| | |
|---|---|
| **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 |
| **Instructor(s) Name** | Dr. V. Venkataramana (Co-ordinator)<br>Dr. T. Sampath Kumar<br>Dr. Pramoda Patro<br>Dr. Brij Kishor Tiwari<br>Dr.J.Ravichander<br>Dr. Mohammand Ali Shaik<br>Dr. Anirodh Kumar<br>Mr. S.Naresh Kumar<br>Dr. RAJESH VELPULA<br>Mr. Kundhan Kumar<br>Ms. Ch.Rajitha<br>Mr. M Prakash<br>Mr. B.Raju<br>Intern 1 (Dharma teja)<br>Intern 2 (Sai Prasad)<br>Intern 3 (Sowmya)<br>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**<br>**Lab Objectives**<br>• Use AI for automated code review and quality enhancement.<br>• Identify and fix syntax, logical, performance, and security issues in Python code.<br>• 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.

**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.

**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.

**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.

**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.

**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:<br>    if score >= 70:<br>        return "C"<br>    else:<br>        if score >= 60:<br>            return "D"<br>        else:<br>            return "F"<br>Expected Output:<br>• Cleaner logic using elif or dictionary mapping. |  |

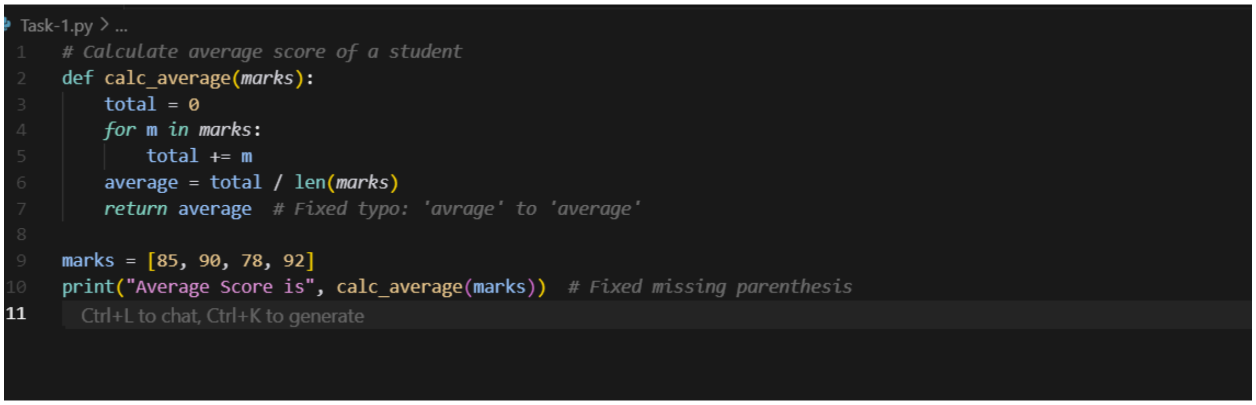
TASK-1:

PROMPT:

```
# 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)
```

identify and fix the syntax and logical errors.

CODE:

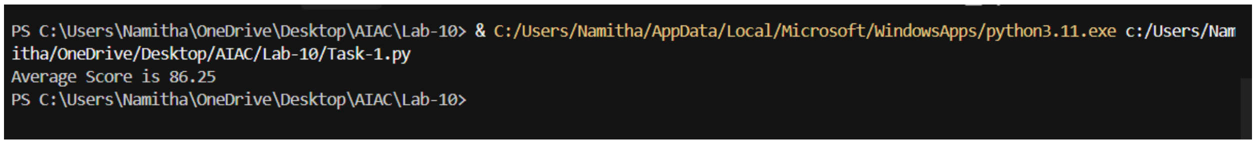


Task-1.py > ...

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

marks = [85, 90, 78, 92]
print("Average Score is", calc_average(marks))  # Fixed missing parenthesis

```

Ctrl+L to chat, Ctrl+K to generate

OUTPUT:



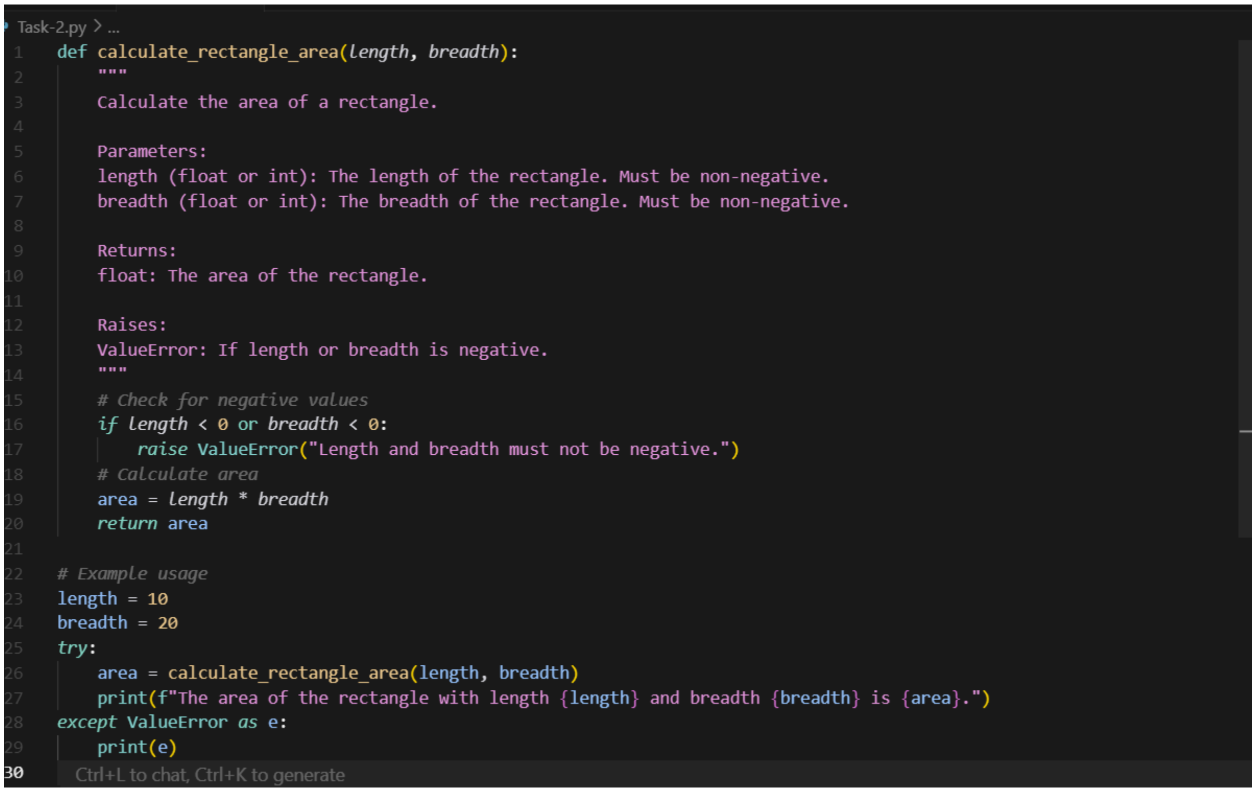
```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10> & C:/Users/Namitha/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Nam
itha/OneDrive/Desktop/AIAC/Lab-10/Task-1.py
Average Score is 86.25
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10>
```

TASK-2:

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

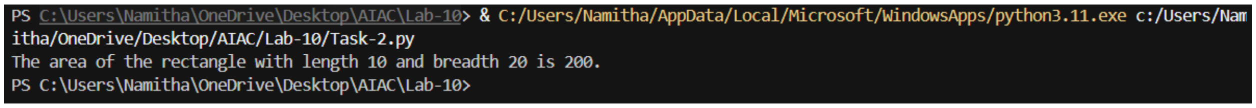
enhance the code by keepind docstring ,comments and proper function name parameters name and proper print statement..length and breadthe should not be negative.

CODE:



```python
def calculate_rectangle_area(length, breadth):
    """
    Calculate the area of a rectangle.

    Parameters:
    length (float or int): The length of the rectangle. Must be non-negative.
    breadth (float or int): The breadth of the rectangle. Must be non-negative.

    Returns:
    float: The area of the rectangle.

    Raises:
    ValueError: If length or breadth is negative.
    """
    # Check for negative values
    if length < 0 or breadth < 0:
        raise ValueError("Length and breadth must not be negative.")
    # Calculate area
    area = length * breadth
    return area

# Example usage
length = 10
breadth = 20
try:
    area = calculate_rectangle_area(length, breadth)
    print(f"The area of the rectangle with length {length} and breadth {breadth} is {area}.")
except ValueError as e:
    print(e)
```

Output:



```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10> & C:/Users/Namitha/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Nam
itha/OneDrive/Desktop/AIAC/Lab-10/Task-2.py
The area of the rectangle with length 10 and breadth 20 is 200.
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10>
```

TASK-3:

PROMPT:

def c(x,y):

return x*y/100

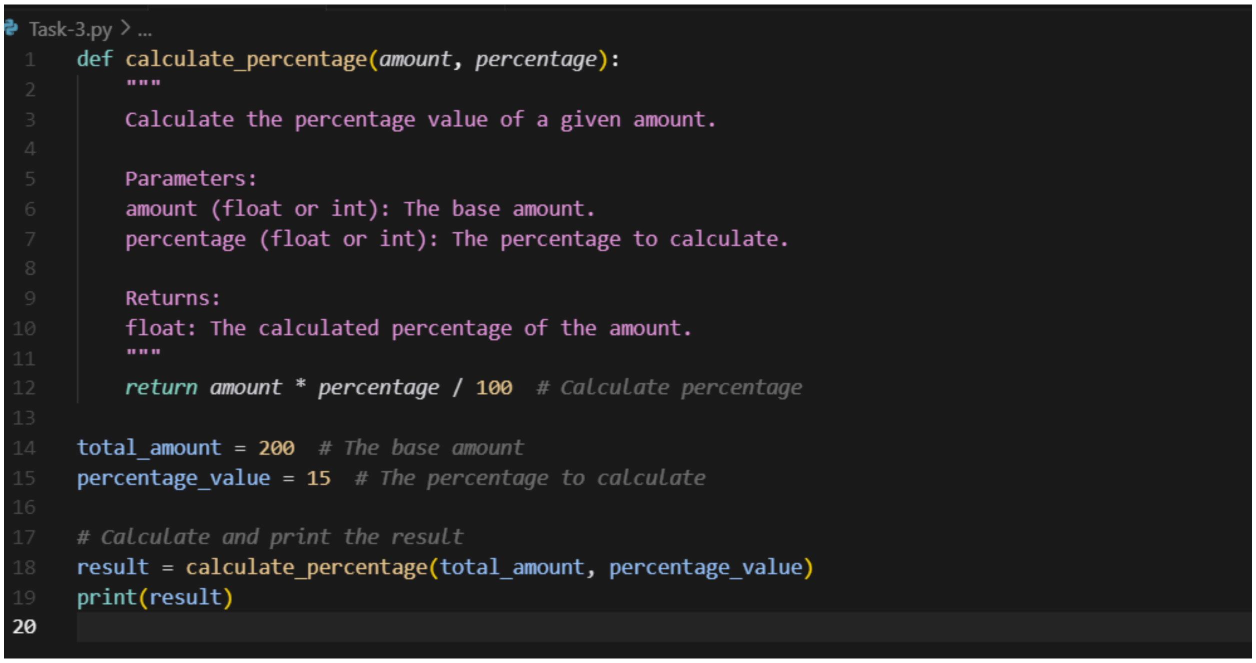
a=200

b=15

print(c(a,b))

make code more readable with descriptive variable names, inline comments, and clear formatting.

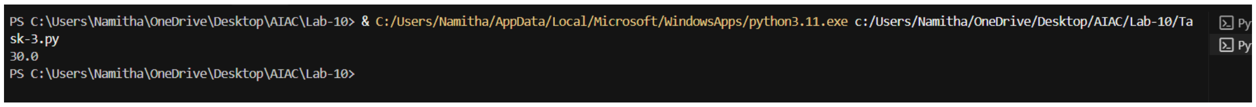
CODE:



Task-3.py > ...

```python
def calculate_percentage(amount, percentage):
    """
    Calculate the percentage value of a given amount.

    Parameters:
    amount (float or int): The base amount.
    percentage (float or int): The percentage to calculate.

    Returns:
    float: The calculated percentage of the amount.
    """
    return amount * percentage / 100  # Calculate percentage

total_amount = 200  # The base amount
percentage_value = 15  # The percentage to calculate

# Calculate and print the result
result = calculate_percentage(total_amount, percentage_value)
print(result)

```

OUTPUT:



```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10> & C:/Users/Namitha/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Namitha/OneDrive/Desktop/AIAC/Lab-10/Ta
sk-3.py
30.0
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10>
```

TASK-4:

PROMPT:

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

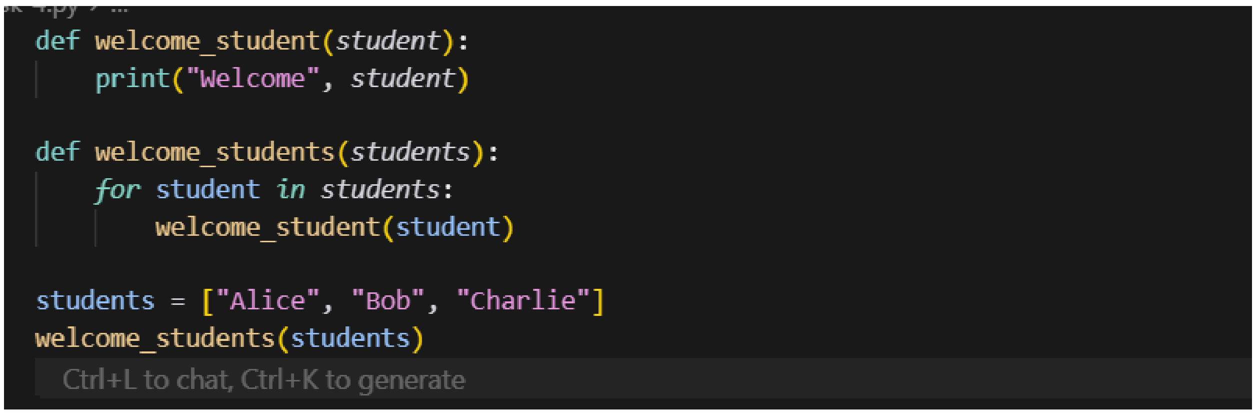
print("Welcome", students[0])

print("Welcome", students[1])

print("Welcome", students[2])

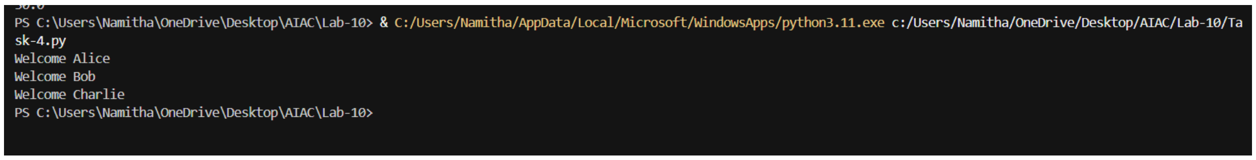
convert this long code into reusable functions.

CODE:



```python
def welcome_student(student):
    print("Welcome", student)

def welcome_students(students):
    for student in students:
        welcome_student(student)

students = ["Alice", "Bob", "Charlie"]
welcome_students(students)
```

OUTPUT:

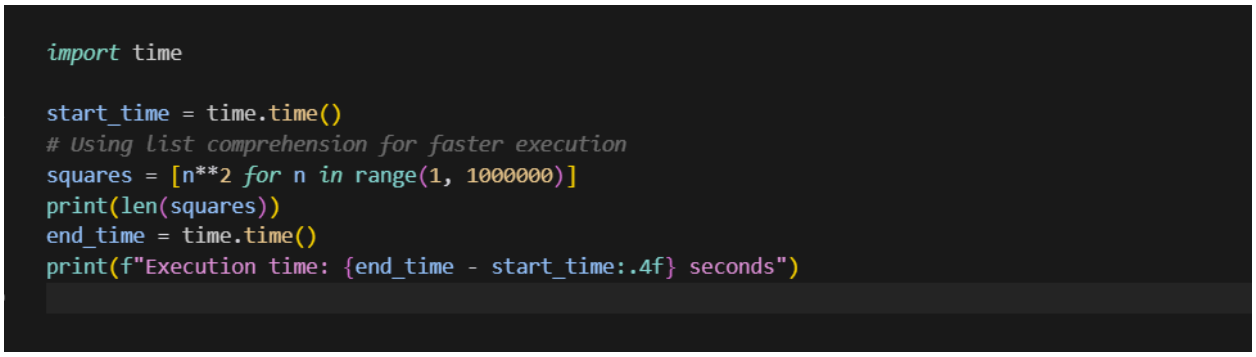


```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10> & C:/Users/Namitha/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Namitha/OneDrive/Desktop/AIAC/Lab-10/Ta
sk-4.py
Welcome Alice
Welcome Bob
Welcome Charlie
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10>
```

TASK-5:

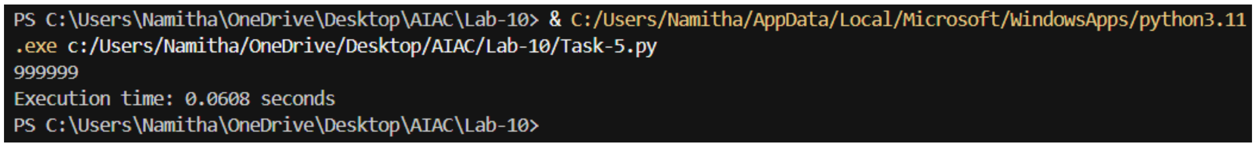
PROMPT:
# Find squares of numbers
nums = [i for i in range(1,1000000)]
squares = []
for n in nums:
squares.append(n**2)
print(len(squares))
make the code run faster.i want an optimized code using list comprehensions or vectorized operations.i want execution time also.

CODE:



```python
import time

start_time = time.time()
# Using list comprehension for faster execution
squares = [n**2 for n in range(1, 1000000)]
print(len(squares))
end_time = time.time()
print(f"Execution time: {end_time - start_time:.4f} seconds")
```

OUTPUT:



```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10> & C:/Users/Namitha/AppData/Local/Microsoft/WindowsApps/python3.11
.exe c:/Users/Namitha/OneDrive/Desktop/AIAC/Lab-10/Task-5.py
999999
Execution time: 0.0608 seconds
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10>
```

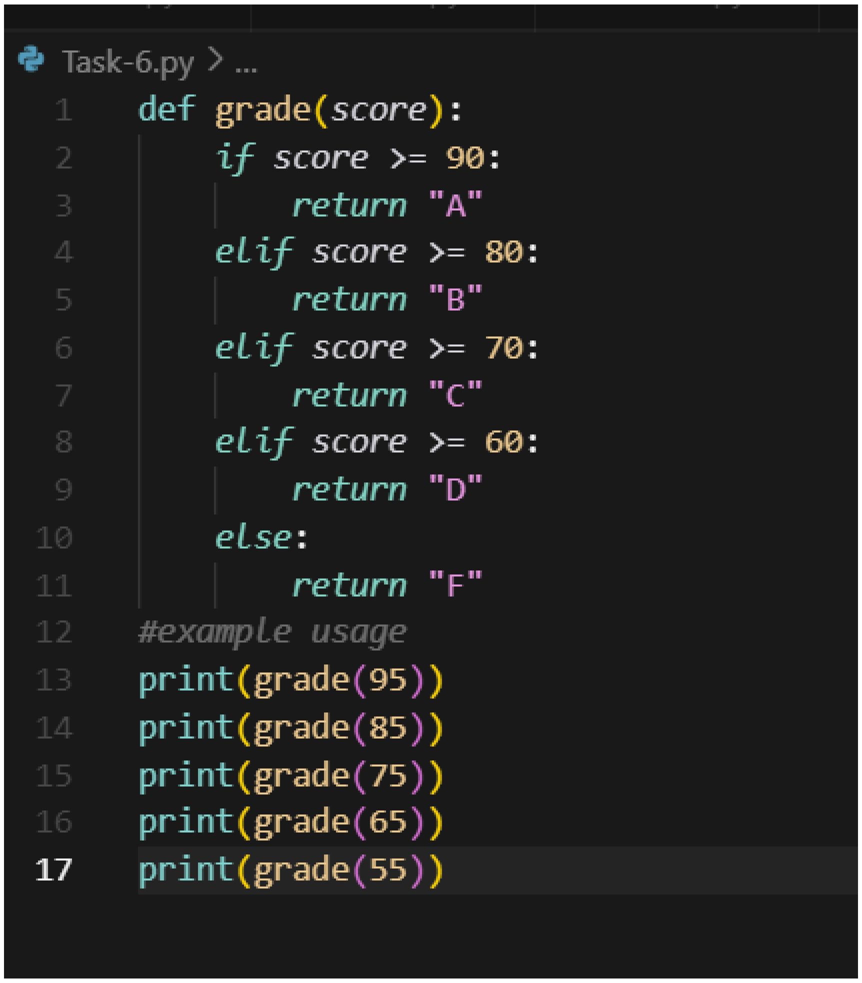
TASK-6:

PROMPT:
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"
```

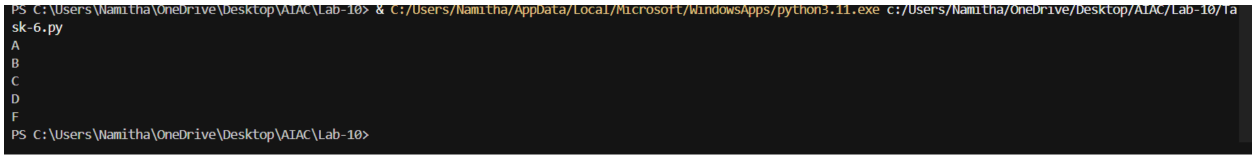
i wnt to reduce the complexity of the code and simplify it.

CODE:


Task-6.py > ...

```python
def grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"
#example usage
print(grade(95))
print(grade(85))
print(grade(75))
print(grade(65))
print(grade(55))
```

OUTPUT:



```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10> & C:/Users/Namitha/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Namitha/OneDrive/Desktop/AIAC/Lab-10/Ta
sk-6.py
A
B
C
D
F
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-10>
```