

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:

- Username length must be between 5 and 15 characters.
- Must contain only alphabets and digits.
- Must not start with a digit.
- 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 > ...
1 def is_valid_username(username):
2     if len(username) < 5 or len(username) > 15:
3         return False
4     if username[0].isdigit():
5         return False
6     if " " in username:
7         return False
8     if not username.isalnum():
9         return False
10    return True
11
12 assert is_valid_username("User123") == True
13 assert is_valid_username("12User") == False
14 assert is_valid_username("Us er") == False
15 print("username all tests passed")
16
17
```

Output:

PROBLEMS 39 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\.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:

- If input is an integer, classify as "Even" or "Odd".
- If input is 0, return "Zero".
- 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 > ...
1  def classify_value(x):
2      if type(x) != int:
3          return "Invalid Input"
4      if x == 0:
5          return "Zero"
6      if x % 2 == 0:
7          return "Even"
8      return "Odd"
9
10 assert classify_value(8) == "Even"
11 assert classify_value(7) == "Odd"
12 assert classify_value("abc") == "Invalid Input"
13 print("all tests passed")
14
15
16 |
```

Output:

The screenshot shows a terminal window with the following content:

```
PROBLEMS 44 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRE
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c;; cd 'c:\Users\NITHIN\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\libs\debugpy\launcher' '53711' '--' 'c:\Users\NITHIN\OneDrive\AI - ASS>
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:

- Ignore case, spaces, and punctuation.
 - 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 > ...
1 def is_palindrome(text):
2     cl = ""
3     for ch in text.lower():
4         if ch.isalnum():
5             cl += ch
6     return cl == cl[::-1]
7
8 assert is_palindrome("Madam") == True
9 assert is_palindrome("Python") == False
10 assert is_palindrome("") == True
11 print("palindrome tests passed")
12 |
```

Output:

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the command-line output of running the Python script task3.py. The output shows the script's logic for checking palindromes and its test cases, which all pass.

```
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\.vscode\ndled\libs\debugpy\launcher' '55824' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS> 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
1 def validate_email(email):
2     if "@" not in email or "." not in email:
3         return False
4     if email[0] in "@" or email[-1] in "@":
5         return False
6     return True
7
8 assert validate_email("user@example.com") == True
9 assert validate_email("userexample.com") == False
10 assert validate_email("@gmail.com") == False
11 print("email tests passed")
12
```

Output:

The screenshot shows a terminal window with the following interface elements at the top:

- PROBLEMS (54)
- OUTPUT
- DEBUG CONSOLE
- TERMINAL
- PORTS
- GITLENS

The terminal window displays the following text:

```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c;; cd 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS'; python task4.py
email tests passed
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

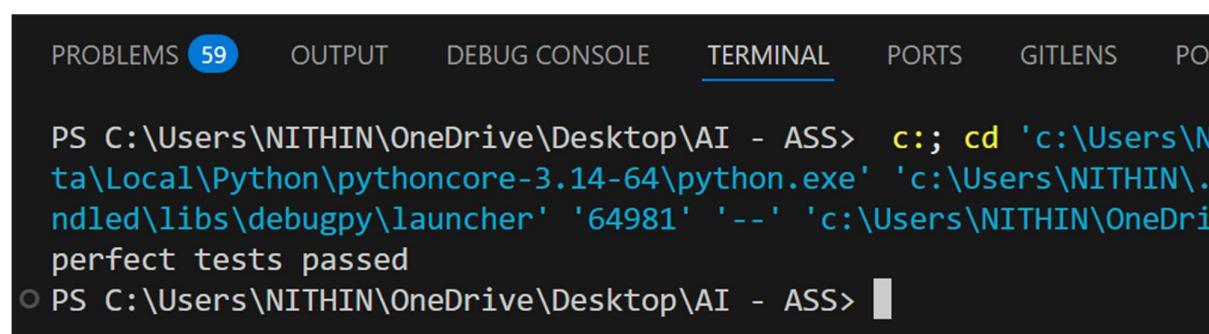
Task 5 (Perfect Number Checker – Test Case Design)

- Function: Check if a number is a perfect number (sum of divisors = number).
- Test Cases to Design:
 - Normal case: 6 → True, 10 → False.
 - Edge case: 1.
 - Negative number case.
 - Larger case: 28.
- Requirement: Validate correctness with assertions.

CODE:

```
task5.py
1  def is_perfect(n):
2      if n <= 1:
3          return False
4      total = 0
5      for i in range(1, n):
6          if n % i == 0:
7              total += i
8      return total == n
9
10 assert is_perfect(6) == True
11 assert is_perfect(10) == False
12 assert is_perfect(28) == True
13 print("perfect tests passed")
14 |
```

Output:



```
PROBLEMS 59 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POS
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c:; cd 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS'; python 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\task5.py'
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:
 - Normal case: 12 → True, 15 → False.
 - Edge case: 1.
 - Negative number case.
 - Large case: 945.

Requirement: Validate correctness with unittest

CODE:

```
task6.py
1  def is_abundant(n):
2      total = 0
3      for i in range(1, n):
4          if n % i == 0:
5              total += i
6      return total > n
7
8  import unittest
9
10 class TestAbundant(unittest.TestCase):
11     def test_cases(self):
12         self.assertTrue(is_abundant(12))
13         self.assertFalse(is_abundant(15))
14         self.assertTrue(is_abundant(945))
15
16 if __name__ == "__main__":
17     unittest.main()
18
```

Output:

```
PROBLEMS 64 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRESQL QUERY RESULTS AUGMENT
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c:; cd 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.vscode-pylint\lib\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:

- Normal case: 8 → True, 12 → False.

- Edge case: 1.

- Negative number case.

- Large case: 546.

Requirement: Validate correctness with pytest.

Code:

```
task8.py
1  def is_deficient(n):
2      total = 0
3      for i in range(1, n):
4          if n % i == 0:
5              total += i
6      return total < n
7
8  def test_is_deficient():
9      assert is_deficient(8) == True
10     assert is_deficient(12) == False
11     assert is_deficient(546) == True
```

Output:

```
PROBLEMS 70    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS    POSTGRES 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 =====
```

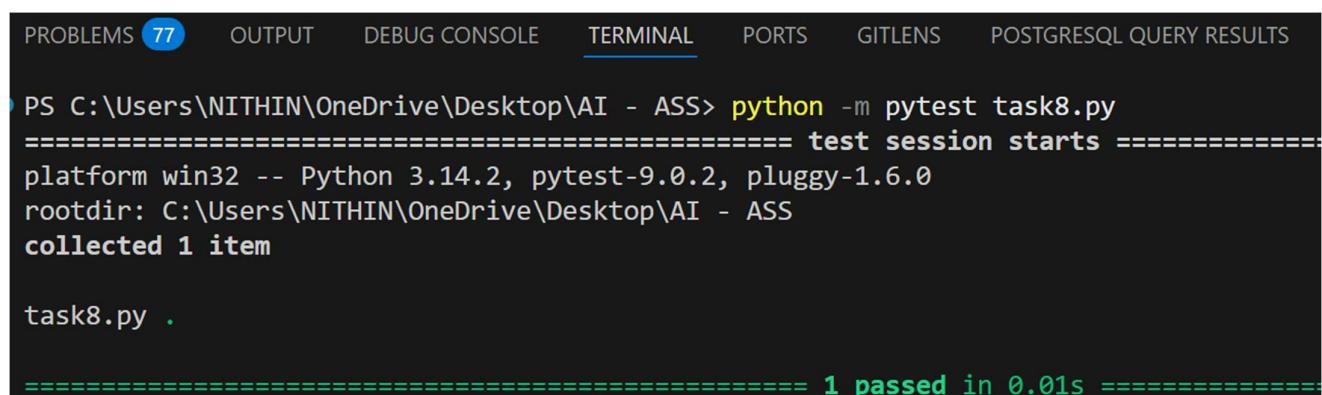
Task 8 :

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

CODE:

```
task8.py > ...
1  def leap_year_checker(year):
2      if year % 400 == 0:
3          return True
4      if year % 100 == 0:
5          return False
6      if year % 4 == 0:
7          return True
8      return False
9
10 def test_leap_year():
11     assert leap_year_checker(2000) == True
12     assert leap_year_checker(1900) == False
13     assert leap_year_checker(2024) == True
14     assert leap_year_checker(2023) == False
```

Output:



The screenshot shows a terminal window with the following output:

```
PROBLEMS 77 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES QUERY RESULTS
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> python -m pytest task8.py
=====
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 > ...
1  def sum_of_digits(n):
2      total = 0
3      for d in str(abs(n)):
4          total += int(d)
5      return total
6
7  def test_sum_digits():
8      assert sum_of_digits(123) == 6
9      assert sum_of_digits(0) == 0
10     assert sum_of_digits(-456) == 15
11
```

Output:

```
PROBLEMS 79 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES QUERY RESULTS
● PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> python -m pytest task9.py
=====
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 =====
```

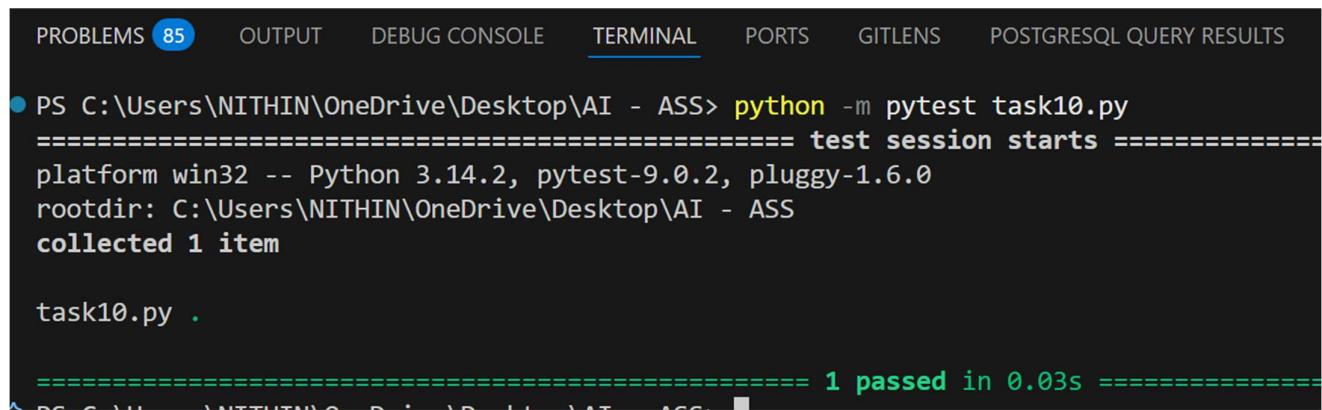
Task 10 :

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

Code:

```
task10.py > ...
1  def sort_numbers(arr):
2      arr = arr.copy()
3      for i in range(len(arr)):
4          for j in range(len(arr)-1):
5              if arr[j] > arr[j+1]:
6                  arr[j], arr[j+1] = arr[j+1], arr[j]
7      return arr
8
9  def test_sort():
10     assert sort_numbers([3,2,1]) == [1,2,3]
11     assert sort_numbers([5,1,4]) == [1,4,5]
12     assert sort_numbers([1]) == [1]
13
```

Output:



The screenshot shows a terminal window with the following output:

```
PROBLEMS 85 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES QUERY RESULTS
● 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 > ...
1  def reverse_string(text):
2      return text[::-1]
3
4  import unittest
5
6  class TestReverse(unittest.TestCase):
7      def test_cases(self):
8          self.assertEqual(reverse_string("hello"), "olleh")
9          self.assertEqual(reverse_string("a"), "a")
10         self.assertEqual(reverse_string("123"), "321")
11
12     if __name__ == "__main__":
13         unittest.main()
```

Output:

The screenshot shows a terminal window with several tabs at the top: PROBLEMS (91), OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS, GITLENS, and POSTG. The terminal content is as follows:

```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> cd 'c:\Users\NITHIN\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\vscode\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 > ...
1  def anagram_checker(a, b):
2      return sorted(a.lower()) == sorted(b.lower())
3
4  import unittest
5
6  class TestAnagram(unittest.TestCase):
7      def test_cases(self):
8          self.assertTrue(anagram_checker("listen", "silent"))
9          self.assertFalse(anagram_checker("hello", "world"))
10         self.assertTrue(anagram_checker("race", "care"))
11
12 if __name__ == "__main__":
13     unittest.main()
14
```

Output:

```
PROBLEMS 96 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
● PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c:; cd 'c:\Users\ta\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\ndled\libs\debugpy\launcher' '50063' '--' 'c:\Users\NITHIN\One
.
-----
Ran 1 test in 0.000s
OK
○ PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> 
```

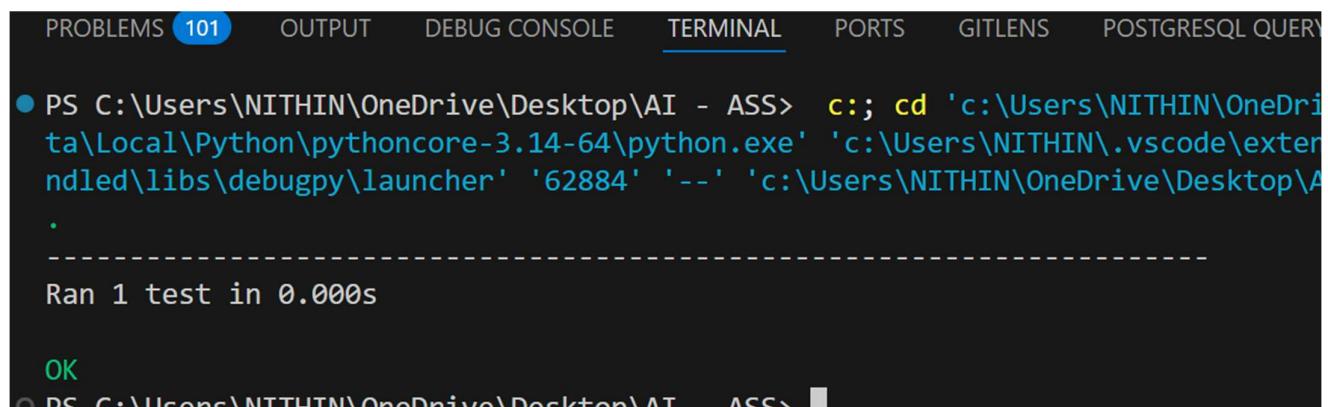
Task 13 :

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

Code:

```
task13.py > ...
1  def armstrong_checker(n):
2      total = 0
3      power = len(str(n))
4      for d in str(n):
5          total += int(d) ** power
6      return total == n
7
8  import unittest
9
10 class TestArmstrong(unittest.TestCase):
11     def test_cases(self):
12         self.assertTrue(armstrong_checker(153))
13         self.assertFalse(armstrong_checker(10))
14         self.assertTrue(armstrong_checker(370))
15
16 if __name__ == "__main__":
17     unittest.main()
18
```

Output:



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the execution of the task13.py script. The output shows the script's logic for calculating Armstrong numbers and running 8 test cases using the unittest module. The test results indicate that all three provided test cases pass, resulting in an overall 'OK' status.

```
PROBLEMS 101 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES QUERY

● PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c;; cd 'c:\Users\NITHIN\OneDrive\Desktop\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\debugpy\launcher' '62884' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS> .
-----
Ran 1 test in 0.000s

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