Assignment -8

Name: K. Nithin Kumar
Hall Ticket: 2303A51630

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

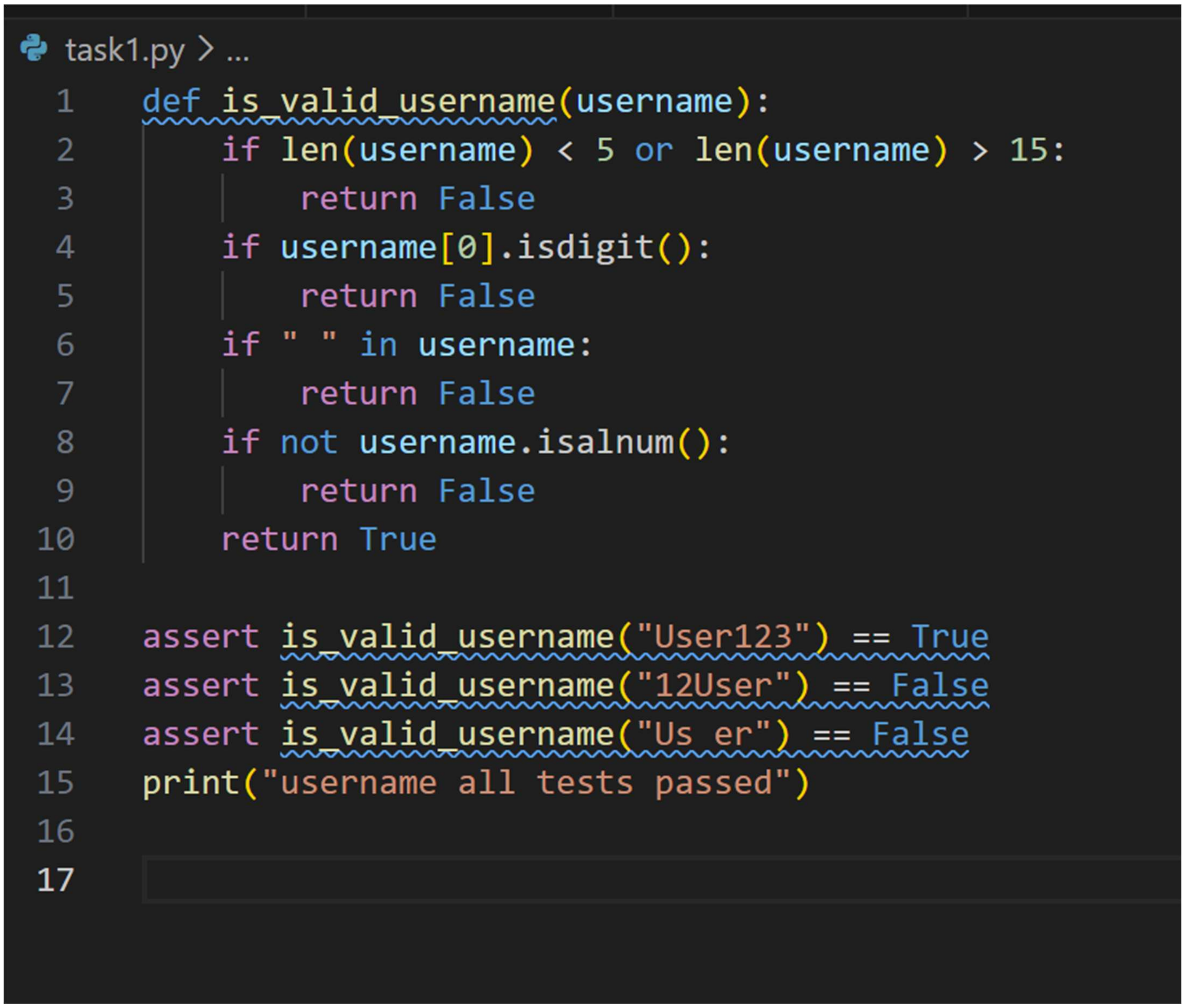
assert is_valid_username("Us er") == False

Expected Output #1:

• Username validation logic successfully passing all AI-

generated test cases.

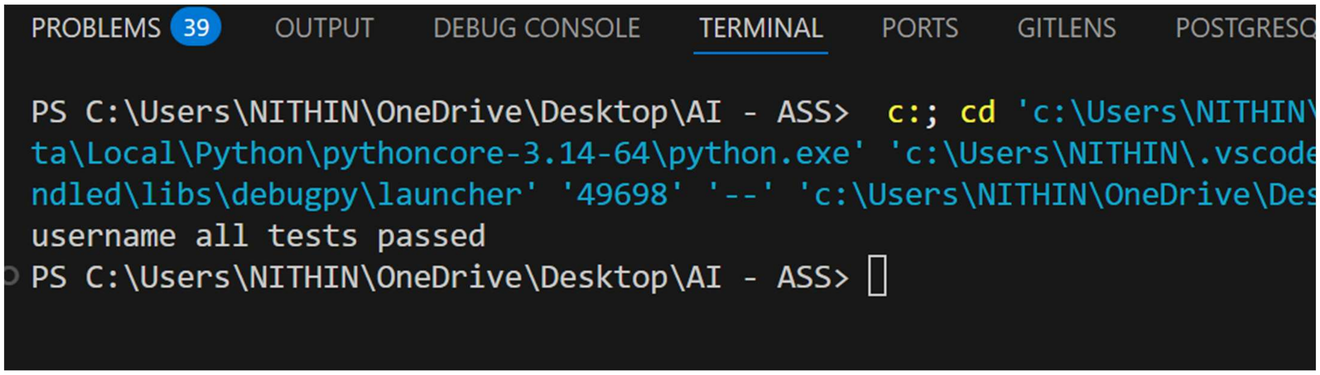
CODE :



task1.py > ...

```python
def is_valid_username(username):
    if len(username) < 5 or len(username) > 15:
        return False
    if username[0].isdigit():
        return False
    if " " in username:
        return False
    if not username.isalnum():
        return False
    return True

assert is_valid_username("User123") == True
assert is_valid_username("12User") == False
assert is_valid_username("Us er") == False
print("username all tests passed")

```

Output:



```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN\
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode
ndled\libs\debugpy\launcher' '49698' '--' 'c:\Users\NITHIN\OneDrive\Des
username all tests passed
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> 
```

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".

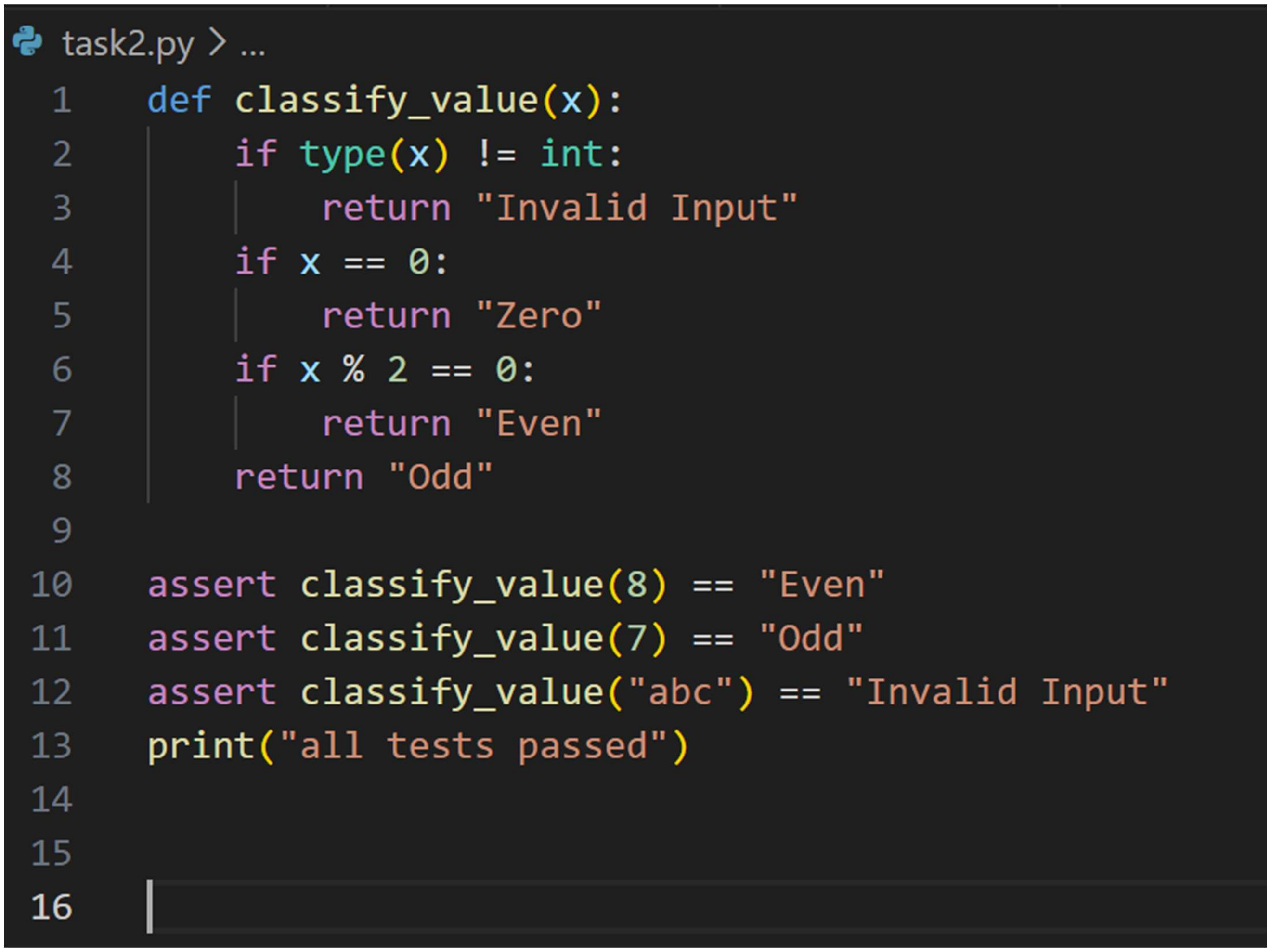
Example Assert Test Cases:

```
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.

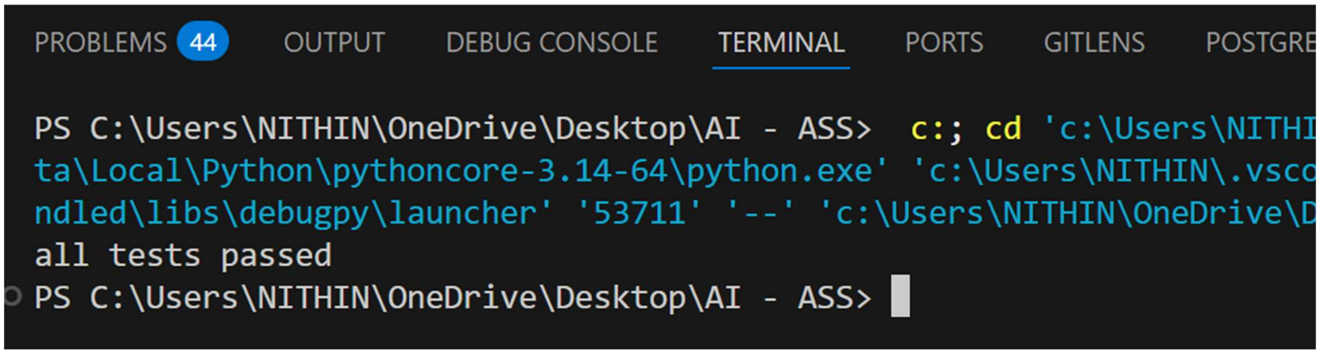
CODE:



task2.py > ...

```python
def classify_value(x):
    if type(x) != int:
        return "Invalid Input"
    if x == 0:
        return "Zero"
    if x % 2 == 0:
        return "Even"
    return "Odd"

assert classify_value(8) == "Even"
assert classify_value(7) == "Odd"
assert classify_value("abc") == "Invalid Input"
print("all tests passed")



```

Output:



```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHI
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vsco
ndled\libs\debugpy\launcher' '53711' '--' 'c:\Users\NITHIN\OneDrive\D
all tests passed
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> 
```

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

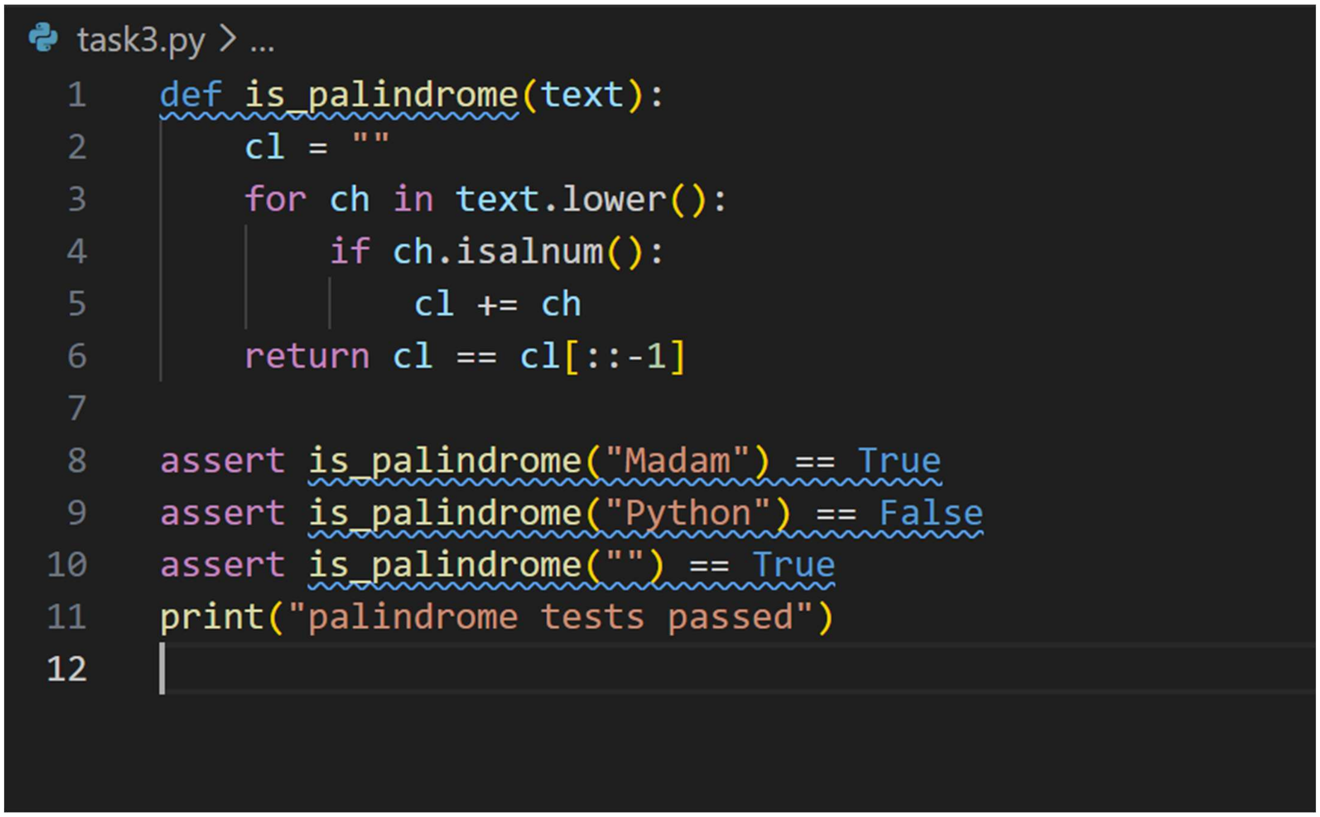
assert is_palindrome("Python") == False

Expected Output #3:

• Function correctly identifying palindromes and passing all

AI-generated tests.

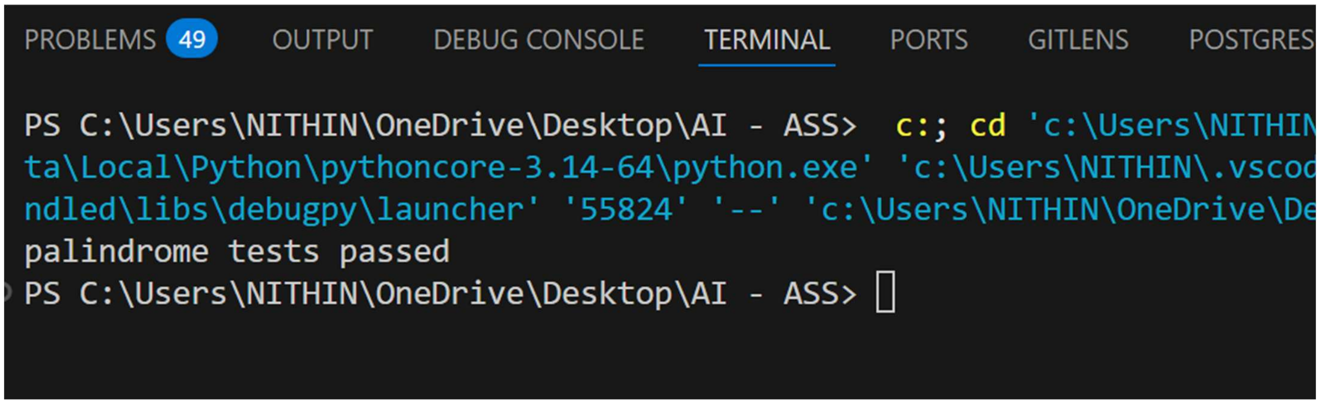
CODE:



task3.py > ...

```python
def is_palindrome(text):
    cl = ""
    for ch in text.lower():
        if ch.isalnum():
            cl += ch
    return cl == cl[::-1]

assert is_palindrome("Madam") == True
assert is_palindrome("Python") == False
assert is_palindrome("") == True
print("palindrome tests passed")

```

Output:



PROBLEMS 49 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES

```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscod
ndled\libs\debugpy\launcher' '55824' '--' 'c:\Users\NITHIN\OneDrive\De
palindrome tests passed
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> 
```

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

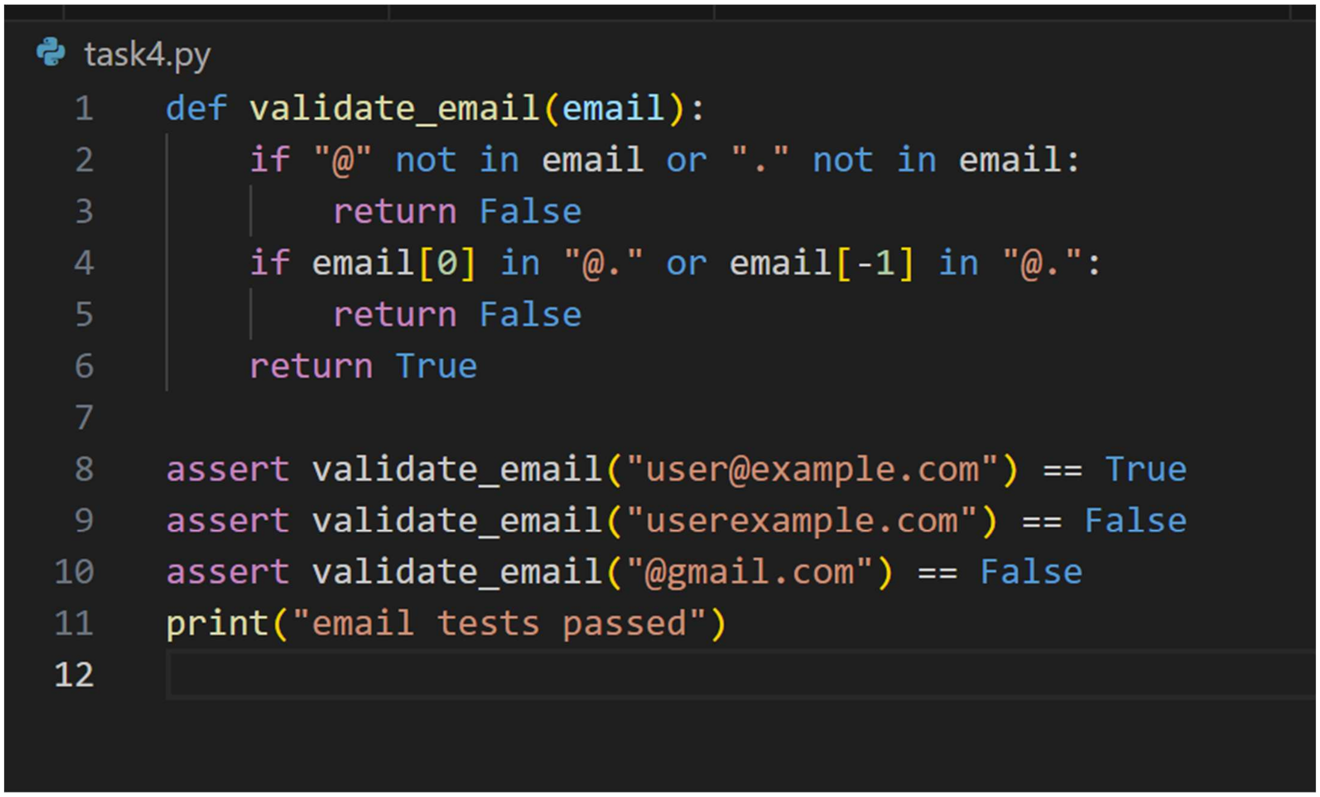
assert validate_email("@gmail.com") == False

Expected Output #5:

• Email validation function passing all AI-generated test cases

and handling edge cases correctly.

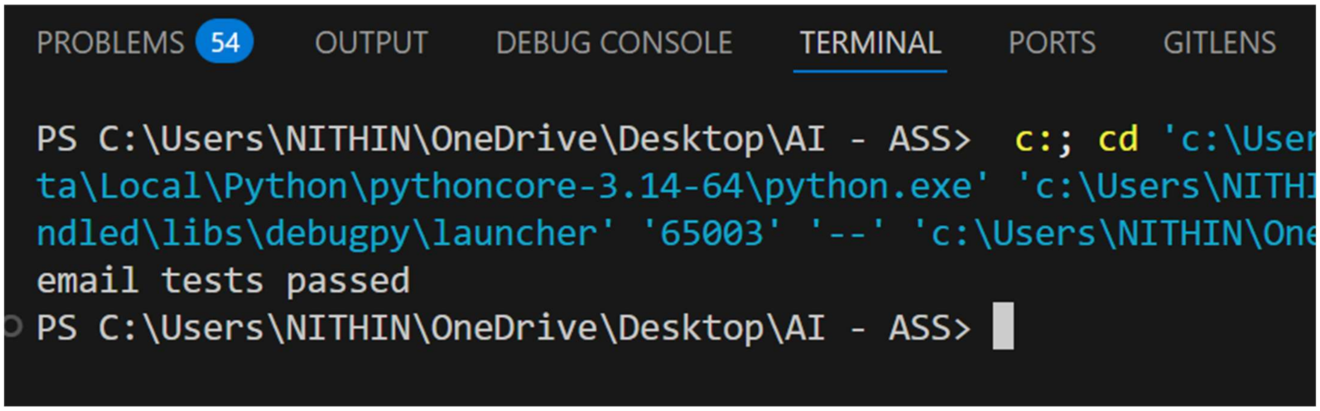
CODE:



task4.py

```python
def validate_email(email):
    if "@" not in email or "." not in email:
        return False
    if email[0] in "@." or email[-1] in "@.":
        return False
    return True

assert validate_email("user@example.com") == True
assert validate_email("userexample.com") == False
assert validate_email("@gmail.com") == False
print("email tests passed")

```

Output:



```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\User
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHI
ndled\libs\debugpy\launcher' '65003' '--' 'c:\Users\NITHIN\One
email tests passed
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> 
```

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.

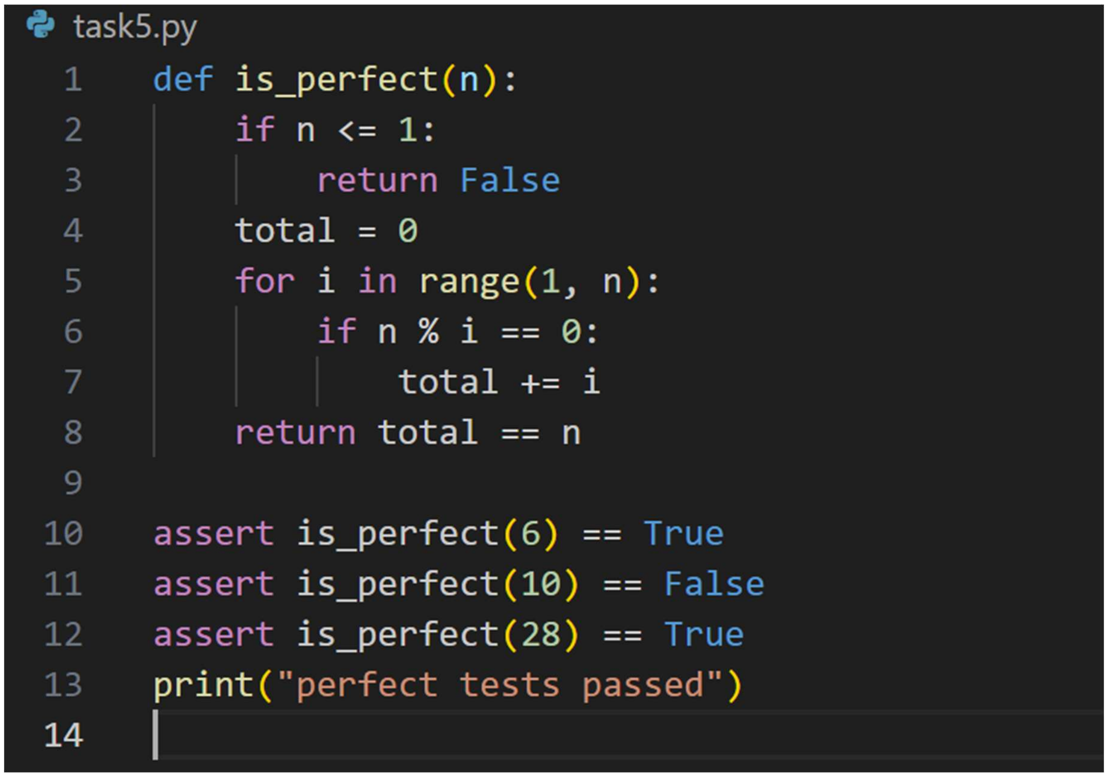
o Edge case: 1.

o Negative number case.

o Larger case: 28.

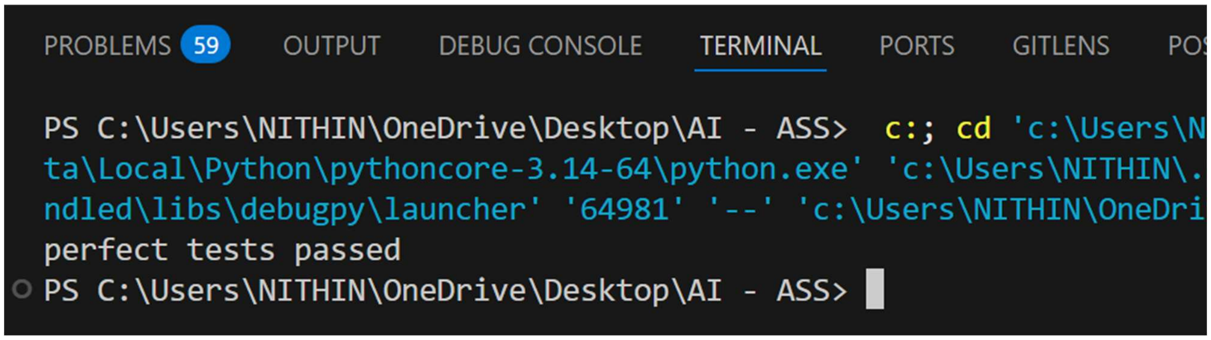
• Requirement: Validate correctness with assertions.

CODE:



```python
def is_perfect(n):
    if n <= 1:
        return False
    total = 0
    for i in range(1, n):
        if n % i == 0:
            total += i
    return total == n

assert is_perfect(6) == True
assert is_perfect(10) == False
assert is_perfect(28) == True
print("perfect tests passed")
```

Output:



```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\N
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.
ndled\libs\debugpy\launcher' '64981' '--' 'c:\Users\NITHIN\OneDri
perfect tests passed
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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.

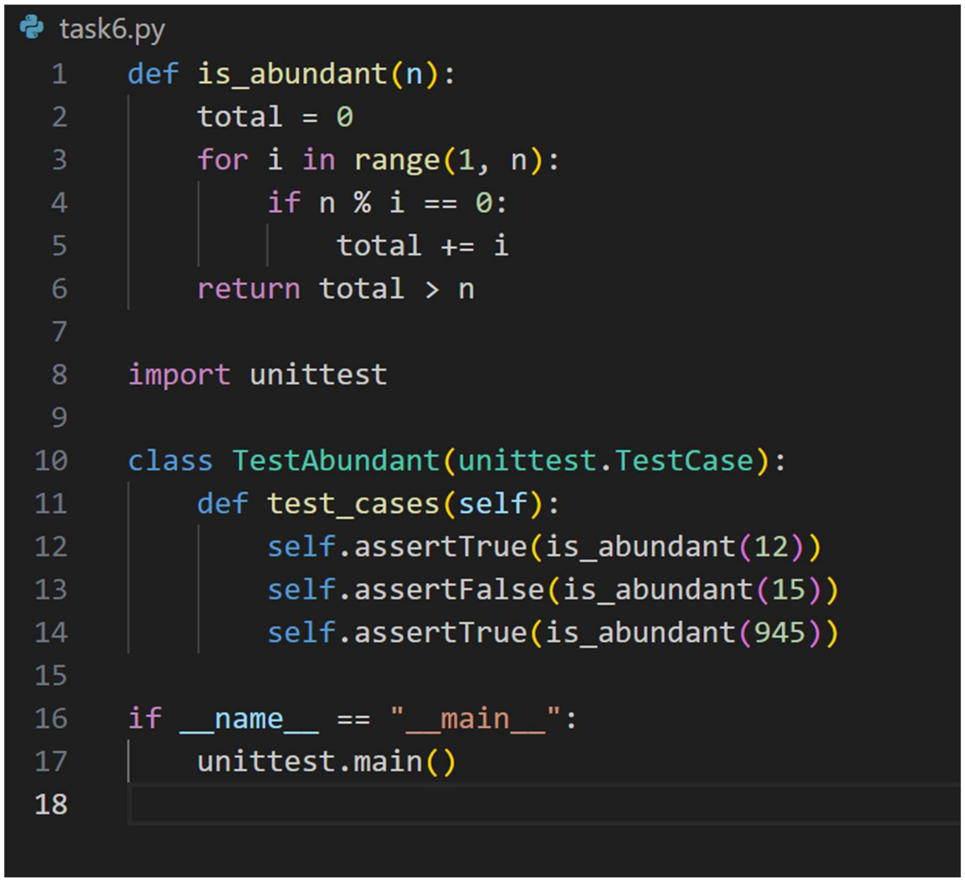
o Edge case: 1.

o Negative number case.

o Large case: 945.

Requirement: Validate correctness with unittest

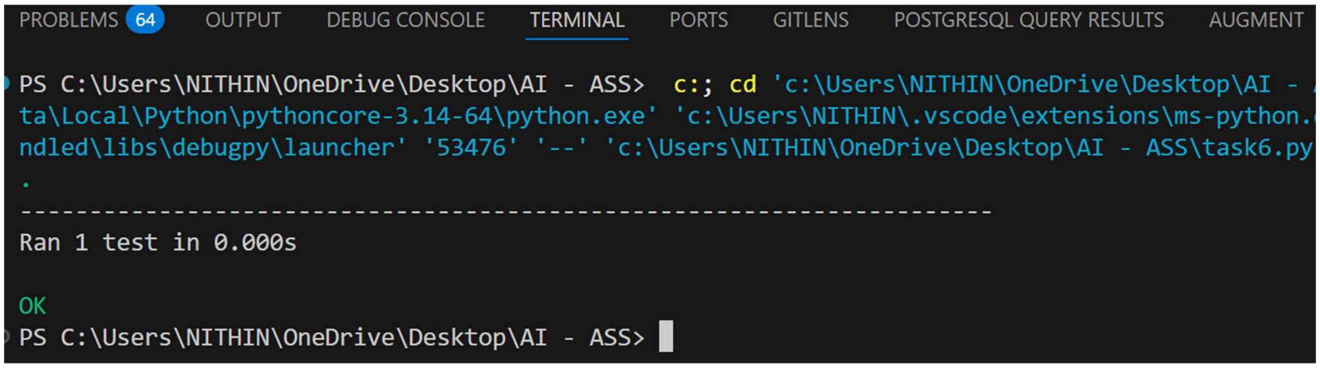
CODE:



task6.py

```python
def is_abundant(n):
    total = 0
    for i in range(1, n):
        if n % i == 0:
            total += i
    return total > n

import unittest

class TestAbundant(unittest.TestCase):
    def test_cases(self):
        self.assertTrue(is_abundant(12))
        self.assertFalse(is_abundant(15))
        self.assertTrue(is_abundant(945))

if __name__ == "__main__":
    unittest.main()

```

Output:



```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN\OneDrive\Desktop\AI -
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.
ndled\libs\debugpy\launcher' '53476' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\task6.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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.

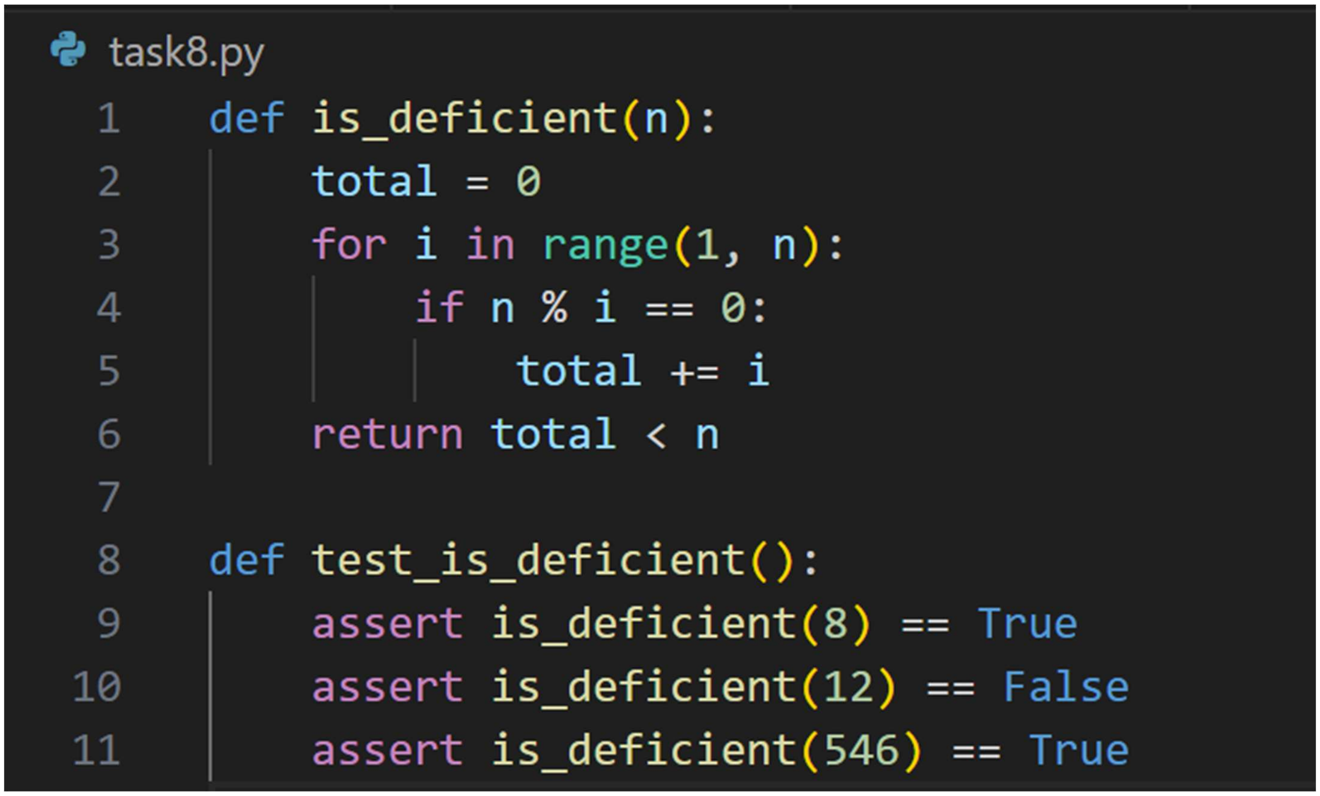
o Edge case: 1.

o Negative number case.

o Large case: 546.

Requirement: Validate correctness with pytest.

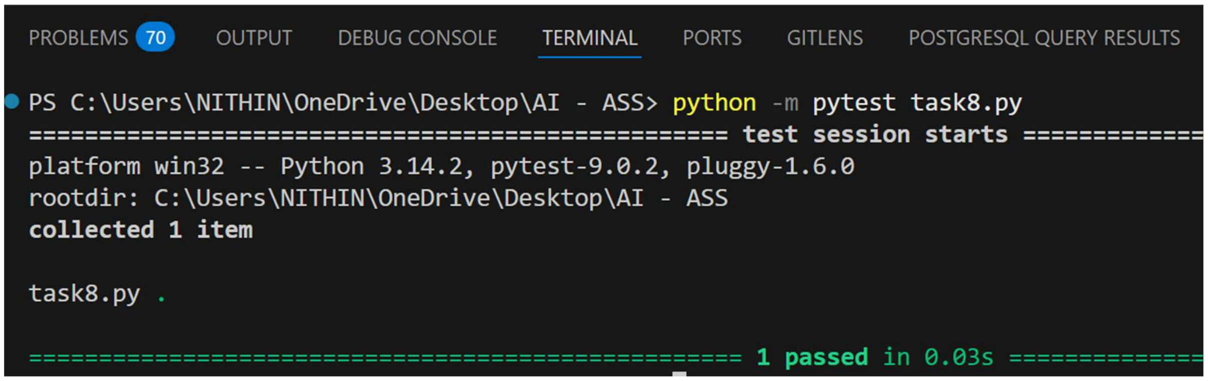
Code:



task8.py

```python
def is_deficient(n):
    total = 0
    for i in range(1, n):
        if n % i == 0:
            total += i
    return total < n

def test_is_deficient():
    assert is_deficient(8) == True
    assert is_deficient(12) == False
    assert is_deficient(546) == True
```

Output:



PROBLEMS 70 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRESQL QUERY RESULTS

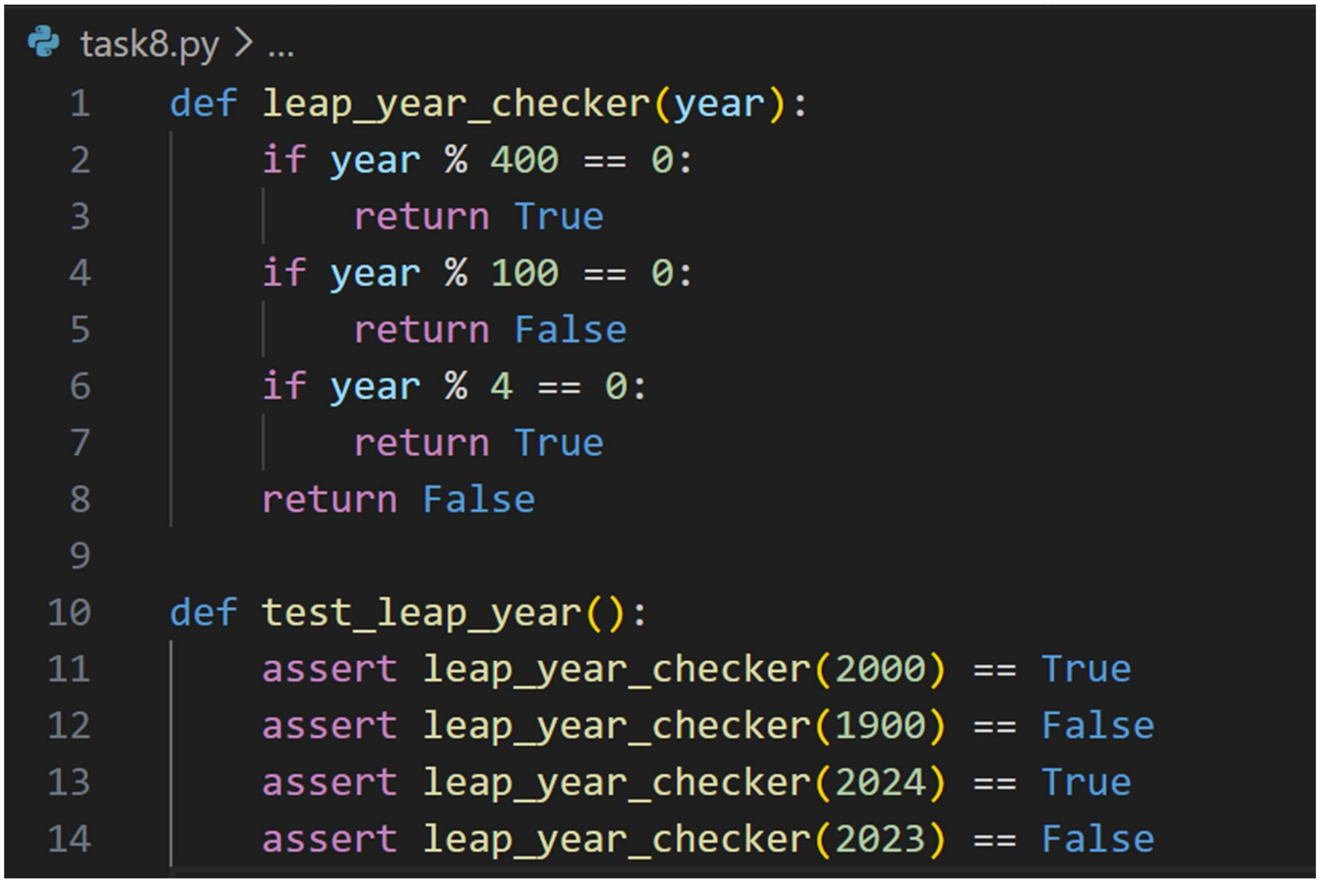
```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> python -m pytest task8.py
================================================== test session starts ============
platform win32 -- Python 3.14.2, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\NITHIN\OneDrive\Desktop\AI - ASS
collected 1 item

task8.py .

=================================================== 1 passed in 0.03s =============
```

Task 8 :

Write a function LeapYearChecker and validate its implementation

using 10 pytest test cases

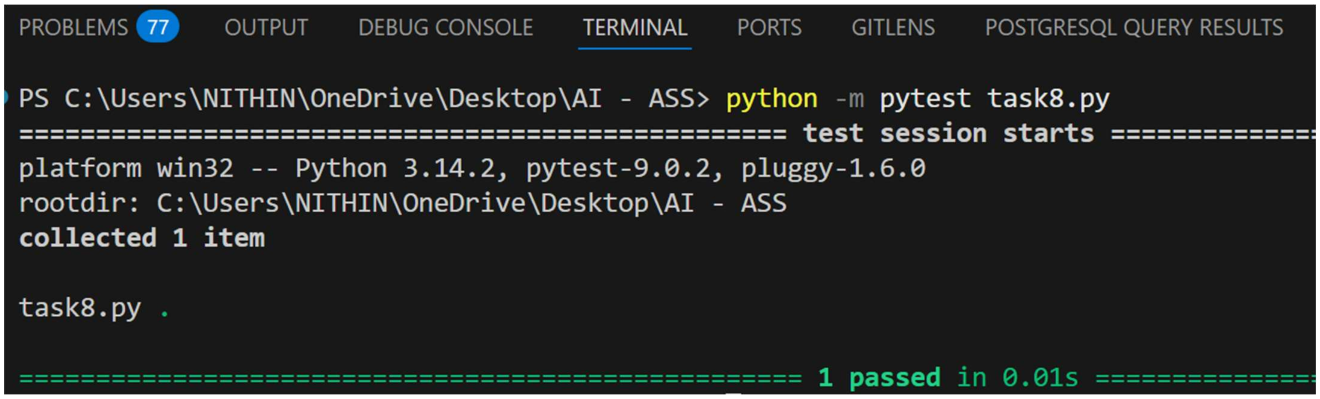
CODE:



task8.py > ...

```python
def leap_year_checker(year):
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    if year % 4 == 0:
        return True
    return False

def test_leap_year():
    assert leap_year_checker(2000) == True
    assert leap_year_checker(1900) == False
    assert leap_year_checker(2024) == True
    assert leap_year_checker(2023) == False
```

Output:



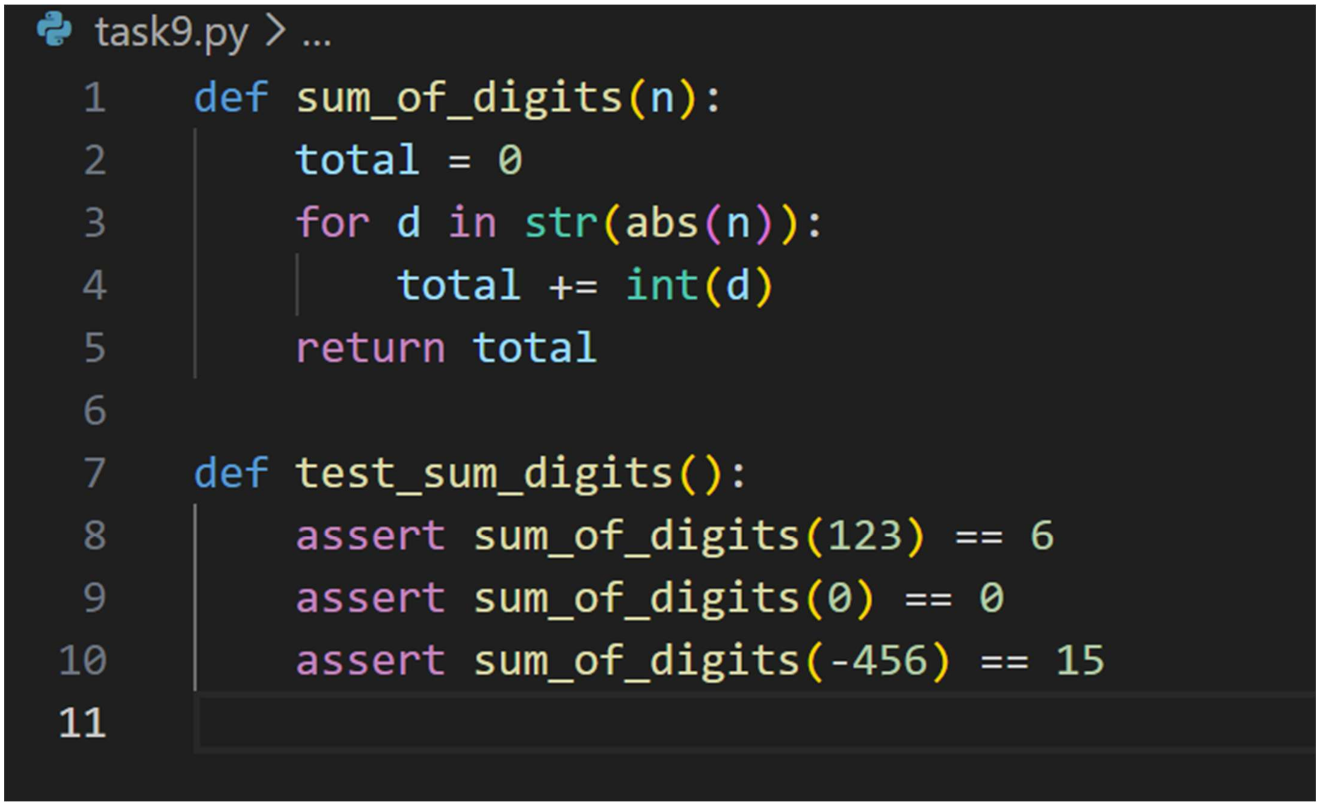
```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> python -m pytest task8.py
================================================== test session starts =============
platform win32 -- Python 3.14.2, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\NITHIN\OneDrive\Desktop\AI - ASS
collected 1 item

task8.py .

================================================== 1 passed in 0.01s ==============
```

Task 9 :

Write a function SumOfDigits and validate its implementation

using 7 pytest test cases.

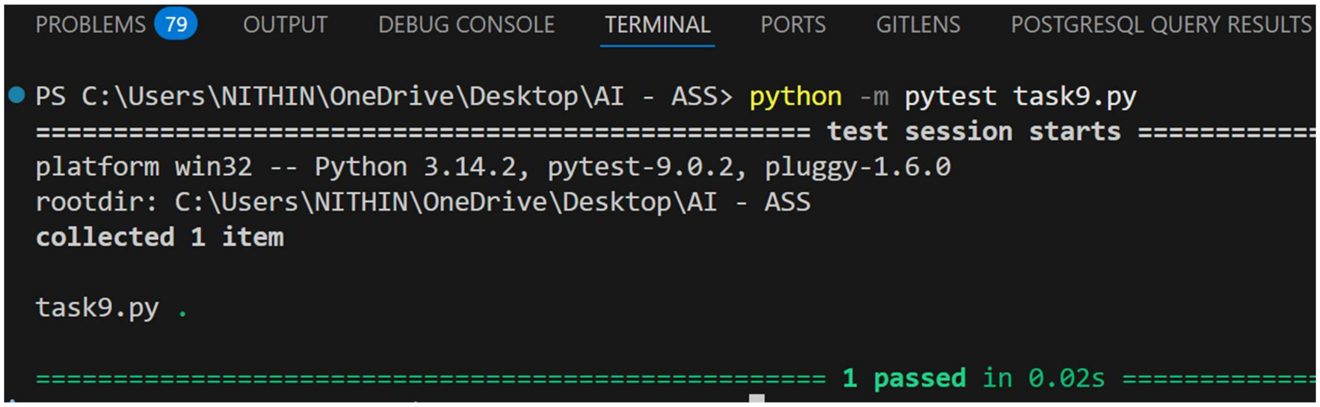
Code:



task9.py > ...

```python
def sum_of_digits(n):
    total = 0
    for d in str(abs(n)):
        total += int(d)
    return total

def test_sum_digits():
    assert sum_of_digits(123) == 6
    assert sum_of_digits(0) == 0
    assert sum_of_digits(-456) == 15
```

Output:



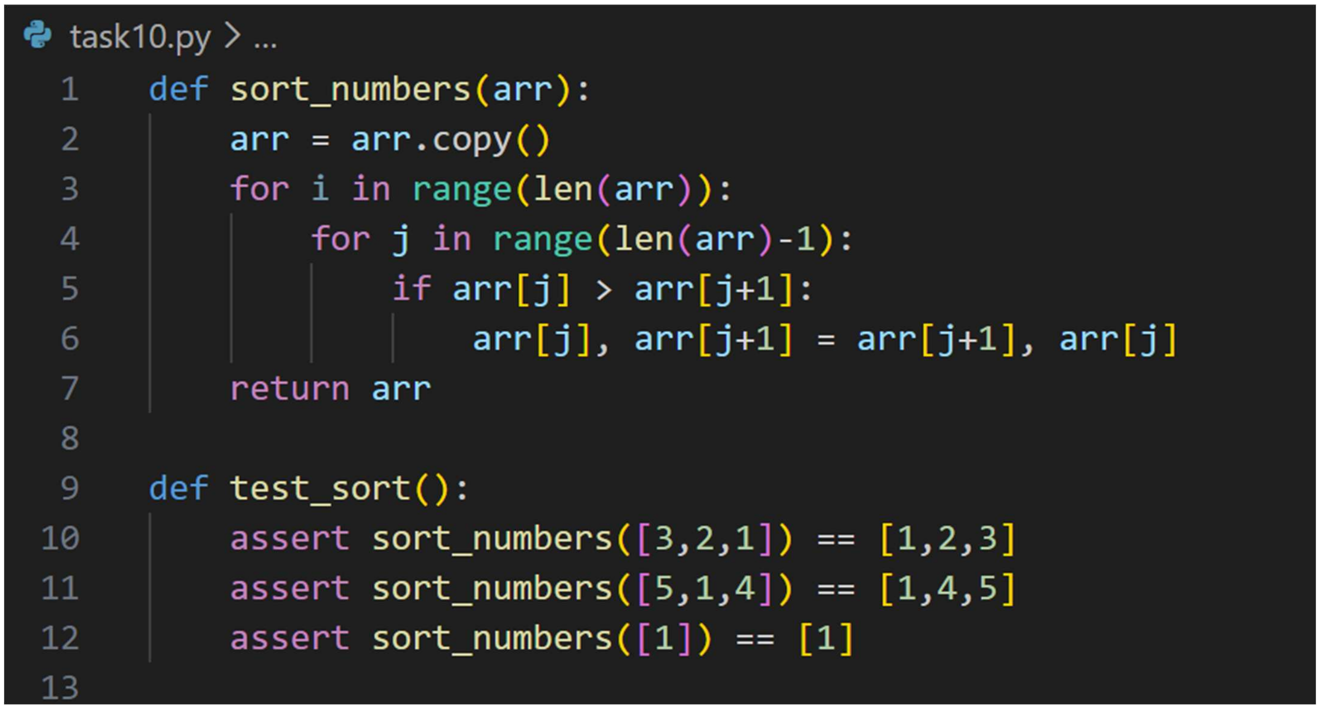
```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> python -m pytest task9.py
==================================================== test session starts ===========
platform win32 -- Python 3.14.2, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\NITHIN\OneDrive\Desktop\AI - ASS
collected 1 item

task9.py .

==================================================== 1 passed in 0.02s ============
```

Task 10 :

Write a function SortNumbers (implement bubble sort) and validate

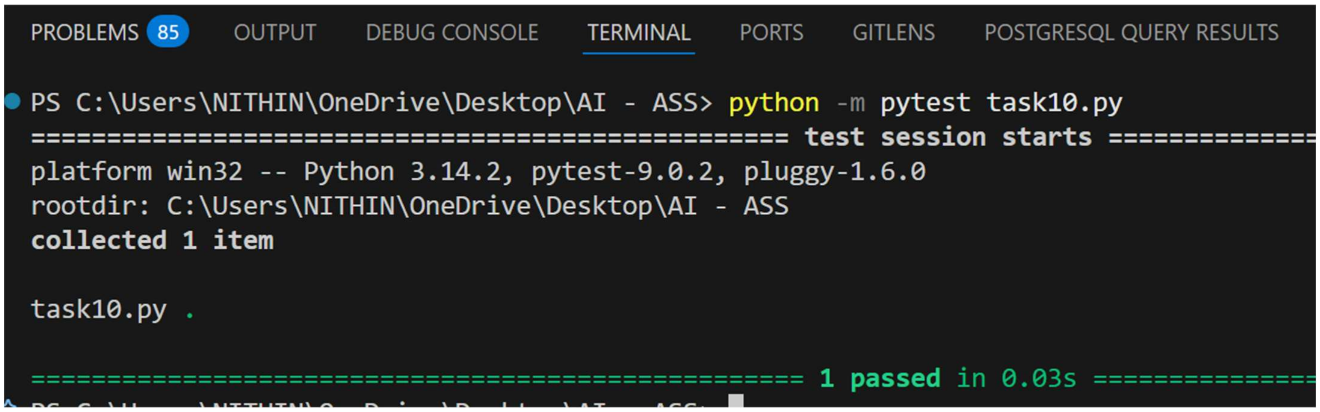
its implementation using 25 pytest test cases.

Code:



```python
def sort_numbers(arr):
    arr = arr.copy()
    for i in range(len(arr)):
        for j in range(len(arr)-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

def test_sort():
    assert sort_numbers([3,2,1]) == [1,2,3]
    assert sort_numbers([5,1,4]) == [1,4,5]
    assert sort_numbers([1]) == [1]

```

Output:



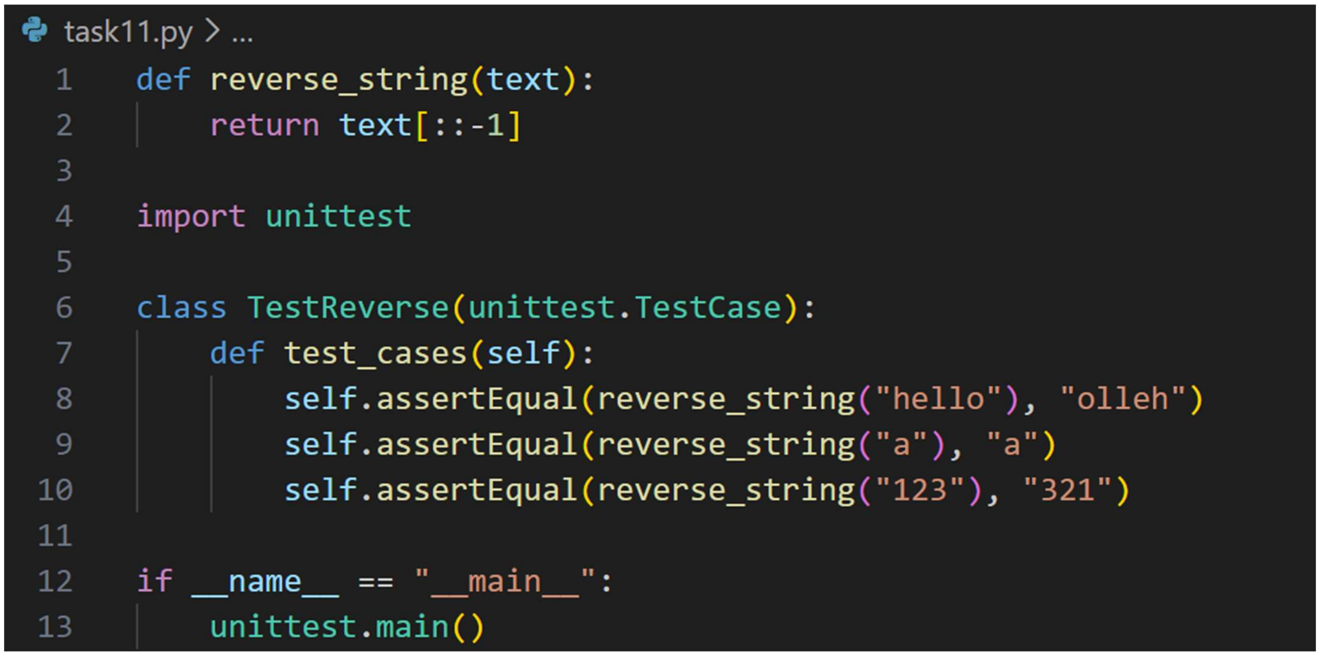
```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> python -m pytest task10.py
===================================================== test session starts =============
platform win32 -- Python 3.14.2, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\NITHIN\OneDrive\Desktop\AI - ASS
collected 1 item

task10.py .

====================================================== 1 passed in 0.03s ==============
```

Task 11 :

Write a function ReverseString and validate its implementation

using 5 unittest test cases

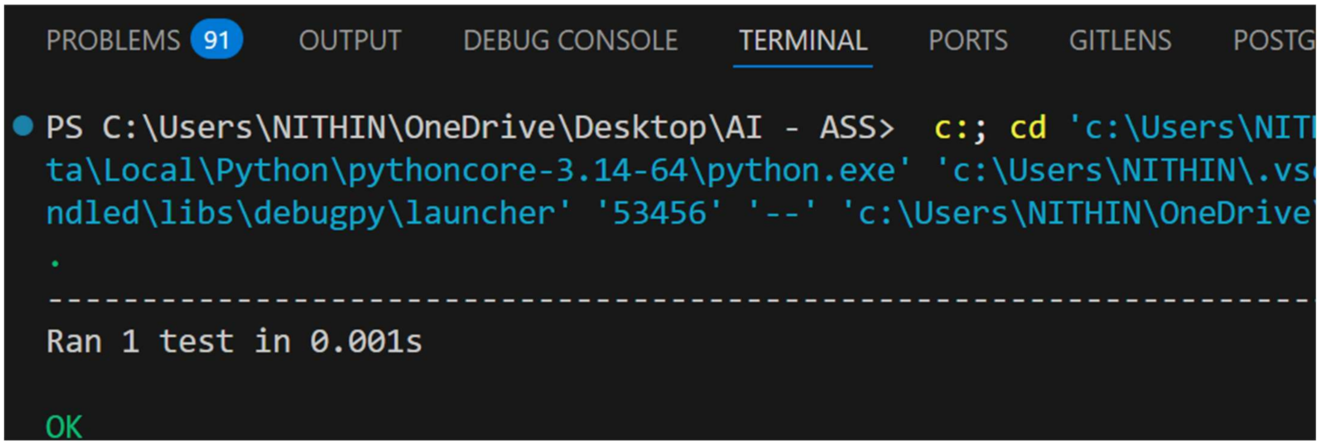
Code:



task11.py > ...

```python
def reverse_string(text):
    return text[::-1]

import unittest

class TestReverse(unittest.TestCase):
    def test_cases(self):
        self.assertEqual(reverse_string("hello"), "olleh")
        self.assertEqual(reverse_string("a"), "a")
        self.assertEqual(reverse_string("123"), "321")

if __name__ == "__main__":
    unittest.main()
```

Output:



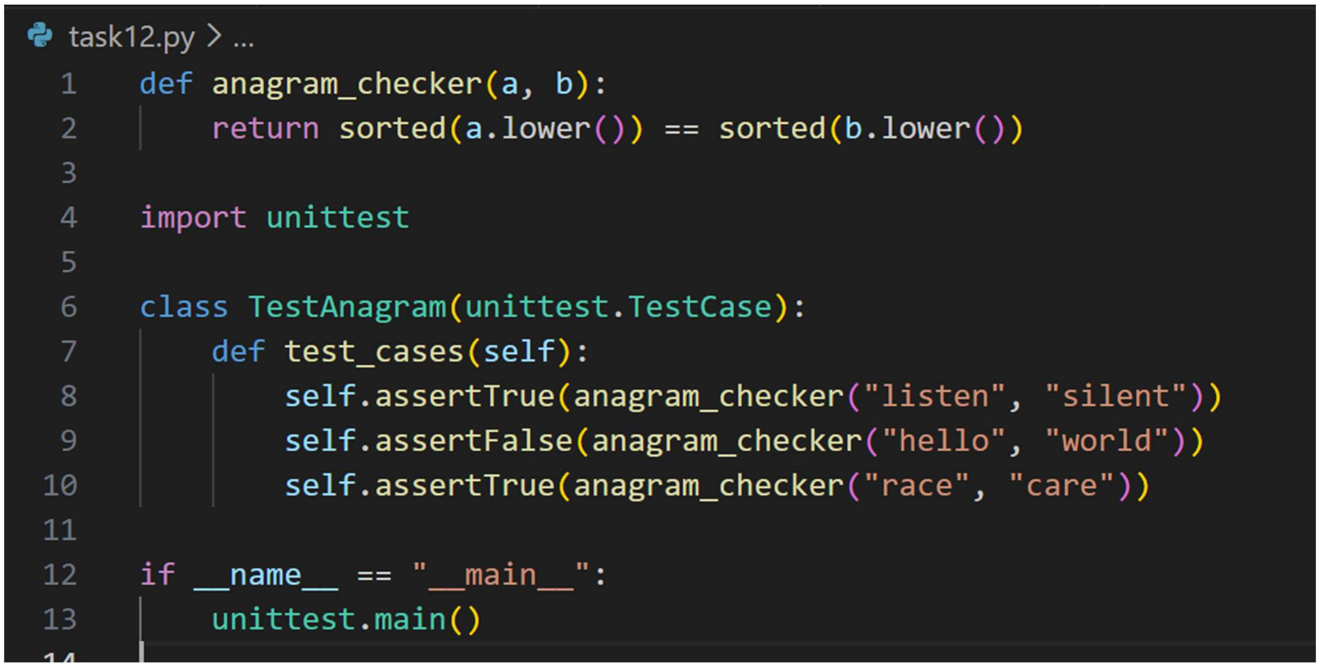
```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITI
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vs
ndled\libs\debugpy\launcher' '53456' '--' 'c:\Users\NITHIN\OneDrive
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

Task 12 :

Write a function AnagramChecker and validate its implementation

using 10 unittest test cases.

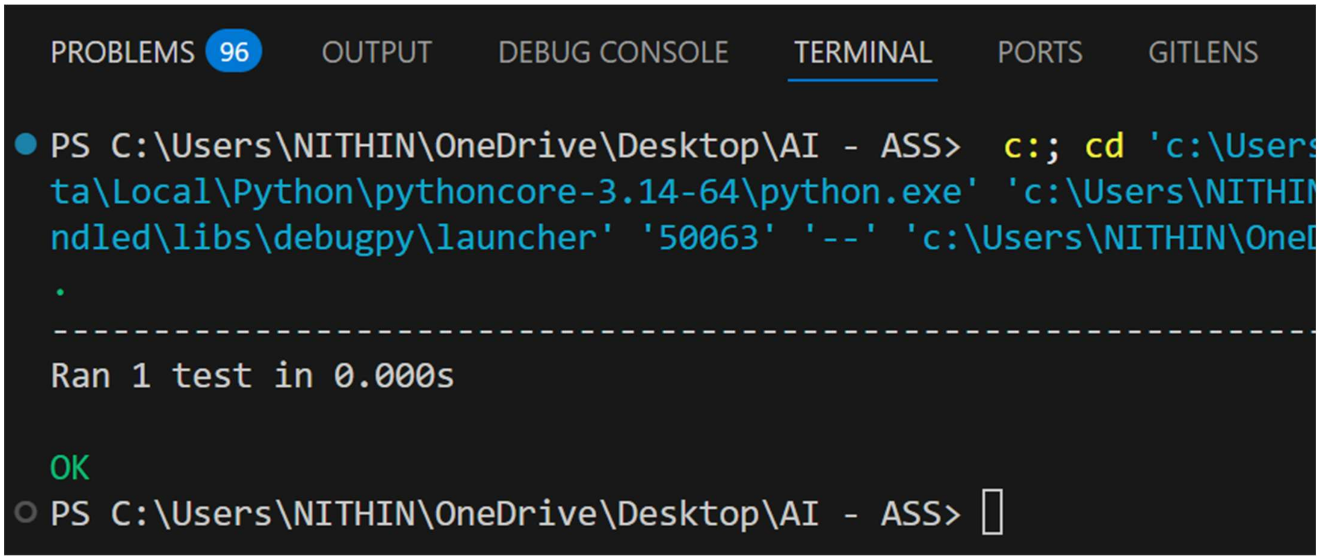
Code:



task12.py > ...

```python
def anagram_checker(a, b):
    return sorted(a.lower()) == sorted(b.lower())

import unittest

class TestAnagram(unittest.TestCase):
    def test_cases(self):
        self.assertTrue(anagram_checker("listen", "silent"))
        self.assertFalse(anagram_checker("hello", "world"))
        self.assertTrue(anagram_checker("race", "care"))

if __name__ == "__main__":
    unittest.main()
```

Output:



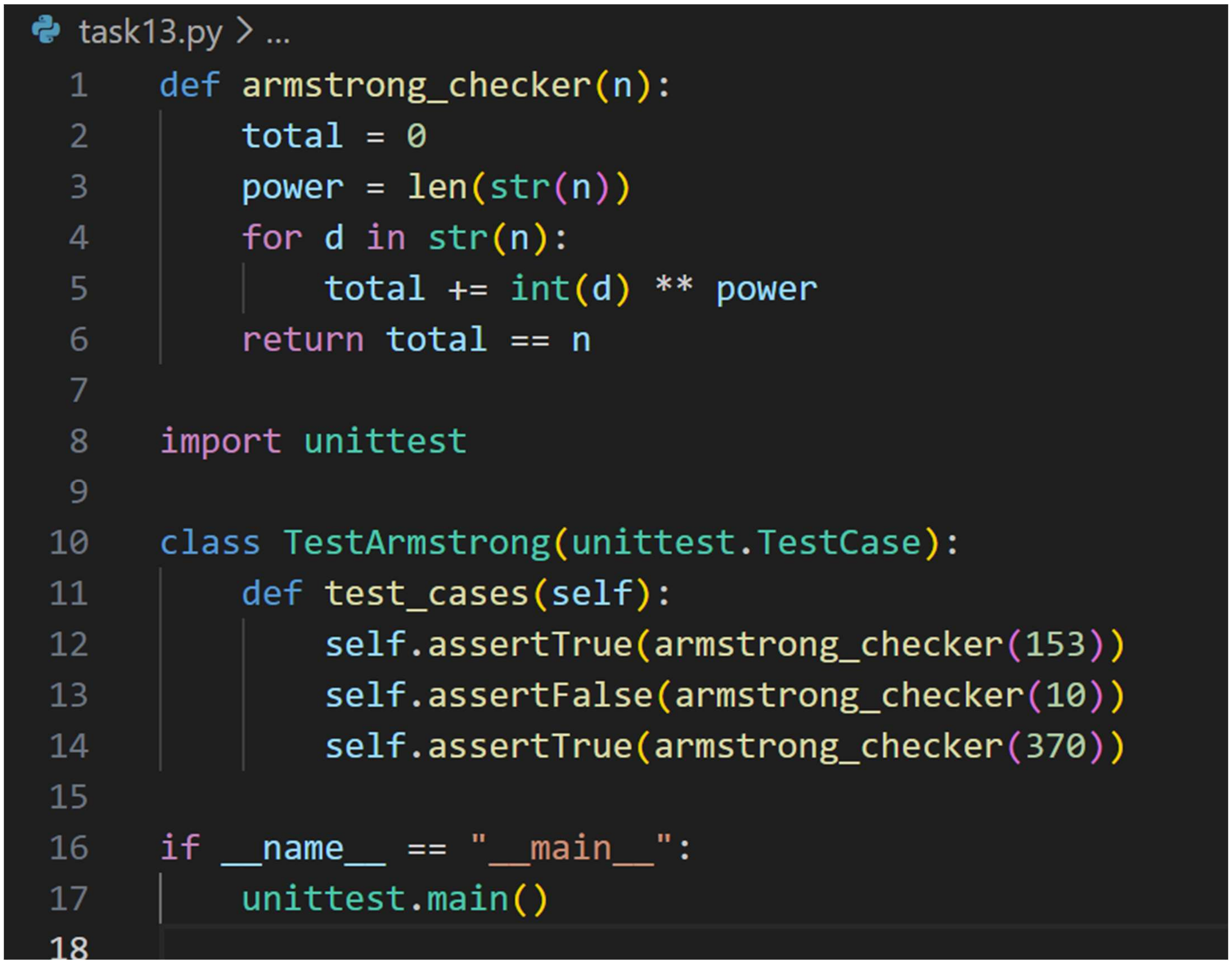
```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHI
ndled\libs\debugpy\launcher' '50063' '--' 'c:\Users\NITHIN\OneD
.
----------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

Task 13 :

Write a function ArmstrongChecker and validate its implementation

using 8 unittest test cases.

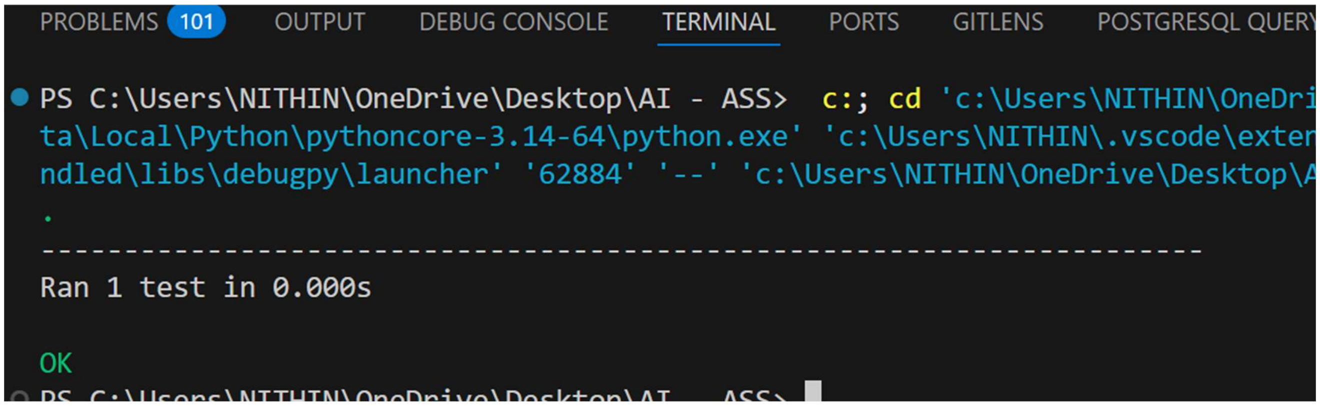
Code:



task13.py > ...

```python
def armstrong_checker(n):
    total = 0
    power = len(str(n))
    for d in str(n):
        total += int(d) ** power
    return total == n

import unittest

class TestArmstrong(unittest.TestCase):
    def test_cases(self):
        self.assertTrue(armstrong_checker(153))
        self.assertFalse(armstrong_checker(10))
        self.assertTrue(armstrong_checker(370))

if __name__ == "__main__":
    unittest.main()
```

Output:



```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN\OneDri
ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\exten
ndled\libs\debugpy\launcher' '62884' '--' 'c:\Users\NITHIN\OneDrive\Desktop\A
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```