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) 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** • 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.

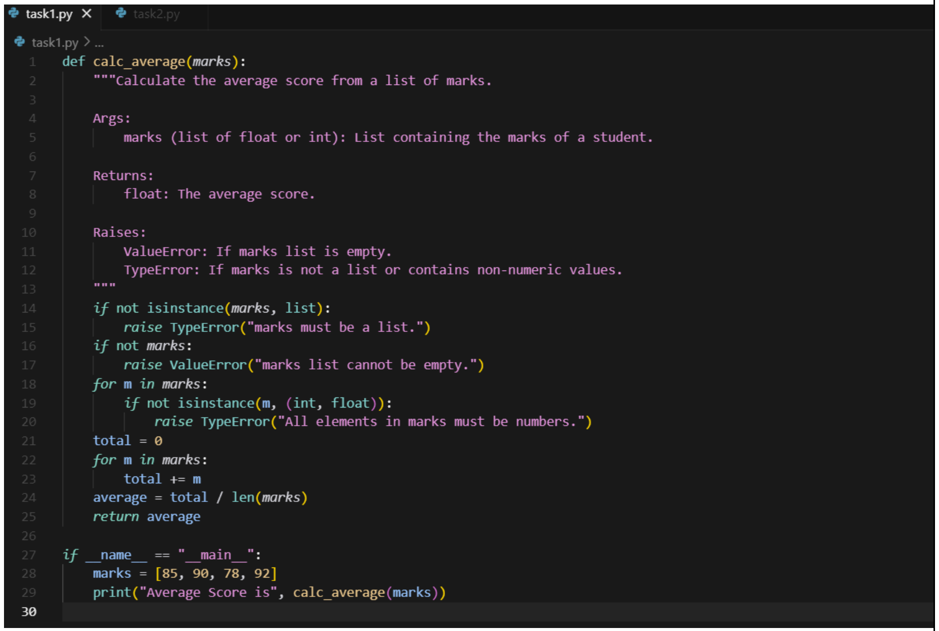
Sample Input Code:

```
# 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:

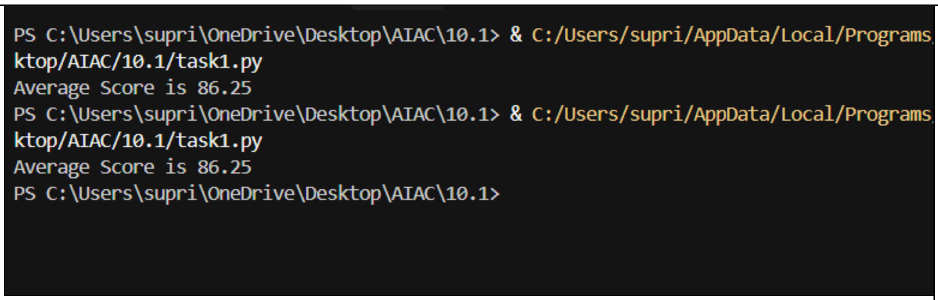
- Corrected and runnable Python code with explanations of the fixes.

CODE:



```python
def calc_average(marks):
    """Calculate the average score from a list of marks.

    Args:
        marks (list of float or int): List containing the marks of a student.

    Returns:
        float: The average score.

    Raises:
        ValueError: If marks list is empty.
        TypeError: If marks is not a list or contains non-numeric values.
    """
    if not isinstance(marks, list):
        raise TypeError("marks must be a list.")
    if not marks:
        raise ValueError("marks list cannot be empty.")
    for m in marks:
        if not isinstance(m, (int, float)):
            raise TypeError("All elements in marks must be numbers.")
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average

if __name__ == "__main__":
    marks = [85, 90, 78, 92]
    print("Average Score is", calc_average(marks))

```

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/Local/Programs
ktop/AIAC/10.1/task1.py
Average Score is 86.25
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/Local/Programs
ktop/AIAC/10.1/task1.py
Average Score is 86.25
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1>
```

Task Description #2 – PEP 8 Compliance

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

Sample Input Code:

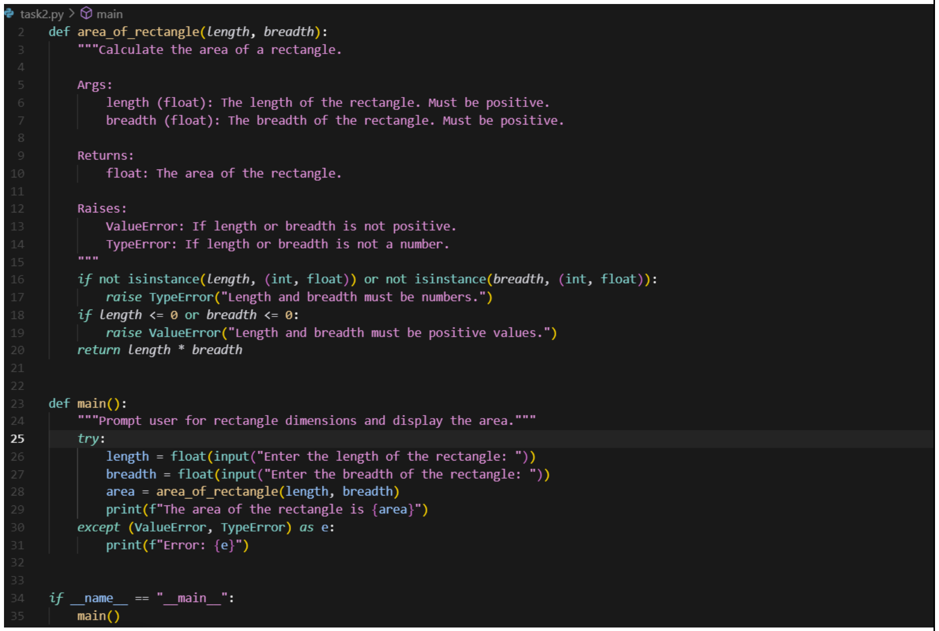
def area_of_rect(L,B):return L*B

print(area_of_rect(10,20))

Expected Output:

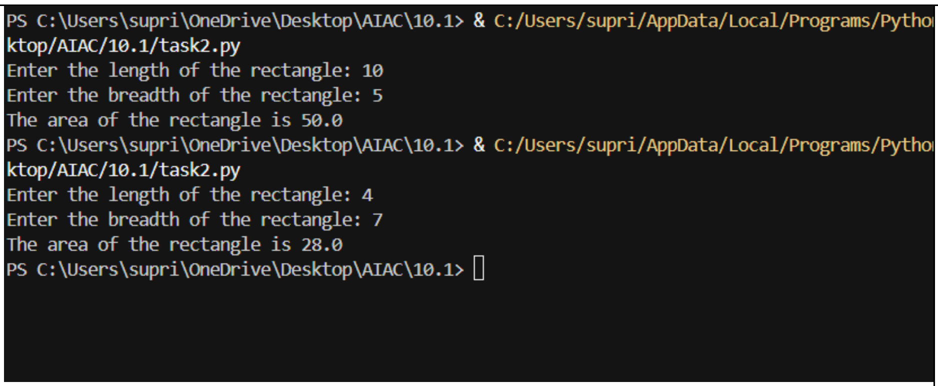
- Well-formatted PEP 8-compliant Python code.

CODE:



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

    Args:
        length (float): The length of the rectangle. Must be positive.
        breadth (float): The breadth of the rectangle. Must be positive.

    Returns:
        float: The area of the rectangle.

    Raises:
        ValueError: If length or breadth is not positive.
        TypeError: If length or breadth is not a number.
    """
    if not isinstance(length, (int, float)) or not isinstance(breadth, (int, float)):
        raise TypeError("Length and breadth must be numbers.")
    if length <= 0 or breadth <= 0:
        raise ValueError("Length and breadth must be positive values.")
    return length * breadth


def main():
    """Prompt user for rectangle dimensions and display the area."""
    try:
        length = float(input("Enter the length of the rectangle: "))
        breadth = float(input("Enter the breadth of the rectangle: "))
        area = area_of_rectangle(length, breadth)
        print(f"The area of the rectangle is {area}")
    except (ValueError, TypeError) as e:
        print(f"Error: {e}")


if __name__ == "__main__":
    main()
```

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/Local/Programs/Pytho
ktop/AIAC/10.1/task2.py
Enter the length of the rectangle: 10
Enter the breadth of the rectangle: 5
The area of the rectangle is 50.0
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/Local/Programs/Pytho
ktop/AIAC/10.1/task2.py
Enter the length of the rectangle: 4
Enter the breadth of the rectangle: 7
The area of the rectangle is 28.0
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1>
```

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

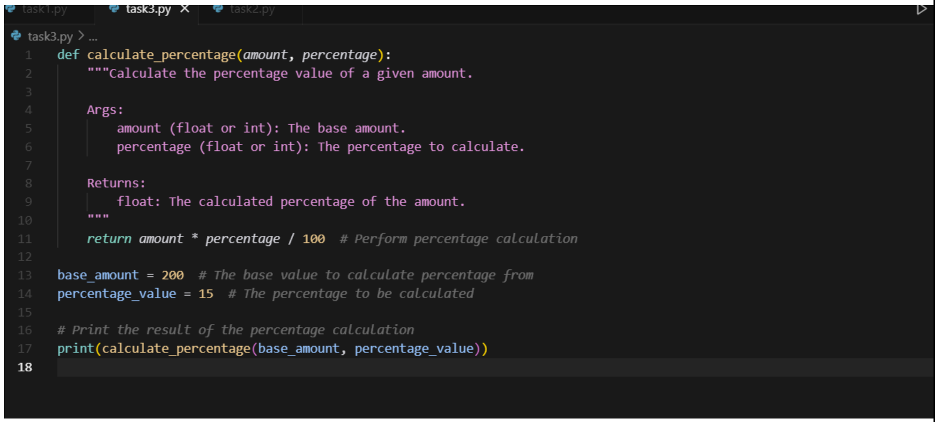
b=15

print(c(a,b))

Expected Output:

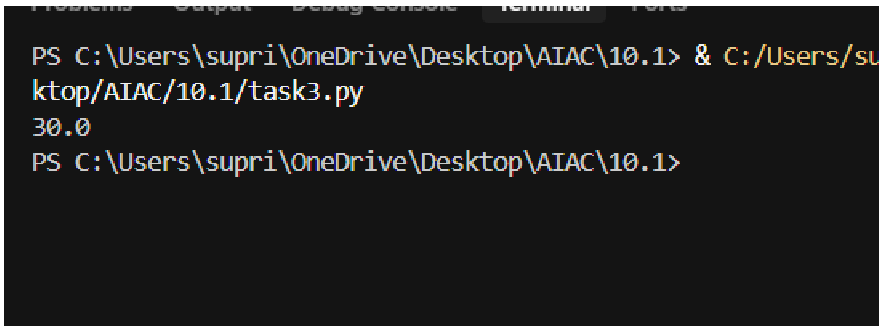
- Python code with descriptive variable names, inline comments, and clear formatting.

CODE:



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

    Args:
        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  # Perform percentage calculation

base_amount = 200  # The base value to calculate percentage from
percentage_value = 15  # The percentage to be calculated

# Print the result of the percentage calculation
print(calculate_percentage(base_amount, percentage_value))
```

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/su
ktop/AIAC/10.1/task3.py
30.0
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1>
```

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])

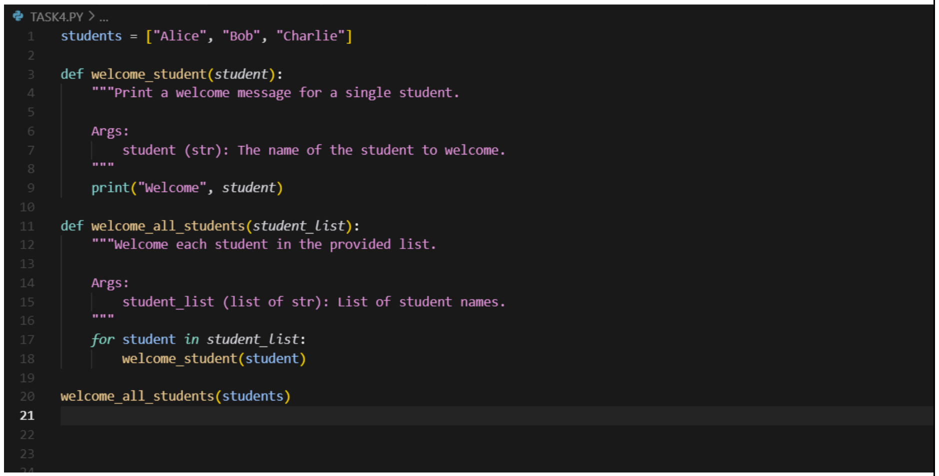
print("Welcome", students[1])

print("Welcome", students[2])

Expected Output:

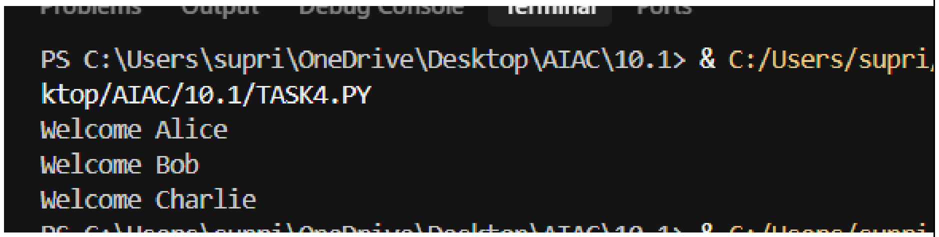
- Modular code with reusable functions.

CODE:



```
TASK4.PY > ...
students = ["Alice", "Bob", "Charlie"]

def welcome_student(student):
    """Print a welcome message for a single student.

    Args:
        student (str): The name of the student to welcome.
    """
    print("Welcome", student)

def welcome_all_students(student_list):
    """Welcome each student in the provided list.

    Args:
        student_list (list of str): List of student names.
    """
    for student in student_list:
        welcome_student(student)

welcome_all_students(students)
```

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri
ktop/AIAC/10.1/TASK4.PY
Welcome Alice
Welcome Bob
Welcome Charlie
```

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:

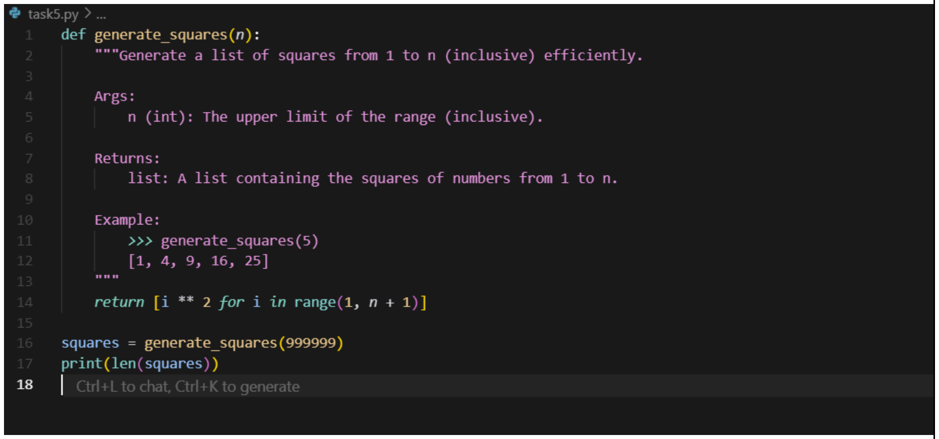
    squares.append(n**2)

print(len(squares))

Expected Output:

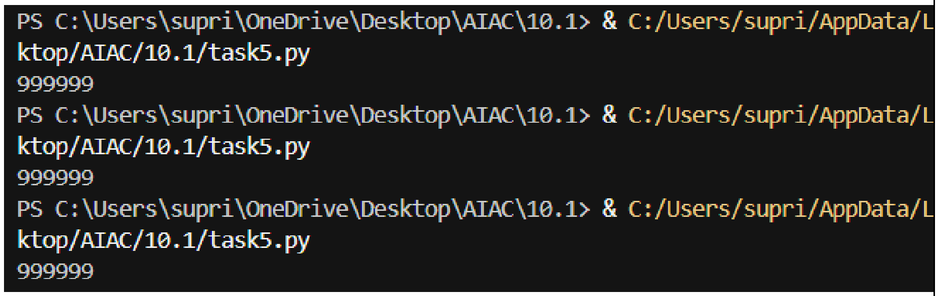
- Optimized code using list comprehensions or vectorized operations.

CODE:



```
task5.py > ...
def generate_squares(n):
    """Generate a list of squares from 1 to n (inclusive) efficiently.

    Args:
        n (int): The upper limit of the range (inclusive).

    Returns:
        list: A list containing the squares of numbers from 1 to n.

    Example:
        >>> generate_squares(5)
        [1, 4, 9, 16, 25]
    """
    return [i ** 2 for i in range(1, n + 1)]

squares = generate_squares(999999)
print(len(squares))
 Ctrl+L to chat, Ctrl+K to generate
```

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/L
ktop/AIAC/10.1/task5.py
999999
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/L
ktop/AIAC/10.1/task5.py
999999
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/L
ktop/AIAC/10.1/task5.py
999999
```

Task Description #6 – Complexity Reduction

Task: Use AI he.

Sample Input Code:

def grade(score):

    if score >= 90:

        return "A"

    else:

        if score >= 80:

            return "B"

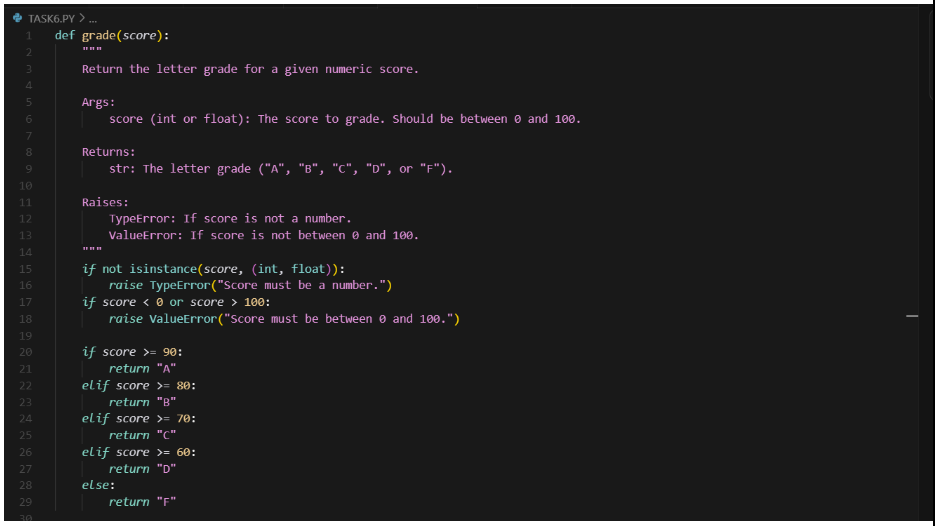
        else:

```
        if score >= 70:
            return "C"
        else:
            if score >= 60:
                return "D"
            else:
                return "F"
```

Expected Output:

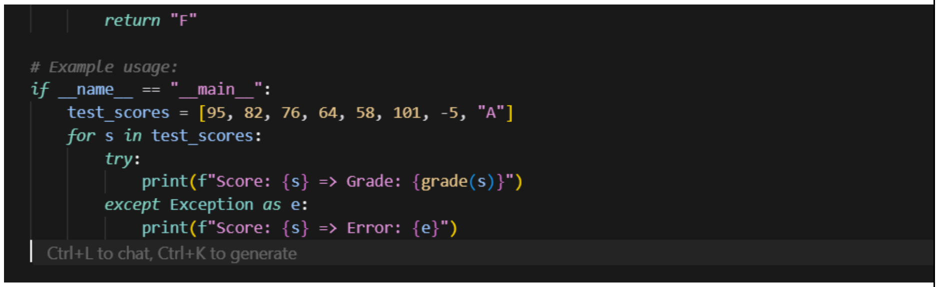
- Cleaner logic using elif or dictionary mapping.

CODE:



TASK6.PY > ...

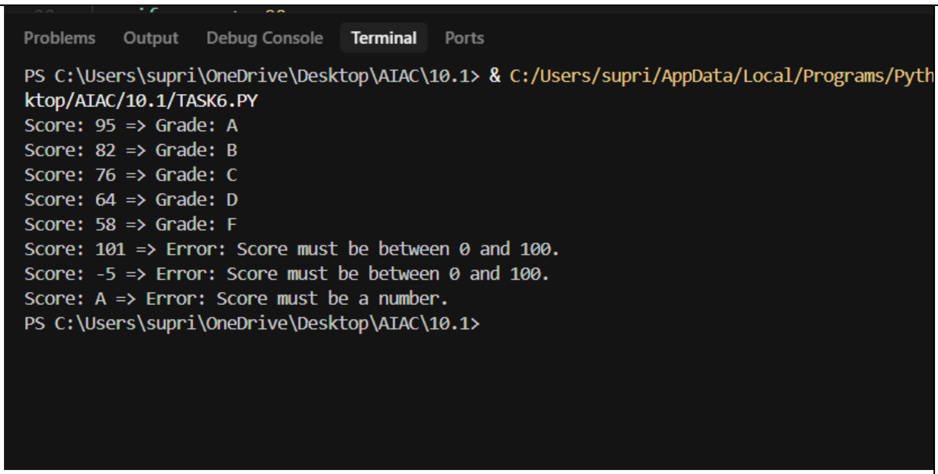
```python
def grade(score):
    """
    Return the letter grade for a given numeric score.

    Args:
        score (int or float): The score to grade. Should be between 0 and 100.

    Returns:
        str: The letter grade ("A", "B", "C", "D", or "F").

    Raises:
        TypeError: If score is not a number.
        ValueError: If score is not between 0 and 100.
    """
    if not isinstance(score, (int, float)):
        raise TypeError("Score must be a number.")
    if score < 0 or score > 100:
        raise ValueError("Score must be between 0 and 100.")

    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"
```



```python
        return "F"

# Example usage:
if __name__ == "__main__":
    test_scores = [95, 82, 76, 64, 58, 101, -5, "A"]
    for s in test_scores:
        try:
            print(f"Score: {s} => Grade: {grade(s)}")
        except Exception as e:
            print(f"Score: {s} => Error: {e}")
```

Ctrl+L to chat, Ctrl+K to generate

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1> & C:/Users/supri/AppData/Local/Programs/Pyth
ktop/AIAC/10.1/TASK6.PY
Score: 95 => Grade: A
Score: 82 => Grade: B
Score: 76 => Grade: C
Score: 64 => Grade: D
Score: 58 => Grade: F
Score: 101 => Error: Score must be between 0 and 100.
Score: -5 => Error: Score must be between 0 and 100.
Score: A => Error: Score must be a number.
PS C:\Users\supri\OneDrive\Desktop\AIAC\10.1>
```