

# Ai Assisted Coding

## Assignment-8

**Name :** G.Samshray

**Ht.no :** 2303A51670

**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.

```
Assignment-8.py X
Wed.py > Assignment-8.py > ...
1 def is_valid_username(username):
2     if len(username) < 5 or len(username) > 15:
3         return False
4     if not username[0].isalpha():
5         return False
6     for char in username:
7         if not (char.isalnum() or char == '_'):
8             return False
9     return True
10 # Test cases
11 assert is_valid_username("user_123") == True
12 assert is_valid_username("1user") == False
13 assert is_valid_username("us") == False
14 print("All test cases passed!")

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe"
● All test cases passed!
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>
```

### 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.

The screenshot shows a code editor window with the file `Assignment-8.py` open. The code defines a function `classify_value` that returns "Negative" for negative numbers, "Zero" for zero, "Even" for even positive numbers, and "Odd" for odd positive numbers. It includes test cases for -5, 0, and "abc". The terminal below shows a command prompt in Windows, running Python 3.12, and executing the script. It catches a `TypeError` because it tries to compare a string ("abc") with an integer (0) using the less than operator (<). The error message points to line 18 of the script.

```
Assignment-8.py X
Wed.py > Assignment-8.py > ...

16
17 def classify_value(x):
18     if x < 0:
19         return "Negative"
20     elif x == 0:
21         return "Zero"
22     elif x%2==0:
23         return "Even"
24     else:
25         return "Odd"
26 # Test cases
27 assert classify_value(-5) == "Negative"
28 assert classify_value(0) == "zero"
29 assert classify_value("abc") == "Invalid input"

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

PS C:\Users\Ganne\OneDrive\Desktop\Ai Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/On
Traceback (most recent call last):
File "c:/Users/Ganne/OneDrive/Desktop/Ai Assisted Coding/Wed.py\Assignment-8.py", line 29, in <module>
    assert classify_value("abc") == "Invalid input"
    ^^^^^^^^^^^^^^
File "c:/Users/Ganne/OneDrive/Desktop/Ai Assisted Coding/Wed.py\Assignment-8.py", line 18, in classify_value
    if x < 0:
    ^
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\Ganne\OneDrive\Desktop\Ai Assisted Coding>
```

### 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
```

```
Assignment-8.py X D ▾
Wed.py > Assignment-8.py > ...
30
31     def is_palindrome(text):
32         cleaned_text = ''.join(char.lower() for char in
33             text if char.isalnum())
34
35     # Test cases
36     assert is_palindrome("Madam") == True
37     assert is_palindrome("A man a plan a canal Panama")
38     == True
39     assert is_palindrome("python") == False
40     print("All test cases for is_palindrome passed!")

PROBLEMS TERMINAL ... Python + ⌂ ⌁ ⌂ ⌁ ...
● PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Profiles/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/Ai_Assisted_Coding/Wed.py/Assignment-8.py"
All test cases for is_palindrome passed!
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>
```

#### 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.

Assignment-8.py X

Wed.py > Assignment-8.py > ...

```
40
41     def validate_email(email):
42         if email.count('@') != 1:
43             return False
44         at_index = email.index('@')
45         dot_index = email.rfind('.')
46         if at_index < 1 or dot_index < at_index + 2 or
47             dot_index >= len(email) - 1:
48             return False
49     return True
50
51 # Test cases
52 assert validate_email("user@example.com") == True
53 assert validate_email("invalid.email") == False
54 assert validate_email("user@domain") == False
55 assert validate_email("user@@domain.com") == False
56 print("All test cases for validate_email passed!")
```

PROBLEMS TERMINAL ...

Python + ▾

PS C:\Users\Ganne\OneDrive\Desktop\Ai\_Assisted\_Coding> & "C:/Profiles/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/Ai\_Assisted\_Coding/Wed.py/Assignment-8.py"

- All test cases for validate\_email passed!
- PS C:\Users\Ganne\OneDrive\Desktop\Ai\_Assisted\_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:
    - Normal case: 6 → True, 10 → False.
    - Edge case: 1.
    - Negative number case.
    - Larger case: 28.
  - Requirement: Validate correctness with assertions.

```

55
56     # generate a python code to display whether the given number is perfect or not.
57     def is_perfect_number(n):
58         if n < 1:
59             return False
60         sum_of_divisors = sum(i for i in range(1, n) if n % i == 0)
61         return sum_of_divisors == n
62     # Test cases
63     assert is_perfect_number(6) == True
64     assert is_perfect_number(28) == True
65     assert is_perfect_number(12) == False
66     assert is_perfect_number(0) == False
67     print("All test cases for is_perfect_number passed!")

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/AI Assisted Coding/is_perfect_number.py"
● All test cases for is_perfect_number passed!
○ PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding>

```

## 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

```

68
69     def Abundant_number(n):
70         if n < 1:
71             return False
72         sum_of_divisors = sum(i for i in range(1, n) if n % i == 0)
73         return sum_of_divisors > n
74     import unittest
75     class TestAbundantNumber(unittest.TestCase):
76         def test_abundant_number(self):
77             self.assertTrue(Abundant_number(12))
78             self.assertTrue(Abundant_number(15))
79             self.assertFalse(Abundant_number(1))
80             self.assertFalse(Abundant_number(-1))
81             self.assertFalse(Abundant_number(987))
82     if __name__ == '__main__':
83         unittest.main()

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/AI Assisted Coding/test_abundant_number.py"
F
-----
FAIL: test_abundant_number (__main__.TestAbundantNumber.test_abundant_number)

Traceback (most recent call last):
  File "c:/Users/Ganne/OneDrive/Desktop/AI Assisted Coding/wed.py/Assignment-8.py", line 78, in test_abundant_number
    self.assertTrue(Abundant_number(15))
AssertionError: False is not true

-----Ran 1 test in 0.001s

FAILED (failures=1)
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding>

```

## 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

```
Assignment-8.py X
Wed.py > Assignment-8.py > test_deficient_number_checker

1 def deficient_number_checker(n):
2     if n < 1:
3         return False
4     sum_of_divisors = sum(i for i in range(1, n) if n % i == 0)
5     return sum_of_divisors < n
6
7 def test_deficient_number_checker():
8     assert deficient_number_checker(1) == False
9     assert deficient_number_checker(2) == True
10    assert deficient_number_checker(3) == True
11    assert deficient_number_checker(4) == True
12    assert deficient_number_checker(5) == True
13    assert deficient_number_checker(6) == False
14    print("All test cases for deficient_number_checker passed!")

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL NOTES powerhell - weday + v

PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted - Coding\wed.py> python -m pytest Assignment-8.py
platform win32 -- Python 3.12.0b1, pytest-9.0.1, pluggy-1.6.0
rootdir: C:\Users\Ganne\OneDrive\Desktop\AI Assisted - Coding\wed.py
plugins: anyio-4.11.0
collected 1 item

Assignment-8.py F

FAILURES
test_deficient_number_checker
=====
>   def test_deficient_number_checker():
E     assert deficient_number_checker(1) == False
E     assert True == False
E     + where True = deficient_number_checker(1)

Assignment-8.py:92: AssertionError
short test summary info
FAILED Assignment-8.py::test_deficient_number_checker - assert True == False
z failed in 0.11s
```

## Task 8 :

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

```

Assignment-8.py X
Wed.py > Assignment-8.py > test_leap_year_checker

100 def LeapYearChecker(year):
101     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
102         return True
103     else:
104         return False
105 def test_leap_year_checker():
106     assert LeapYearChecker(2020) == True
107     assert LeapYearChecker(1900) == False
108     assert LeapYearChecker(2000) == True
109     assert LeapYearChecker(2021) == False
110     print("All test cases for LeapYearChecker passed!")

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/AI Assisted Coding/wed.py
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> cd wed.py
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py> python -m pytest Assignment-8.py
===== test session starts =====
platform win32 -- Python 3.12.10, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py
plugins: anyio-4.11.0
collected 1 item

Assignment-8.py . [100%]

===== 1 passed in 0.02s =====
% PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py>

```

### Task 9 :

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

```

111
112 def sum_of_digits(n):
113     return sum(int(digit) for digit in str(abs(n)) if digit.isdigit())
114 def test_sum_of_digits():
115     assert sum_of_digits(123) == 6
116     assert sum_of_digits(-456) == 15
117     assert sum_of_digits(0) == 0
118     assert sum_of_digits(78910) == 25
119     print("All test cases for sum_of_digits passed!")


```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/AI Assisted Coding/wed.py"
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> cd wed.py
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py> python -m pytest Assignment-8.py
===== test session starts =====
platform win32 -- Python 3.12.10, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py
plugins: anyio-4.11.0
collected 1 item

Assignment-8.py . [100%]

===== 1 passed in 0.02s =====
% PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py>

```

## Task 10 :

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

```
120
121     def sortNumbers(numbers) :
122         n = len(numbers)
123         for i in range(n):
124             for j in range(0, n-i-1):
125                 if numbers[j] > numbers[j+1] :
126                     numbers[j], numbers[j+1] = numbers[j+1], numbers[j]
127         return numbers
128     def test_sort_numbers():
129         assert sortNumbers([5, 2, 9, 1, 5, 6]) == [1, 2, 5, 5, 6, 9]
130         assert sortNumbers([]) == []
131         assert sortNumbers([3]) == [3]
132         assert sortNumbers([3, 2]) == [2, 3]
133         assert sortNumbers([1, 2, 3, 4, 5]) == [1, 2, 3, 4, 5]
134         assert sortNumbers([5, 4, 3, 2, 1]) == [1, 2, 3, 4, 8]
135         print("All test cases for sortNumbers passed!")
```

```
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py> python -m pytest Assignment-8.py
===== test session starts =====
platform win32 -- Python 3.12.10, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py
plugins: anyio-4.11.0
collected 1 item

Assignment-8.py F [100%]

def test_sort_numbers():
    assert sortNumbers([5, 2, 9, 1, 5, 6]) == [1, 2, 5, 5, 6, 9]
    assert sortNumbers([]) == []
def test_sort_numbers():
def test_sort_numbers():
    assert sortNumbers([5, 2, 9, 1, 5, 6]) == [1, 2, 5, 5, 6, 9]
    assert sortNumbers([]) == []
    assert sortNumbers([3]) == [3]
    assert sortNumbers([3, 2]) == [2, 3]
    assert sortNumbers([1]) == []
    assert sortNumbers([3]) == [3]
    assert sortNumbers([3, 2]) == [2, 3]
    assert sortNumbers([1, 2, 3, 4, 5]) == [1, 2, 3, 4, 5]
    assert sortNumbers([5, 4, 3, 2, 1]) == [1, 2, 3, 4, 8]
> E   assert [1, 2, 3, 4, 5] == [1, 2, 3, 4, 8]
E
E       At index 4 diff: 5 != 8
E       Use -v to get more diff

Assignment-8.py:134: AssertionError
short test summary info
FAILED Assignment-8.py::test_sort_numbers - assert [1, 2, 3, 4, 5] == [1, 2, 3, 4, 8]
    1 failed in 0.11s
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding\wed.py>
```

### Task 11 :

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

```
137     def Reverse_string(s):
138         return s[::-1]
139     import unittest
140     class TestReverseString(unittest.TestCase):
141         def test_reverse_string(self):
142             self.assertEqual(Reverse_string("hello"), "olleh")
143             self.assertEqual(Reverse_string("Python"), "nohtyP")
144             self.assertEqual(Reverse_string(""), "")
145             self.assertEqual(Reverse_string("a"), "a")
146     if __name__ == '__main__':
147         unittest.main()

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/0
● .

Ran 1 test in 0.000s

OK
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>
```

### Task 12 :

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

```
148
149     def Anagram_checker(str1, str2):
150         return sorted(str1.replace(" ", "").lower()) == sorted(str2.replace(" ", "").lower())
151     import unittest
152     class TestAnagramChecker(unittest.TestCase):
153         def test_anagram_checker(self):
154             self.assertTrue(Anagram_checker("listen", "silent"))
155             self.assertTrue(Anagram_checker("Triangle", "Integral"))
156             self.assertFalse(Anagram_checker("hello", "world"))
157             self.assertFalse(Anagram_checker("Python", "Java"))
158             self.assertTrue(Anagram_checker("Dormitory", "Dirty Room"))
159     if __name__ == '__main__':
160         unittest.main()

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/Ai
● .

Ran 1 test in 0.000s

OK
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>
```

### Task 13 :

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

```
162     def Armstrong_number(n):
163         num_str = str(n)
164         num_digits = len(num_str)
165         armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)
166         return armstrong_sum == n
167
168     import unittest
169     class TestArmstrongNumber(unittest.TestCase):
170         def test_armstrong_number(self):
171             self.assertTrue(Armstrong_number(153))
172             self.assertTrue(Armstrong_number(9474))
173             self.assertFalse(Armstrong_number(123))
174             self.assertFalse(Armstrong_number(0))
175             self.assertTrue(Armstrong_number(1))
176
177     if __name__ == '__main__':
178         unittest.main()

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/AI Assisted Coding/Med.py\Assignment-8.py"
❸ F
-----
FAIL: test_armstrong_number (__main__.TestArmstrongNumber.test_armstrong_number)
-----
Traceback (most recent call last):
  File "c:/Users/Ganne/OneDrive/Desktop/AI Assisted_Coding/Med.py\Assignment-8.py", line 173, in test_armstrong_number
    self.assertFalse(Armstrong_number(0))
AssertionError: True is not false

-----
Ran 1 test in 0.0001s
FAILED (failures=1)
❸ PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted_Coding>
```