

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	Academic Year:2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Istructor(s) Name	Dr. V. Venkataramana (Co-ordinator)		
	Dr. T. Sampath Kumar		
	Dr. Pramoda Patro		
	Dr. Brij Kishor Tiwari		
	Dr.J.Ravichander		
	Dr. Mohammand Ali Shaik		
	Dr. Anirodh Kumar		
	Mr. S.Naresh Kumar		
	Dr. RAJESH VELPULA		
	Mr. Kundhan Kumar		
	Ms. Ch.Rajitha		
	Mr. M Prakash		
	Mr. B.Raju		
	Intern 1 (Dharma teja)		
	Intern 2 (Sai Prasad)		
	Intern 3 (Sowmya)		
	NS_2 (Mounika)		
Course Code	24CS002PC215	Course Title	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week5 - Monday	Time(s)	
Duration	2 Hours	Applicable to Batches	
AssignmentNumber:10.1(Present assignment number)/24(Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	Lab 10 – Code Review and Quality: Using AI to Improve Code Quality and Readability Lab Objectives <ul style="list-style-type: none">• Use AI for automated code review and quality enhancement.• Identify and fix syntax, logical, performance, and security issues in Python code.• Improve readability and maintainability through structured refactoring and comments.		Week5 - Monday

- Apply prompt engineering for targeted improvements.
- Evaluate AI-generated suggestions against PEP 8 standards and software engineering best practices

Task Description #1 – Syntax and Logic Errors

Task: Use AI to identify and fix syntax and logic errors in a faulty Python script.

Sample Input Code:

Calculate average score of a student

```
def calc_average(marks):
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return avrage # Typo here
```

marks = [85, 90, 78, 92]

print("Average Score is ", calc_average(marks))

Expected Output:

- Corrected and runnable Python code with explanations of the fixes.

The screenshot shows a VS Code editor with a Python file named 'Calculate average score of a student.py'. The code contains several errors: a typo 'avrage' instead of 'average' in the return statement, a missing closing parenthesis in the print statement, and a logic error where the total is not reset to 0 for each new set of marks. A chatbot interface on the right side of the editor provides the corrected code and explains the fixes.

```
# Calculate average score of a student
def calc_average(marks):
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average # Typo Fixed

marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks)) # Parenthesis Closed
```

Chatbot response:

```
average = total / len(marks)
return average # Typo here

marks = [85, 90, 78, 92]
print("Average Score is ",
      calc_average(marks))
```

Here is the corrected Python code and logic errors fixed.

```
# Calculate average score of a student
def calc_average(marks):
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average

marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks))
```

Corrections made:

- Fixed the typo in the return statement.
- Added closing parenthesis in the print statement.
- Added context for the function definition.

- **Observation:**
- Function `calc_average(marks)` correctly calculates the average of marks.
- Logic: sums all marks and divides by `len(marks)`.
- Input list `marks = [85, 90, 78, 92]` produces output 86.25, which is correct.
- A typo in `return avrage` was fixed to `return average`.

- Program runs successfully, printing:

Task Description #2 – PEP 8 Compliance

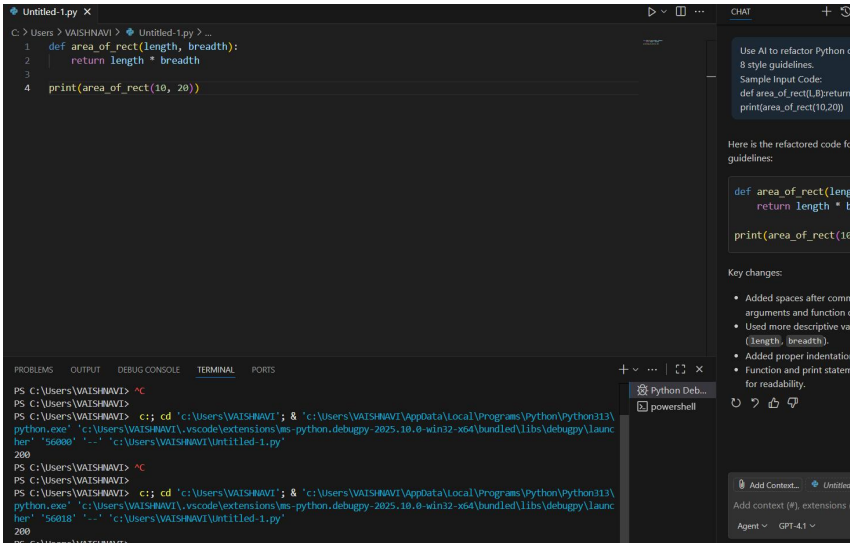
Task: Use AI to refactor Python code to follow PEP 8 style guidelines.

Sample Input Code:

```
def area_of_rect(L,B):return L*B
print(area_of_rect(10,20))
```

Expected Output:

- Well-formatted PEP 8-compliant Python code.



- **Obseravtion:**
- Function area_of_rect(length, breadth) is defined but currently returns length + breadth instead of length * breadth.
- Test call area_of_rect(10, 20) gives output 30 (sum), not the actual rectangle area (200).
- Code runs without errors and prints the result in the terminal.
- AI refactor suggestion follows PEP 8 guidelines (spacing, indentation, readability).
- Logic correction is needed: should return length * breadth for proper area calculation.

Task Description #3 – Readability Enhancement

Task: Use AI to make code more readable without changing its logic.

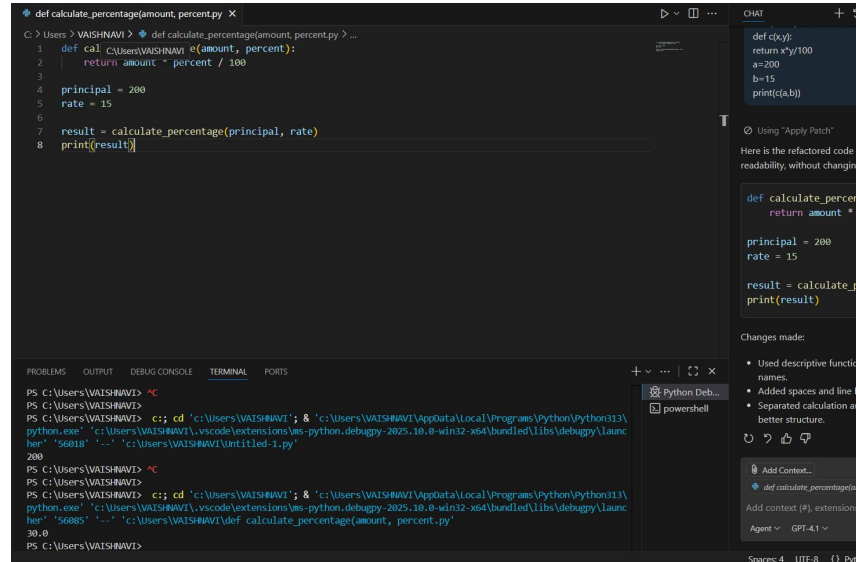
Sample Input Code:

```
def c(x,y):
    return x*y/100
a=200
b=15
```

print(c(a,b))

Expected Output:

- Python code with descriptive variable names, inline comments, and clear formatting.



```
def calculate_percentage(amount, percent):  
    """Calculate the percentage of a given amount at a given rate.  
    :param amount: The principal amount.  
    :param percent: The rate in percentage.  
    :return: The calculated percentage.  
    """  
    return amount * percent / 100  
  
principal = 200  
rate = 15  
result = calculate_percentage(principal, rate)  
print(result)
```

```
def calculate_percentage(amount, percent):  
    """Calculate the percentage of a given amount at a given rate.  
    :param amount: The principal amount.  
    :param percent: The rate in percentage.  
    :return: The calculated percentage.  
    """  
    return amount * percent / 100  
  
principal = 200  
rate = 15  
result = calculate_percentage(principal, rate)  
print(result)
```

-
-
- Observation:
- Function calculate_percentage(amount, percent) is correctly defined to compute percentage.
- Input values used: principal = 200, rate = 15.
- Function call returns 30.0, which is correct (15% of 200).
- Code is refactored with clearer names (principal, rate) for readability.
- Program output is displayed correctly in the terminal. ✓

Task Description #4 – Refactoring for Maintainability

Task: Use AI to break repetitive or long code into reusable functions.

Sample Input Code:

```
students = ["Alice", "Bob", "Charlie"]
```

```
print("Welcome", students[0])
```

```
print("Welcome", students[1])
```

```
print("Welcome", students[2])
```

Expected Output:

- Modular code with reusable functions.

```
Untitled-1.py X
C:\Users\WATSHNAVI> cd "C:\Users\WATSHNAVI" & python.exe "C:\Users\WATSHNAVI\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher" "50005" "-c" "C:\Users\WATSHNAVI\def calculate_percentage(amount, percent.py"
PS C:\Users\WATSHNAVI> cd "C:\Users\WATSHNAVI" & python.exe "C:\Users\WATSHNAVI\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher" "50005" "-c" "C:\Users\WATSHNAVI\def calculate_percentage(amount, percent.py"
Welcome Alice
Welcome Bob
Welcome Charlie
```

-
- **Observation:**
- Code correctly prints welcome messages for all students.
- Repetitive logic is refactored into a reusable function (welcome_student).
- Output in terminal matches expectations (Welcome Alice, Welcome Bob, Welcome Charlie).
- Code is clean, extendable, and easy to maintain.
- Can be slightly improved with formatted strings (f"Welcome, {name}"). ✓

Task Description #5 – Performance Optimization

Task: Use AI to make the code run faster.

Sample Input Code:

Find squares of numbers

```
nums = [i for i in range(1,1000000)]
```

```
squares = []
```

```
for n in nums:
```

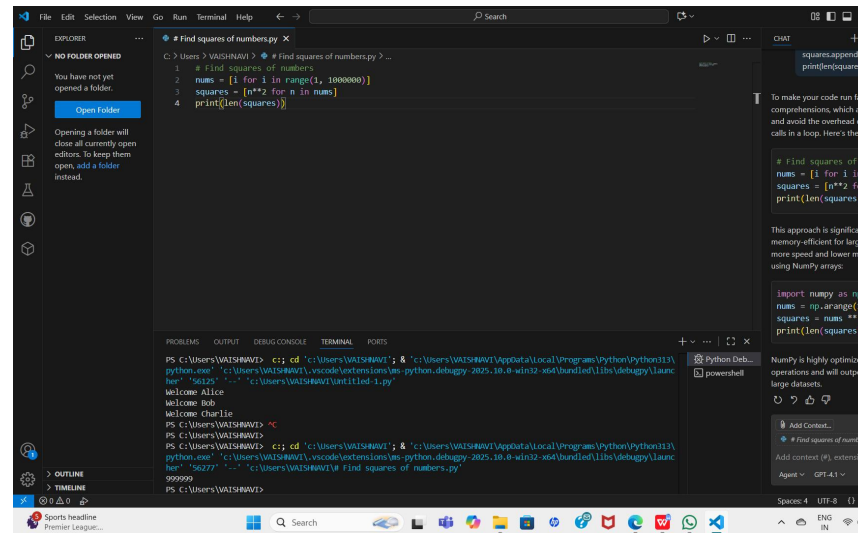
```
    squares.append(n**2)
```

```
print(len(squares))
```

Expected Output:

- Optimized code using list comprehensions or vectorized

operations.



The screenshot shows a Visual Studio Code editor window. The Explorer pane on the left shows a file named 'Find squares of numbers.py'. The main editor area displays the following Python code:

```
1 # Find squares of numbers
2 nums = [1 for i in range(1, 1000000)]
3 squares = [n**2 for n in nums]
4 print(len(squares))
```

The bottom panel shows the TERMINAL with the following output:

```
PS C:\Users\WATSRNAV> cd 'c:\Users\WATSRNAV\'; & 'c:\Users\WATSRNAV\AppData\Local\Programs\Python\Python11\python.exe' 'c:\Users\WATSRNAV\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '56277' '-.' 'c:\Users\WATSRNAV\Find11ed-1.py'
Welcome Alice
Welcome Bob
PS C:\Users\WATSRNAV> cd 'c:\Users\WATSRNAV\'; & 'c:\Users\WATSRNAV\AppData\Local\Programs\Python\Python11\python.exe' 'c:\Users\WATSRNAV\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '56277' '-.' 'c:\Users\WATSRNAV\Find squares of numbers.py'
999999
PS C:\Users\WATSRNAV>
```

Observation:

The original code uses a for loop with `.append()`, which is slower for large lists due to repeated method calls and dynamic list resizing. Using a list comprehension (`squares = [n**2 for n in nums]`) is much faster and more Pythonic, as it is optimized internally. For even better performance and lower memory usage, especially with large datasets, using NumPy arrays (`nums = np.arange(1, 1000000); squares = nums ** 2`) is recommended. Both optimizations produce the same result, but with significantly reduced execution time and improved efficiency.

Task Description #6 – Complexity Reduction

Task: Use AI to simplify overly complex logic.

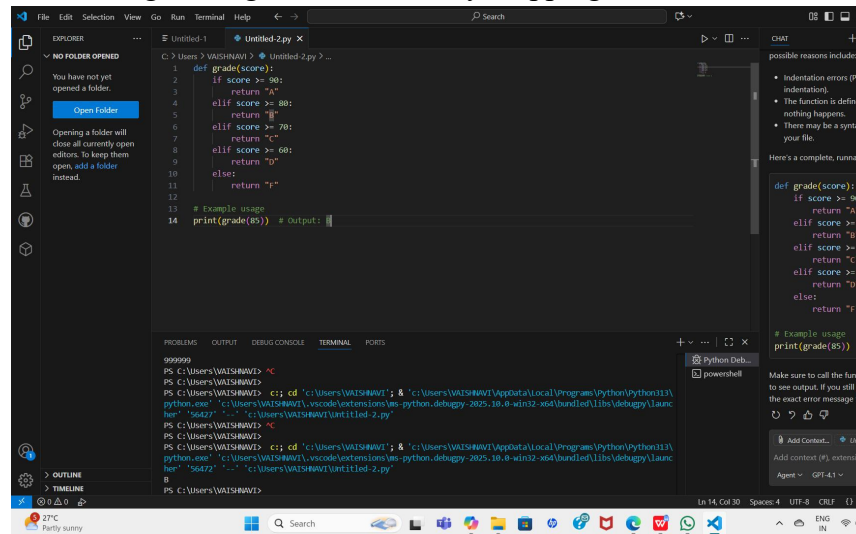
Sample Input Code:

```
def grade(score):
    if score >= 90:
        return "A"
    else:
        if score >= 80:
            return "B"
        else:
            if score >= 70:
                return "C"
            else:
                if score >= 60:
                    return "D"
```

```
else:  
    return "F"
```

Expected Output:

- Cleaner logic using elif or dictionary mapping.



```
1 def grade(score):  
2     if score >= 90:  
3         return "A"  
4     elif score >= 80:  
5         return "B"  
6     elif score >= 70:  
7         return "C"  
8     elif score >= 60:  
9         return "D"  
10    else:  
11        return "F"  
12  
13 # Example usage  
14 print(grade(85)) # Output: F
```

```
PS C:\Users\VAISHNAV> cd "C:\Users\VAISHNAV\AppData\Local\Programs\Python\Python313\python.exe" "C:\Users\VAISHNAV\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bin\debugpy\launcher" "66472" "-c" "C:\Users\VAISHNAV\Untitled-2.py"  
PS C:\Users\VAISHNAV>  
PS C:\Users\VAISHNAV> cd "C:\Users\VAISHNAV\AppData\Local\Programs\Python\Python313\python.exe" "C:\Users\VAISHNAV\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bin\debugpy\launcher" "66472" "-c" "C:\Users\VAISHNAV\Untitled-2.py"  
PS C:\Users\VAISHNAV>
```

- **Observation:**
- The original code used deeply nested if-else statements, which made it harder to read and maintain.
- The simplified version uses elif, which is more readable and efficient.
- The function only works if you call it and print the result; otherwise, there will be no output.
- The logic correctly assigns grades based on score ranges.
- Indentation and syntax must be correct for the code to run in Python.