

## 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.

```
18-02-26.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 for the is_valid_username function
11 assert is_valid_username("User123") == True
12 assert is_valid_username("12User") == False
13 assert is_valid_username("Us er") == False
14 print("All test cases for is_valid_username 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:/u
ng/18-02-26.py"
All test cases for is_valid_username passed!
PS C:\Users\SRAVANI\Documents\AI Assist 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.

```
18-02-26.py > ...
1 def classify_value(x):
2     if x < 0:
3         return "Negative"
4     elif x == 0:
5         return "Zero"
6     elif x%2 == 0:
7         return "Even"
8     else:
9         return "Odd"
10 #test cases for the classify_value function
11 assert classify_value(8)=="Even"
...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v

PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/p
Users/SRAVANI/Documents/AI Assist Coding/18-02-26.py"
Traceback (most recent call last):
  File "c:\Users\SRAVANI\Documents\AI Assist Coding\18-02-26.py", line 13, in <module>
    assert classify_value("abce")=="Invalid Input"
    ~~~~~^~~~~~
  File "c:\Users\SRAVANI\Documents\AI Assist Coding\18-02-26.py", line 2, in classify_value
    if x < 0:
        ^
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\SRAVANI\Documents\AI Assist 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:
  - 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
```

```
18-02-26.py > ...
1 def is_palindrome(text):
2     cleaned_text = ''.join(char.lower() for char in text if char.isalnum())
3     return cleaned_text == cleaned_text[::-1]
4 #test cases for the is_palindrome function
5 assert is_palindrome("Madam")==True
6 assert is_palindrome("A man a plan a canal panama")==True
7 assert is_palindrome("Python")==False
8 print("All test cases for is_palindrome passed!")

#PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [ ] [ ] ...
PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe
Users/SRAVANI/Documents/AI Assist Coding/18-02-26.py
All test cases for is_palindrome passed!
PS C:\Users\SRAVANI\Documents\AI Assist 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.

```
1 def validate_email(email):
2     if "@" not in email or "." not in email:
3         return False
4     at_index=email.index('@')
5     dot_index=email.rindex('.')
6     if at_index < 1 or dot_index < at_index + 2 or dot_index >= len(email) - 1:
7         return False
8     return True
9 #test cases for the validate_email function
10 assert validate_email("user@example.com")==True
11 assert validate_email("userexample.com")==False
12 assert validate_email("@gmail.com")==False
13 assert validate_email("user@com")==False
14 print("All test cases for validate_email passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

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.

```
18-02-26.py > _
1 def is_perfect_number(num):
2     if num < 1:
3         return False
4     divisors_sum=sum(i for i in range(1,num) if num % i == 0)
5     return divisors_sum == num
6 #test cases for the is_perfect_number function
7 assert is_perfect_number(6)==True
8 assert is_perfect_number(10)==False
9 assert is_perfect_number(28)==True
10 assert is_perfect_number(1)==False
11 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/SRAVANI/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:

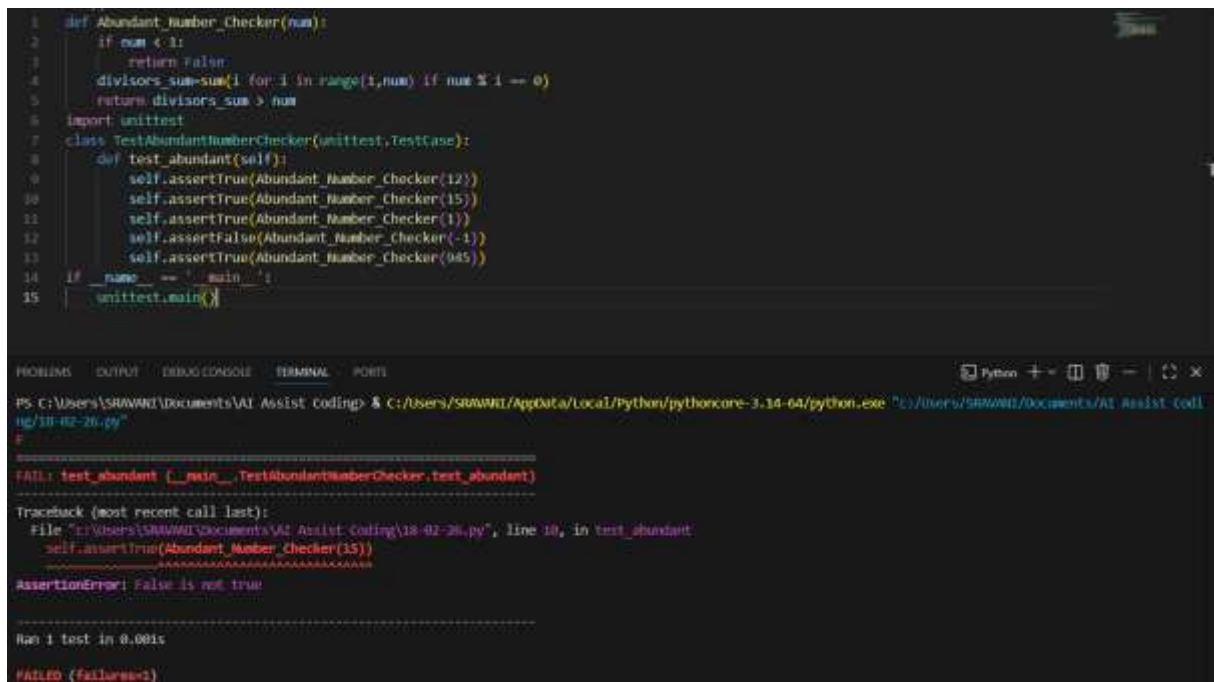
- o Normal case: 12 → True, 15 → False.

- o Edge case: 1.

- o Negative number case.

- o Large case: 945.

Requirement: Validate correctness with unittest



```
1 def Abundant_Number_Checker(num):
2     if num < 1:
3         return False
4     divisors_sum = sum(1 for i in range(1, num) if num % i == 0)
5     return divisors_sum > num
6
7 import unittest
8 class TestAbundantNumberChecker(unittest.TestCase):
9     def test_abundant(self):
10        self.assertTrue(Abundant_Number_Checker(12))
11        self.assertFalse(Abundant_Number_Checker(15))
12        self.assertTrue(Abundant_Number_Checker(1))
13        self.assertFalse(Abundant_Number_Checker(-1))
14        self.assertTrue(Abundant_Number_Checker(945))
15
16 if __name__ == '__main__':
17     unittest.main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [icon] [icon] [icon] [icon] [icon]

PS C:\Users\SRANWID\Documents\AI Assist coding> & C:\Users\SRANWID\AppData\Local\Python\pythoncore-3.14-64\python.exe "C:\Users\SRANWID\Documents\AI Assist coding\18-02-26.py"

FAIL: test\_abundant (\_\_main\_\_.TestAbundantNumberChecker.test\_abundant)

Traceback (most recent call last):

File "C:\Users\SRANWID\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,

- o 12 → False.

- o Edge case: 1.

- o Negative number case.

- o Large case: 546.

Requirement: Validate correctness with pytest

```
18-02-26.py > test_deficient_number_checker
1 def deficient_number_checker(num):
2     if num < 1:
3         return False
4     divisors_sum = sum(1 for i in range(1, num) if num % i == 0)
5     return divisors_sum < num
6
7 def test_deficient_number_checker():
8     assert deficient_number_checker(0) == True
9     assert deficient_number_checker(12) == False
10    assert deficient_number_checker(1) == True
11    assert deficient_number_checker(546) == False
12    print("All test cases for deficient_number_checker passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\SRANWI\Documents\AI Assist Coding> & C:\Users\SRANWI\AppData\Local\Python\pythoncore-3.14-64\python.exe "C:\Users\SRANWI\Documents\AI Assist Coding\18-02-26.py"

PS C:\Users\SRANWI\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\SRANWI\Documents\AI Assist Coding

collected 1 item

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

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

PS C:\Users\SRANWI\Documents\AI Assist Coding>

Task 8 :

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

```
18-02-26.py > test_LeapYearChecker
1 def LeapYearChecker(year):
2     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
3         return True
4     return False
5
6 def test_LeapYearChecker():
7     assert LeapYearChecker(2020) == True
8     assert LeapYearChecker(1900) == False
9     assert LeapYearChecker(2000) == True
10    assert LeapYearChecker(2021) == False
11    print("All test cases for LeapYearChecker passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\SRANWI\Documents\AI Assist Coding> & C:\Users\SRANWI\AppData\Local\Python\pythoncore-3.14-64\python.exe "C:\Users\SRANWI\Documents\AI Assist Coding\18-02-26.py"

PS C:\Users\SRANWI\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\SRANWI\Documents\AI Assist Coding

collected 1 item

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

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

PS C:\Users\SRANWI\Documents\AI Assist Coding>

Task 9 :

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



```
> 18-02-26.py > test_sum_of_digits
```

```
1 def SumOfDigits(num):  
2     return sum(int(digit) for digit in str(abs(num)))  
3  
4 def test_sum_of_digits():  
5     assert SumOfDigits(123)==6  
6     assert SumOfDigits(-456)==15  
7     assert SumOfDigits(0)==0  
8     assert SumOfDigits(9999)==36  
9     assert SumOfDigits(-12345)==15  
10    print("All test cases for SumOfDigits passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] ... |

```
PS C:\Users\SRAWANI\Documents\AI Assist Coding> && C:/Users/SRAWANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAWANI/Documents/AI Assist Coding/18-02-26.py"  
PS C:\Users\SRAWANI\Documents\AI Assist Coding> python -- 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\SRAWANI\Documents\AI Assist Coding  
collected 1 item  
  
18-02-26.py .  
  
===== 1 passed in 0.01s =====  
PS C:\Users\SRAWANI\Documents\AI Assist Coding>
```

### Task 10 :

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

```
1 def SortNumbers(numbers):  
2     n=len(numbers)  
3     for i in range(n):  
4         for j in range(0,n-i-1):  
5             if numbers[j] > numbers[j+1]:  
6                 numbers[j],numbers[j+1]=numbers[j+1],numbers[j]  
7     return numbers  
8  
9 def test_SortNumbers():  
10    assert SortNumbers([5,2,9,1,5,6])==[1,2,5,5,6,9]  
11    assert SortNumbers([])==[]  
12    assert SortNumbers([3])==[3]  
13    assert SortNumbers([3,2])==[2,3]  
14    assert SortNumbers([10,9,8,7])==[7,8,9,10]  
15    assert SortNumbers([-1,-3,0,2])==[-3,-1,0,2]  
16    assert SortNumbers([1,2,3,4,5])==[1,2,3,4,5]  
17    assert SortNumbers([5,4,3,2,1])==[1,2,3,4,5]  
18    assert SortNumbers([1,1,1,1])==[1,1,1,1]  
19    assert SortNumbers([3,1,2,3,1])==[1,1,2,3,3]  
20    assert SortNumbers([0,0,0,0])==[0,0,0,0]  
21    assert SortNumbers([1,2,3,4,5,6,7,8,9,10])==[1,2,3,4,5,6,7,8,9,10]  
22    print("All test cases for SortNumbers passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [x]

```
PS C:\Users\SANWU\Documents\AI Assist Codings> . C:\Users\SANWU\AppData\Local\Python\pythoncore-3.14-64/python.exe "C:\Users\SANWU\Documents\AI Assist Codings\18-02-26.py"  
PS C:\Users\SANWU\Documents\AI Assist Codings> 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\SANWU\Documents\AI Assist coding  
collected 1 item  
  
18-02-26.py .  
  
1 passed in 0.01s
```



### Task 11 :

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

```
1 def ReverseString(s):
2     return s[::-1]
3
4 import unittest
5
6 class TestReverseString(unittest.TestCase):
7     def test_reverse_string(self):
8         self.assertEqual(ReverseString("Hello"), "olleH")
9         self.assertEqual(ReverseString("Python"), "nohtyP")
10        self.assertEqual(ReverseString(""), "")
11        self.assertEqual(ReverseString("A"), "A")
12        self.assertEqual(ReverseString("12345"), "54321")
13
14 if __name__ == '__main__':
15     unittest.main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + -

```
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"
*
.....
Ran 1 test in 0.000s

OK Focus folder in explorer (ctrl + click)
PS C:\Users\SRAVANI\Documents\AI_Assist coding>
```

### Task 12 :

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

```
PS C:\Users\SRUJANI> TestAnagramChecker > test_anagram_checker
1 def AnagramChecker(str1, str2):
2     clean_str1 = str1.replace(" ", "").lower()
3     clean_str2 = str2.replace(" ", "").lower()
4     return sorted(clean_str1) == sorted(clean_str2)
5
6 import unittest
7
8 class TestAnagramChecker(unittest.TestCase):
9     def test_anagram_checker(self):
10         self.assertTrue(AnagramChecker("listen", "silent"))
11         self.assertTrue(AnagramChecker("triangle", "Integral"))
12         self.assertFalse(AnagramChecker("Hello", "World"))
13         self.assertTrue(AnagramChecker("Dormitory", "Dirty Room"))
14         self.assertFalse(AnagramChecker("Python", "Java"))
15         self.assertTrue(AnagramChecker("state", "taste"))
16         self.assertTrue(AnagramChecker("Conversation", "Voices Rant On")) # corrected
17         self.assertTrue(AnagramChecker("School master", "The classroom"))
18
19 if __name__ == '__main__':
20     unittest.main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [ ] [ ] [ ] [ ] [ ]

```
PS C:\Users\SRUJANI\Documents\AI Assist Coding> C:\Users\SRUJANI\AppData\Local\Python\pythoncore-3.14-64/python.exe "C:\Users\SRUJANI\Documents\AI Assist Coding\18-02-26.py"
-----
Run 1 test in 0.000s
OK
PS C:\Users\SRUJANI\Documents\AI Assist Coding>
```

### Task 13 :

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

```

18-02-26.py > % TestArmstrongNumberChecker > test_armstrong_number_checker
1 def ArmstrongNumberChecker(num):
2     num_str=str(num)
3     num_digits=len(num_str)
4     armstrong_sum=sum(int(digit)**num_digits for digit in num_str)
5     return armstrong_sum == num
6
7 import unittest
8 class TestArmstrongNumberChecker(unittest.TestCase):
9     def test_armstrong_number_checker(self):
10         self.assertTrue(ArmstrongNumberChecker(153))
11         self.assertTrue(ArmstrongNumberChecker(370))
12         self.assertTrue(ArmstrongNumberChecker(371))
13         self.assertTrue(ArmstrongNumberChecker(407))
14         self.assertFalse(ArmstrongNumberChecker(123))
15         self.assertFalse(ArmstrongNumberChecker(0))
16         self.assertFalse(ArmstrongNumberChecker(-153))
17
18 if __name__ == '__main__':
19     unittest.main()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\SRWANU\Documents\AI Assist Coding> & C:\Users\SRWANU\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:\Users\SRWANU\Documents\AI Assist Coding\18-02-26.py"

FAIL: test\_armstrong\_number\_checker (\_\_main\_\_.TestArmstrongNumberChecker.test\_armstrong\_number\_checker)

Traceback (most recent call last):

File "c:\Users\SRWANU\Documents\AI Assist Coding\18-02-26.py", line 14, in test\_armstrong\_number\_checker

self.assertFalse(ArmstrongNumberChecker(0))

AssertionError: True is not false

Run 1 test in 0.001s