

ASSIGNMENT - 8

**NAME : R.RONITH REDDY
HTNO : 2303A52280**

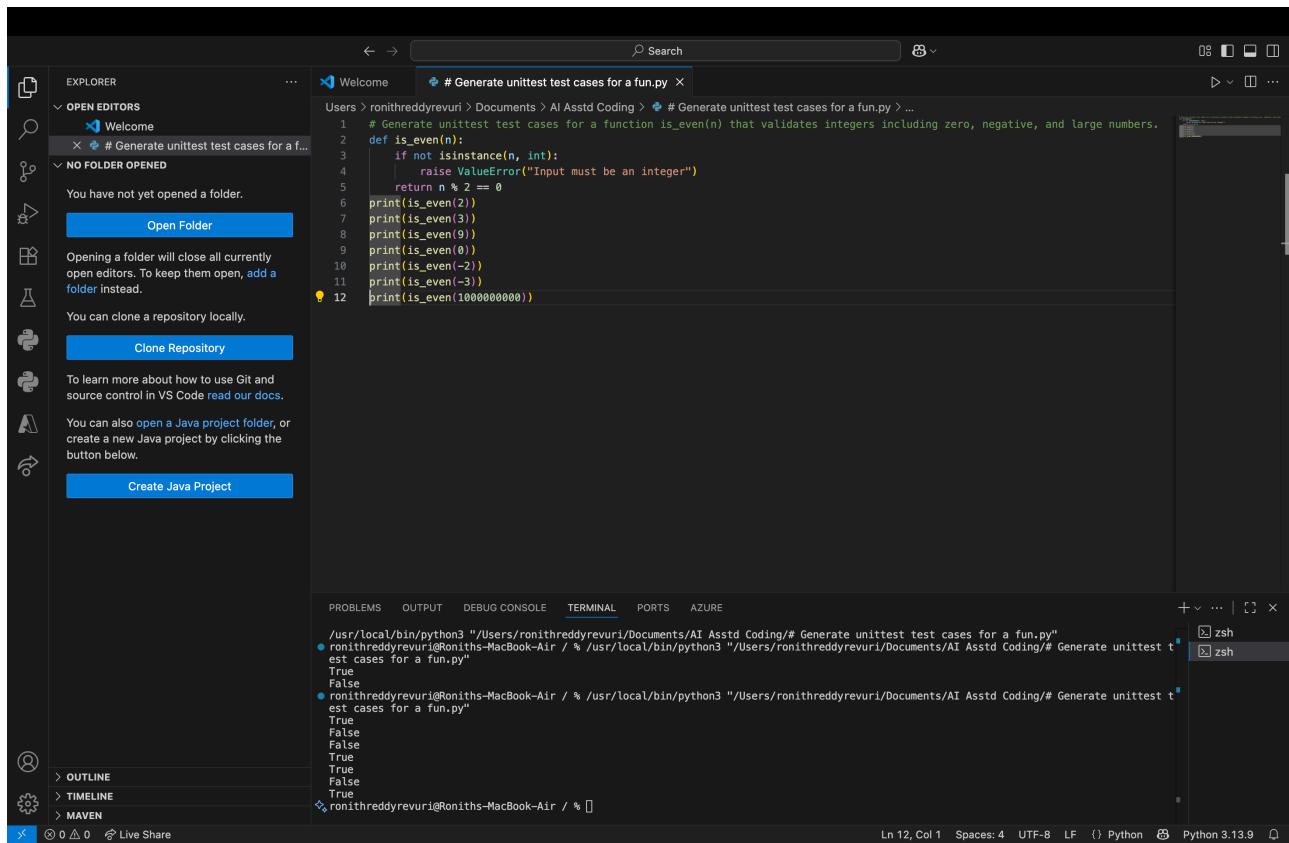
TASK 1 – TEST-DRIVEN DEVELOPMENT FOR EVEN/ODD NUMBER VALIDATOR

PROMPT USED :

GENERATE UNITTEST TEST CASES FOR A FUNCTION IS_EVEN(N) THAT VALIDATES INTEGERS INCLUDING ZERO, NEGATIVE, AND LARGE NUMBERS.

EXPLANATION :

THE FUNCTION USES THE MODULUS OPERATOR TO DETERMINE IF A NUMBER IS EVEN OR ODD. IN ORDER TO KEEP THE PROGRAM SAFE, IT ALSO MAKES SURE THE INPUT IS AN INTEGER.



TASK - 2 :

TEST-DRIVEN DEVELOPMENT FOR STRING CASE CONVERTER

PROMPT:

GENERATE AI TO GENERATE TEST CASES FOR TWO FUNCTIONS:

TO_UPPERCASE(TEXT)

TO LOWERCASE(TEXT)

EXPLANATION :

TEXT CAN BE CONVERTED TO LOWERCASE OR UPPERCASE USING THESE FUNCTIONS.

TEXT CAN BE CONVERTED TO LOWERCASE OR UPPERCASE USING THESE FUNCTIONS.
ADDITIONALLY, THEY SAFELY HANDLE INCORRECT OR EMPTY INPUTS AND VERIFY THAT THE
INPUT IS VALID.

The screenshot shows the Visual Studio Code (VS Code) interface. The left sidebar has sections for 'OPEN EDITORS' (Welcome), 'NO FOLDER OPENED', and buttons for 'Open Folder', 'Clone Repository', and 'Create Java Project'. The main area displays a Python file named 'fun.py' with the following code:

```
1 # Generate AI to generate test cases for two functions:
2 # to_uppercase(text)
3 # to_lowercase(text)
4
5 def to_uppercase(text):
6     return text.upper()
7
8 def to_lowercase(text):
9     return text.lower()
10 print(to_uppercase("Hello world"))
11 print(to_lowercase("HELLO WORLD"))
```

The terminal at the bottom shows the execution of the script:

```
True
False
● ronithreddyrevuri@Roniths-MacBook-Air / % /usr/local/bin/python3 "/Users/ronithreddyrevuri/Documents/AI Asstd Coding/# Generate unittest t
est cases for a fun.py"
True
False
False
True
True
False
True
● ronithreddyrevuri@Roniths-MacBook-Air / % /usr/local/bin/python3 "/Users/ronithreddyrevuri/Documents/AI Asstd Coding/# Generate unittest t
est cases for a fun.py"
HELLO WORLD
hello world
● ronithreddyrevuri@Roniths-MacBook-Air / %
```

Details from the terminal status bar: Ln 11, Col 35, Spaces: 4, UTF-8, LF, Python, Python 3.13.9.

TASK 3 – TEST-DRIVEN DEVELOPMENT FOR LIST SUM CALCULATOR

PROMPT :

PYTHON CODE TO GENERATE TEST CASES FOR A FUNCTION SUM_LIST(NUMBERS) THAT CALCULATES THE SUM OF LIST ELEMENTS.

EXPLANATION :

ONLY NUMERIC VALUES FROM THE LIST ARE ADDED BY THE FUNCTION. IT HANDLES EMPTY LISTS SAFELY AND DISREGARDS NON-NUMERIC ELEMENTS.

The screenshot shows the Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar with sections for 'OPEN EDITORS' (containing 'Welcome' and '# Generate unittest test cases for a function sum_list(numbers)'), 'NO FOLDER OPENED' (with a note about opening a folder), and 'Clone Repository' and 'Create Java Project' buttons. The main area is a code editor with the following Python code:

```
# Python code generate test cases for a function sum_list(numbers) that calculates the sum of list elements.
def sum_list(numbers):
    if not isinstance(numbers, list):
        raise ValueError("Input must be a list")
    return sum(numbers)

print(sum_list([1, 2, 3]))
print(sum_list([-1, 1, 0]))
print(sum_list([1.5, 2.5, 3.5]))
print(sum_list([]))
```

Below the code editor is a 'TERMINAL' tab showing command-line output from a zsh shell:

```
False
True
True
False
True
● ronithreddyrevuri@Roniths-MacBook-Air ~ % /usr/local/bin/python3 "/Users/ronithreddyrevuri/Documents/AI Asstd Coding/# Generate unittest t
est cases for a fun.py"
HELLO WORLD
hello world
● ronithreddyrevuri@Roniths-MacBook-Air ~ % /usr/local/bin/python3 "/Users/ronithreddyrevuri/Documents/AI Asstd Coding/# Generate unittest t
est cases for a fun.py"
6
0
7.5
0
● ronithreddyrevuri@Roniths-MacBook-Air ~ %
```

The status bar at the bottom indicates 'Ln 10, Col 22' and 'Python 3.13.9'.

TASK 4 – TEST CASES FOR STUDENT RESULT CLASS

PROMPT :

GENERATE TEST CASES FOR A STUDENTRESULT CLASS WITH METHODS

ADD_MARKS(MARK), CALCULATE_AVERAGE(), AND GET_RESULT() INCLUDING VALIDATION RULES.

EXPLANATION :

IN ORDER TO DETERMINE WHETHER A STUDENT PASSES OR FAILS, THE CLASS KEEPS TRACK OF THEIR GRADES AND COMPUTES THEIR AVERAGE. ADDITIONALLY, IT CONFIRMS THAT THE MARKS FALL WITHIN THE APPROPRIATE RANGE.

The screenshot shows the Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar with sections for OPEN EDITORS (Welcome, # Generate unittest test cases for a fun.py), NO FOLDER OPENED, and various project-related buttons like Open Folder, Clone Repository, and Create Java Project. The main area displays a Python script named `# Generate unittest test cases for a fun.py`. The script defines a `StudentResult` class with methods for adding marks, calculating average, and getting results. It also includes a `generate_test_cases` function. The terminal at the bottom shows the execution of the script and its output, including a stack trace for a `NameError` related to the `StudentResult` class. The status bar at the bottom right indicates the file is 61 lines long, 1 column wide, and uses Python 3.13.9.

```
1 # Generate test cases for a StudentResult class with methods add_marks(mark), calculate_average(), and get_result() including validation logic.
2 # StudentResult Class Definition
3 class StudentResult:
4     def __init__(self):
5         self.marks = []
6
7     def add_marks(self, mark):
8         if not (0 <= mark <= 100):
9             raise ValueError("Marks must be between 0 and 100")
10            self.marks.append(mark)
11
12    def calculate_average(self):
13        if len(self.marks) == 0:
14            return 0
15        return sum(self.marks) / len(self.marks)
16
17    def get_result(self):
18        if self.calculate_average() >= 40:
19            return "Pass"
20        else:
21            return "Fail"
22
23
24 # Test Case Generator Function
25 def generate_test_cases():
26     test_cases = []
27
28     # Test case 1: Pass Case
29     student1 = StudentResult()
30     student1.add_marks(60)
```

TASK - 5 : TEST-DRIVEN DEVELOPMENT FOR USERNAME VALIDATOR

PROMPT :

GENERATE A PYTHON CODE FOR USERNAME VALIDATION

#MINIMUM LENGTH: 5 CHARACTERS

#NO SPACES ALLOWED

#ONLY ALPHANUMERIC CHARACTERS

EXPLANATION:

THE FUNCTION VALIDATES A USERNAME BY CHECKING ITS LENGTH, ENSURING THERE ARE NO SPACES, AND ALLOWING ONLY ALPHANUMERIC CHARACTERS.

The screenshot shows the Visual Studio Code (VS Code) interface. The left sidebar has icons for file operations, search, and repository management. The main area displays a Python script named `fun.py` which defines a function to validate a username based on length, spaces, and alphanumeric characters.

```
1 # Generate a Python code for Username Validation
2 #Minimum length: 5 characters
3 #No spaces allowed
4 #Only alphanumeric characters
5 def validate_username(username):
6     if len(username) < 5:
7         return False
8     if " " in username:
9         return False
10    if not username.isalnum():
11        return False
12    return True
13 print(validate_username("user1"))
14 print(validate_username("us"))
15 print(validate_username("user 1"))
16 print(validate_username("user@1"))
```

The terminal at the bottom shows the execution of the script and its output:

```
est cases for a fun.py"
[{"description": "Marks: [60,70,80]", "expected_average": 70, "actual_average": 70.0, "expected_result": "Pass", "actual_result": "Pass"}, {"description": "Marks: [30,35,40]", "expected_average": 35, "actual_average": 35.0, "expected_result": "Fail", "actual_result": "Fail"}]
● ronithreddyrevuri@Roniths-MacBook-Air / % /usr/local/bin/python3 "/Users/ronithreddyrevuri/Documents/AI Asstd Coding/# Generate unittest test cases for a fun.py"
True
False
False
False
● ronithreddyrevuri@Roniths-MacBook-Air / %
```

At the bottom, status bar details include: Ln 13, Col 36, Spaces: 4, UTF-8, LF, Python, Python 3.13.9.

CONCLUSION :

THIS LAB HELPED IN UNDERSTANDING TEST-DRIVEN DEVELOPMENT USING AI TOOLS. BY WRITING TEST CASES FIRST AND THEN IMPLEMENTING THE CODE, WE ENSURED CORRECTNESS, VALIDATION, AND RELIABILITY. IT IMPROVED OUR UNDERSTANDING OF TESTING AND WRITING CLEAN, ERROR-FREE PROGRAMS.