble sort) and validate its implementation using 25 pytest test cases.



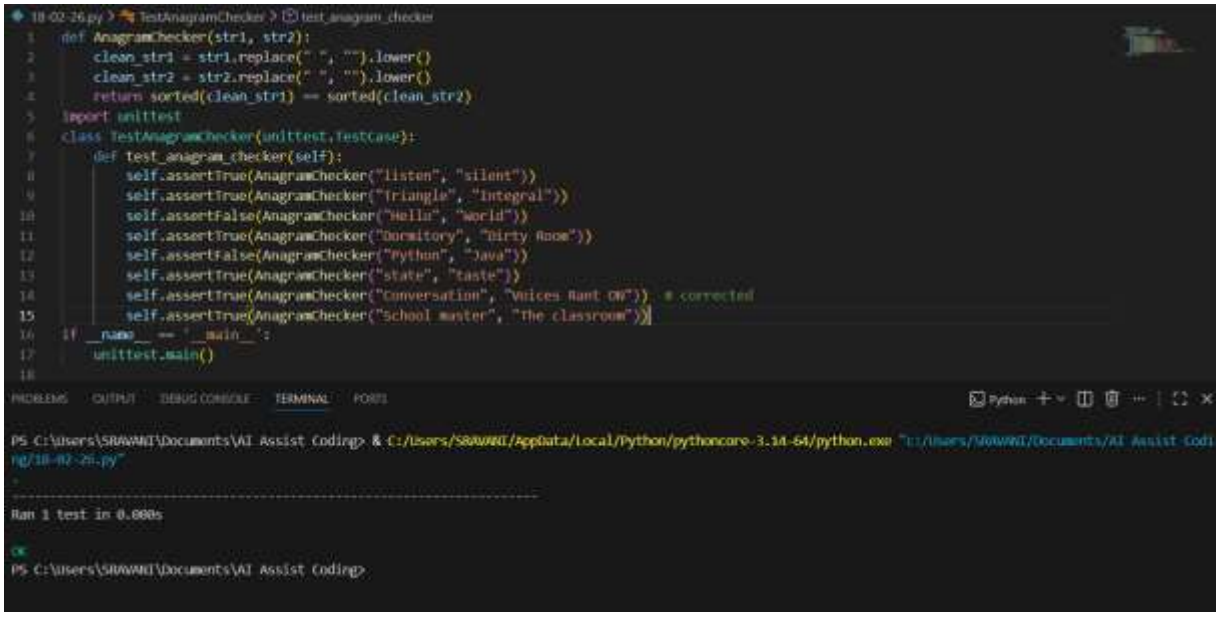
Task 11 :

Write a function ReverseString and validate its implementation using 5 unittest test cases



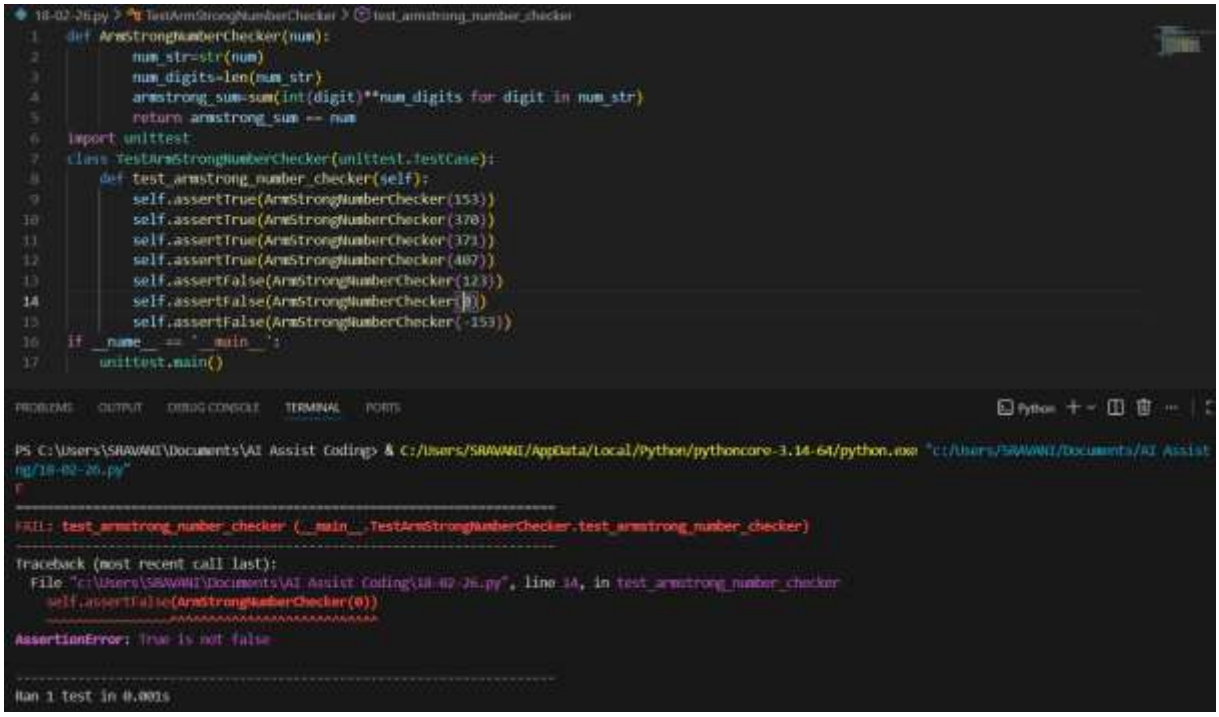
Task 12 :

Write a function AnagramChecker and validate its implementation using 10 unittest test cases.



Task 13 :

Write a function ArmstrongChecker and validate its implementation using 8 unittest test cases.



```
18-02-26.py > TestArmStrongNumberChecker > test_armstrong_number_checker
def ArmStrongNumberChecker(num):
        num_str=str(num)
        num_digits=len(num_str)
        armstrong_sum=sum(int(digit)**num_digits for digit in num_str)
        return armstrong_sum == num
import unittest
class TestArmStrongNumberChecker(unittest.TestCase):
    def test_armstrong_number_checker(self):
        self.assertTrue(ArmStrongNumberChecker(153))
        self.assertTrue(ArmStrongNumberChecker(370))
        self.assertTrue(ArmStrongNumberChecker(371))
        self.assertTrue(ArmStrongNumberChecker(407))
        self.assertFalse(ArmStrongNumberChecker(123))
        self.assertFalse(ArmStrongNumberChecker(0))
        self.assertFalse(ArmStrongNumberChecker(-153))
if __name__ == '__main__':
    unittest.main()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Python

PS C:\Users\SRAVANI\Documents\AI Assist Codings & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAVANI/Documents/AI Assist
ng/18-02-26.py"
F
======================================================================
FAIL: test_armstrong_number_checker (__main__.TestArmStrongNumberChecker.test_armstrong_number_checker)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\SRAVANI\Documents\AI Assist Coding\18-02-26.py", line 14, in test_armstrong_number_checker
    self.assertFalse(ArmStrongNumberChecker(0))
    ~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: True is not false

----------------------------------------------------------------------
Ran 1 test in 0.001s
```