AI ASSISTED CODING

ASSIGNMENT-5.1 AND 6

NAME: Hari Priya

H.T.No.:2303A51104

Task 1:

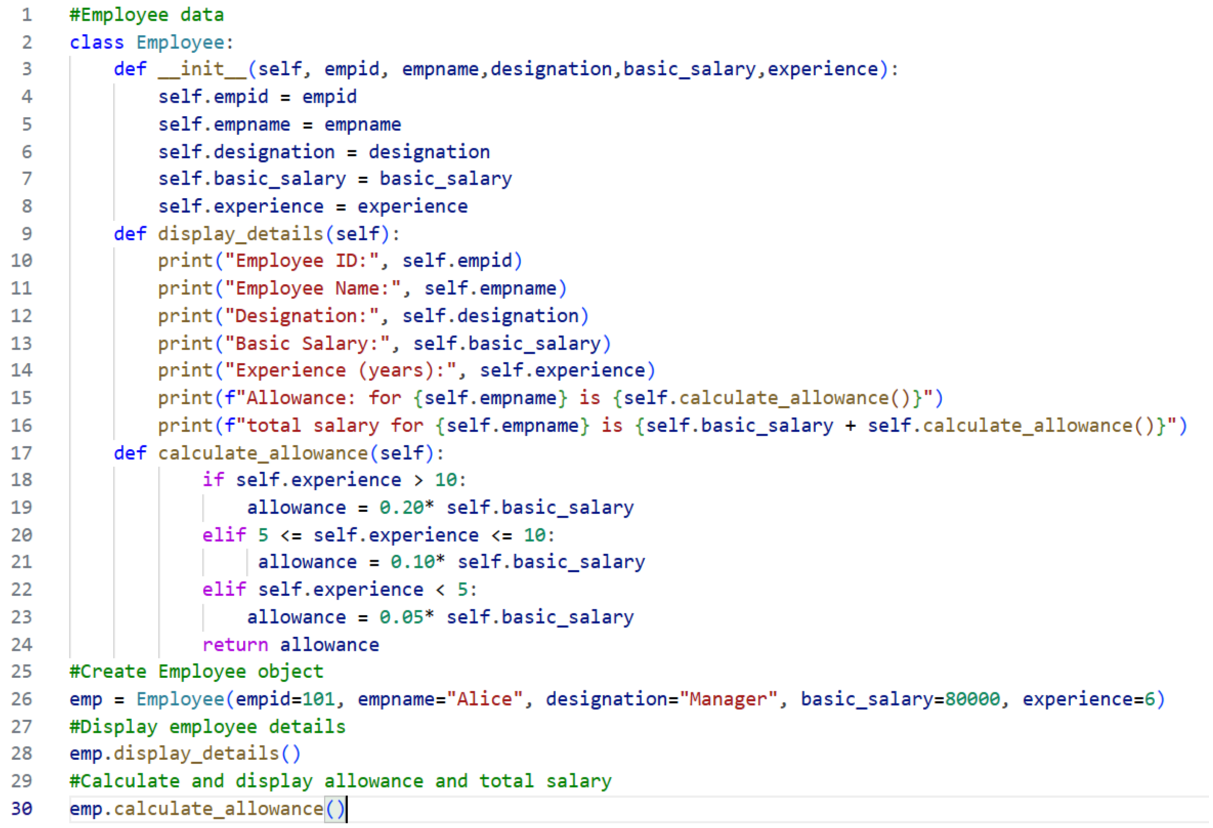
Employee Data: Create Python code that defines a class named `Employee` with the following attributes: `empid`, `empname`,`designation`, `basic_salary`, and `exp`. Implement a method `display_details()` to print all employee details. Implement another method `calculate_allowance()` to determine additional allowance based on experience:

- If `exp > 10 years` → allowance = 20% of `basic_salary`

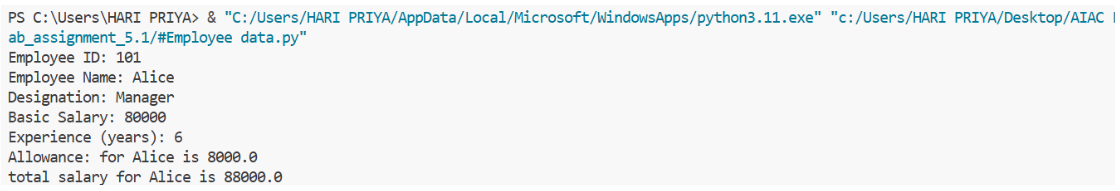
- If `5 ≤ exp ≤ 10 years` → allowance = 10% of `basic_salary`

- If `exp < 5 years` → allowance = 5% of `basic_salary`

Finally, create at least one instance of the `Employee` class, call the `display_details()` method, and print the calculated allowance.



```python
#Employee data
class Employee:
    def __init__(self, empid, empname,designation,basic_salary,experience):
        self.empid = empid
        self.empname = empname
        self.designation = designation
        self.basic_salary = basic_salary
        self.experience = experience
    def display_details(self):
        print("Employee ID:", self.empid)
        print("Employee Name:", self.empname)
        print("Designation:", self.designation)
        print("Basic Salary:", self.basic_salary)
        print("Experience (years):", self.experience)
        print(f"Allowance: for {self.empname} is {self.calculate_allowance()}")
        print(f"total salary for {self.empname} is {self.basic_salary + self.calculate_allowance()}")
    def calculate_allowance(self):
            if self.experience > 10:
                allowance = 0.20* self.basic_salary
            elif 5 <= self.experience <= 10:
                 allowance = 0.10* self.basic_salary
            elif self.experience < 5:
                allowance = 0.05* self.basic_salary
            return allowance
#Create Employee object
emp = Employee(empid=101, empname="Alice", designation="Manager", basic_salary=80000, experience=6)
#Display employee details
emp.display_details()
#Calculate and display allowance and total salary
emp.calculate_allowance()
```



```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC l
ab_assignment_5.1/#Employee data.py"
Employee ID: 101
Employee Name: Alice
Designation: Manager
Basic Salary: 80000
Experience (years): 6
Allowance: for Alice is 8000.0
total salary for Alice is 88000.0
```

Task 2:

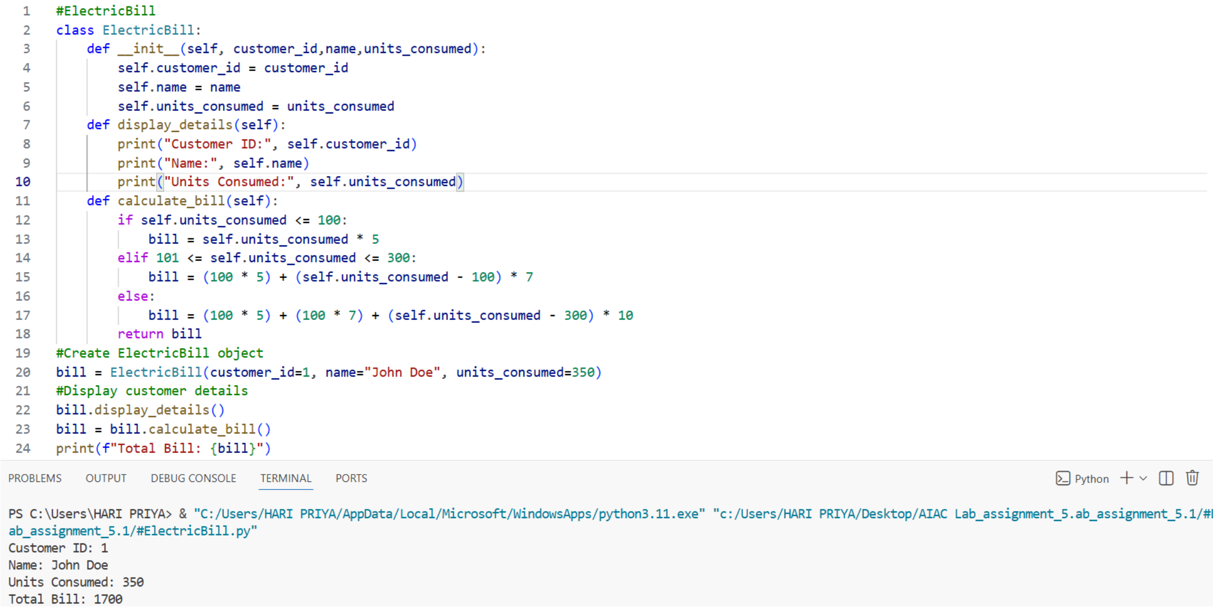
Electricity Bill Calculation- Create Python code that defines a class named `ElectricityBill` with attributes: `customer_id`, `name`, and `units_consumed`. Implement a method `display_details()` to print customer details, and a method `calculate_bill()` where:

- Units ≤ 100 → ₹5 per unit

- 101 to 300 units → ₹7 per unit

- More than 300 units → ₹10 per unit

Create a bill object, display details, and print the total bill amount.



```python
#ElectricBill
class ElectricBill:
    def __init__(self, customer_id,name,units_consumed):
        self.customer_id = customer_id
        self.name = name
        self.units_consumed = units_consumed
    def display_details(self):
        print("Customer ID:", self.customer_id)
        print("Name:", self.name)
        print("Units Consumed:", self.units_consumed)
    def calculate_bill(self):
        if self.units_consumed <= 100:
            bill = self.units_consumed * 5
        elif 101 <= self.units_consumed <= 300:
            bill = (100 * 5) + (self.units_consumed - 100) * 7
        else:
            bill = (100 * 5) + (100 * 7) + (self.units_consumed - 300) * 10
        return bill
#Create ElectricBill object
bill = ElectricBill(customer_id=1, name="John Doe", units_consumed=350)
#Display customer details
bill.display_details()
bill = bill.calculate_bill()
print(f"Total Bill: {bill}")
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.ab_assignment_5.1/#l
ab_assignment_5.1/#ElectricBill.py"
Customer ID: 1
Name: John Doe
Units Consumed: 350
Total Bill: 1700
```

Task 3:

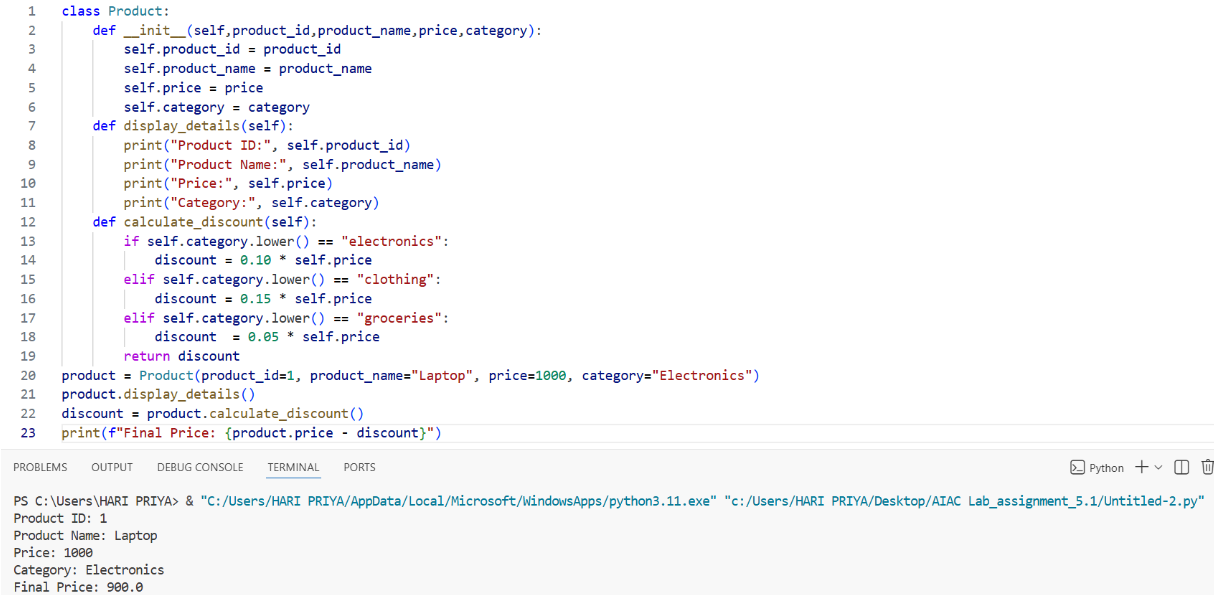
Product Discount Calculation- Create Python code that defines a class named `Product` with attributes: `product_id`, `product_name`,`price`, and `category`. Implement a method `display_details()` to print product details. Implement another method `calculate_discount()` where:

- Electronics → 10% discount

- Clothing → 15% discount

- Grocery → 5% discount

Create at least one product object, display details, and print the final price after discount.



```python
class Product:
    def __init__(self,product_id,product_name,price,category):
        self.product_id = product_id
        self.product_name = product_name
        self.price = price
        self.category = category
    def display_details(self):
        print("Product ID:", self.product_id)
        print("Product Name:", self.product_name)
        print("Price:", self.price)
        print("Category:", self.category)
    def calculate_discount(self):
        if self.category.lower() == "electronics":
            discount = 0.10 * self.price
        elif self.category.lower() == "clothing":
            discount = 0.15 * self.price
        elif self.category.lower() == "groceries":
            discount  = 0.05 * self.price
        return discount
product = Product(product_id=1, product_name="Laptop", price=1000, category="Electronics")
product.display_details()
discount = product.calculate_discount()
print(f"Final Price: {product.price - discount}")
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.1/Untitled-2.py"
Product ID: 1
Product Name: Laptop
Price: 1000
Category: Electronics
Final Price: 900.0
```

Task 4:

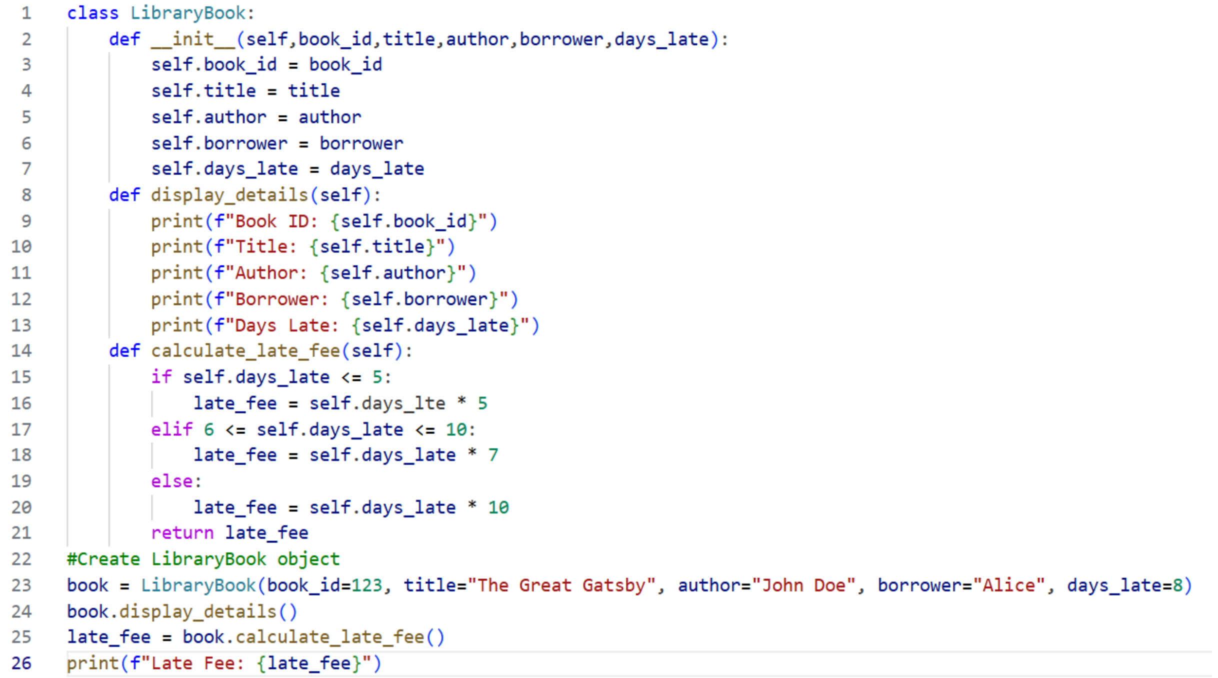
Book Late Fee Calculation- Create Python code that defines a class named `LibraryBook` with attributes: `book_id`, `title`, `author`,`borrower`, and `days_late`. Implement a method `display_details()`to print book details, and a method `calculate_late_fee()` where:

- Days late ≤ 5 → ₹5 per day

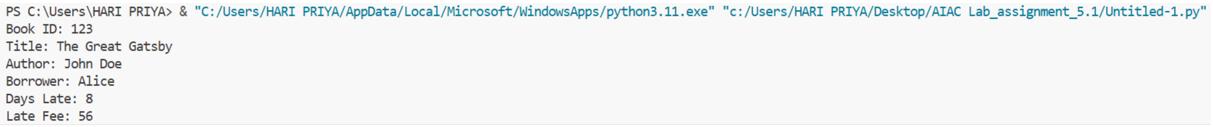
- 6 to 10 days late → ₹7 per day

- More than 10 days late → ₹10 per day

Create a book object, display details, and print the late fee.



```python
class LibraryBook:
    def __init__(self,book_id,title,author,borrower,days_late):
        self.book_id = book_id
        self.title = title
        self.author = author
        self.borrower = borrower
        self.days_late = days_late
    def display_details(self):
        print(f"Book ID: {self.book_id}")
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Borrower: {self.borrower}")
        print(f"Days Late: {self.days_late}")
    def calculate_late_fee(self):
        if self.days_late <= 5:
            late_fee = self.days_lte * 5
        elif 6 <= self.days_late <= 10:
            late_fee = self.days_late * 7
        else:
            late_fee = self.days_late * 10
        return late_fee
#Create LibraryBook object
book = LibraryBook(book_id=123, title="The Great Gatsby", author="John Doe", borrower="Alice", days_late=8)
book.display_details()
late_fee = book.calculate_late_fee()
print(f"Late Fee: {late_fee}")
```



```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.1/Untitled-1.py"
Book ID: 123
Title: The Great Gatsby
Author: John Doe
Borrower: Alice
Days Late: 8
Late Fee: 56
```

Task 5:

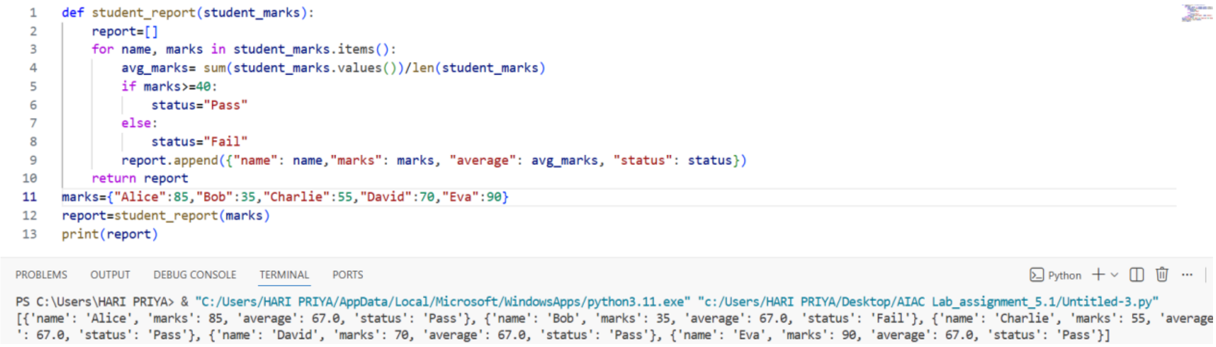
Student Performance Report - Define a function `student_report(student_data)` that accepts a dictionary containing student names and their marks. The function should:

- Calculate the average score for each student

- Determine pass/fail status (pass ≥ 40)

- Return a summary report as a list of dictionaries

Use Copilot suggestions as you build the function and format the output.



```python
def student_report(student_marks):
    report=[]
    for name, marks in student_marks.items():
        avg_marks= sum(student_marks.values())/len(student_marks)
        if marks>=40:
            status="Pass"
        else:
            status="Fail"
        report.append({"name": name,"marks": marks, "average": avg_marks, "status": status})
    return report
marks={"Alice":85,"Bob":35,"Charlie":55,"David":70,"Eva":90}
report=student_report(marks)
print(report)
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.1/Untitled-3.py"
[{'name': 'Alice', 'marks': 85, 'average': 67.0, 'status': 'Pass'}, {'name': 'Bob', 'marks': 35, 'average': 67.0, 'status': 'Fail'}, {'name': 'Charlie', 'marks': 55, 'average
': 67.0, 'status': 'Pass'}, {'name': 'David', 'marks': 70, 'average': 67.0, 'status': 'Pass'}, {'name': 'Eva', 'marks': 90, 'average': 67.0, 'status': 'Pass'}]
```

Task 6:

Taxi Fare Calculation-Create Python code that defines a class named `TaxiRide` with attributes: `ride_id`, `driver_name`, `distance_km`, and `waiting_time_min`. Implement a method `display_details()` to print ride details, and a method `calculate_fare()` where:

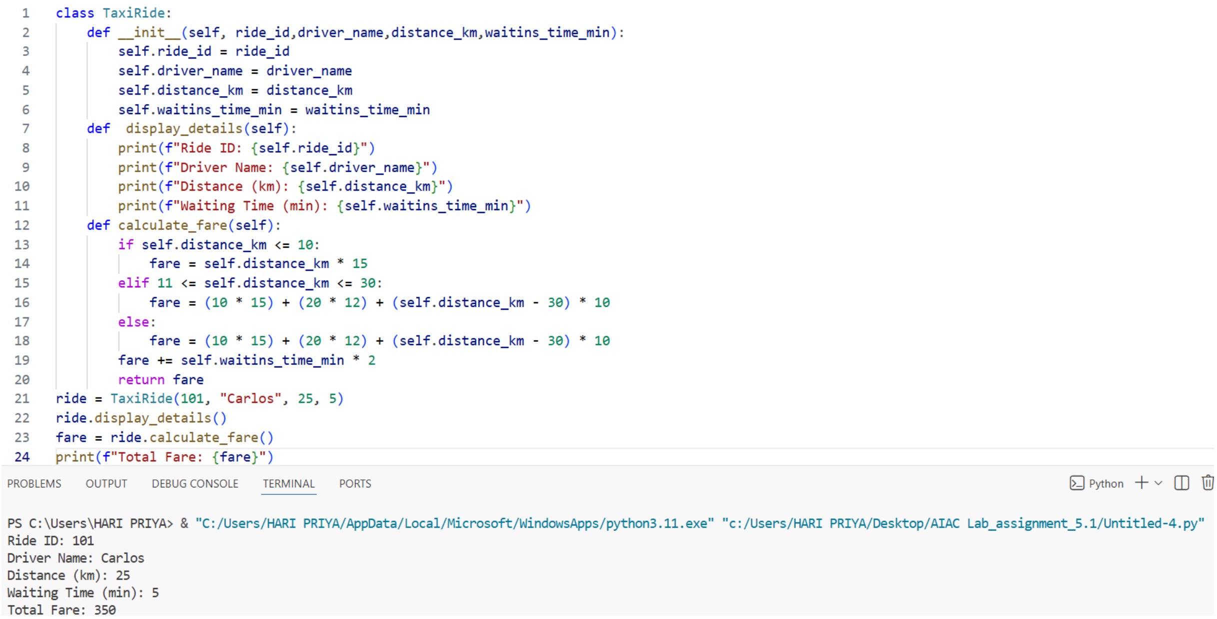
- ₹15 per km for the first 10 km

- ₹12 per km for the next 20 km

- ₹10 per km above 30 km

- Waiting charge: ₹2 per minute

Create a ride object, display details, and print the total fare.



```python
class TaxiRide:
    def __init__(self, ride_id,driver_name,distance_km,waitins_time_min):
        self.ride_id = ride_id
        self.driver_name = driver_name
        self.distance_km = distance_km
        self.waitins_time_min = waitins_time_min
    def  display_details(self):
        print(f"Ride ID: {self.ride_id}")
        print(f"Driver Name: {self.driver_name}")
        print(f"Distance (km): {self.distance_km}")
        print(f"Waiting Time (min): {self.waitins_time_min}")
    def calculate_fare(self):
        if self.distance_km <= 10:
            fare = self.distance_km * 15
        elif 11 <= self.distance_km <= 30:
            fare = (10 * 15) + (20 * 12) + (self.distance_km - 30) * 10
        else:
            fare = (10 * 15) + (20 * 12) + (self.distance_km - 30) * 10
        fare += self.waitins_time_min * 2
        return fare
ride = TaxiRide(101, "Carlos", 25, 5)
ride.display_details()
fare = ride.calculate_fare()
print(f"Total Fare: {fare}")
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.1/Untitled-4.py"
Ride ID: 101
Driver Name: Carlos
Distance (km): 25
Waiting Time (min): 5
Total Fare: 350
```

Task 7:

Statistics Subject Performance - Create a Python function `statistics_subject(scores_list)` that accepts a list of 60 student scores and computes key performance statistics. The function should return the following: - Highest score in the class

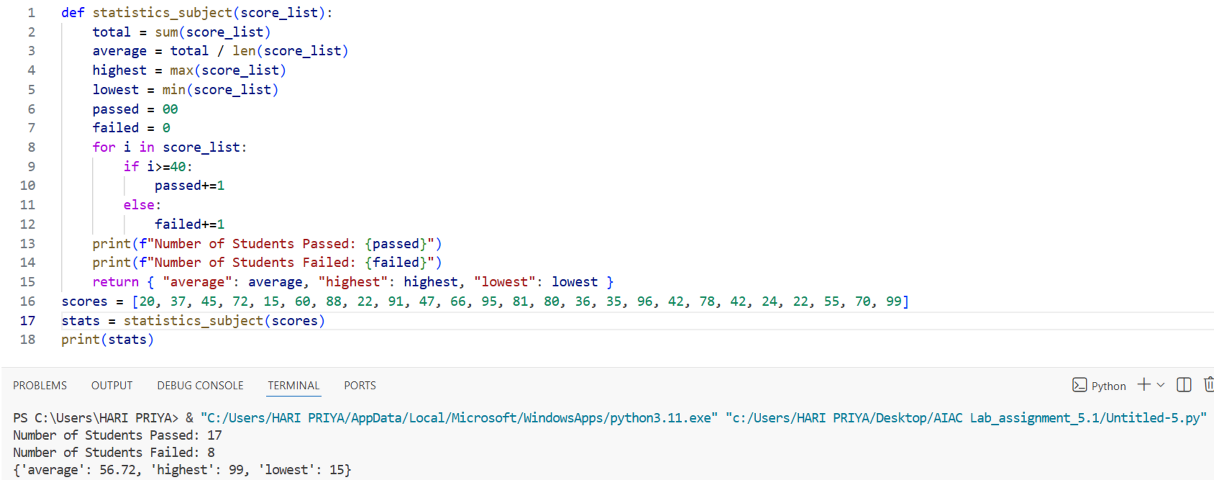
- Lowest score in the class

- Class average score

- Number of students passed (score ≥ 40)

- Number of students failed (score < 40)

Allow Copilot to assist with aggregations and logic



```python
def statistics_subject(score_list):
    total = sum(score_list)
    average = total / len(score_list)
    highest = max(score_list)
    lowest = min(score_list)
    passed = 00
    failed = 0
    for i in score_list:
        if i>=40:
            passed+=1
        else:
            failed+=1
    print(f"Number of Students Passed: {passed}")
    print(f"Number of Students Failed: {failed}")
    return { "average": average, "highest": highest, "lowest": lowest }
scores = [20, 37, 45, 72, 15, 60, 88, 22, 91, 47, 66, 95, 81, 80, 36, 35, 96, 42, 78, 42, 24, 22, 55, 70, 99]
stats = statistics_subject(scores)
print(stats)
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.1/Untitled-5.py"
Number of Students Passed: 17
Number of Students Failed: 8
{'average': 56.72, 'highest': 99, 'lowest': 15}
```

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Task Description #8 (Transparency in Algorithm Optimization)

Task: Use AI to generate two solutions for checking prime numbers:

• Naive approach(basic)

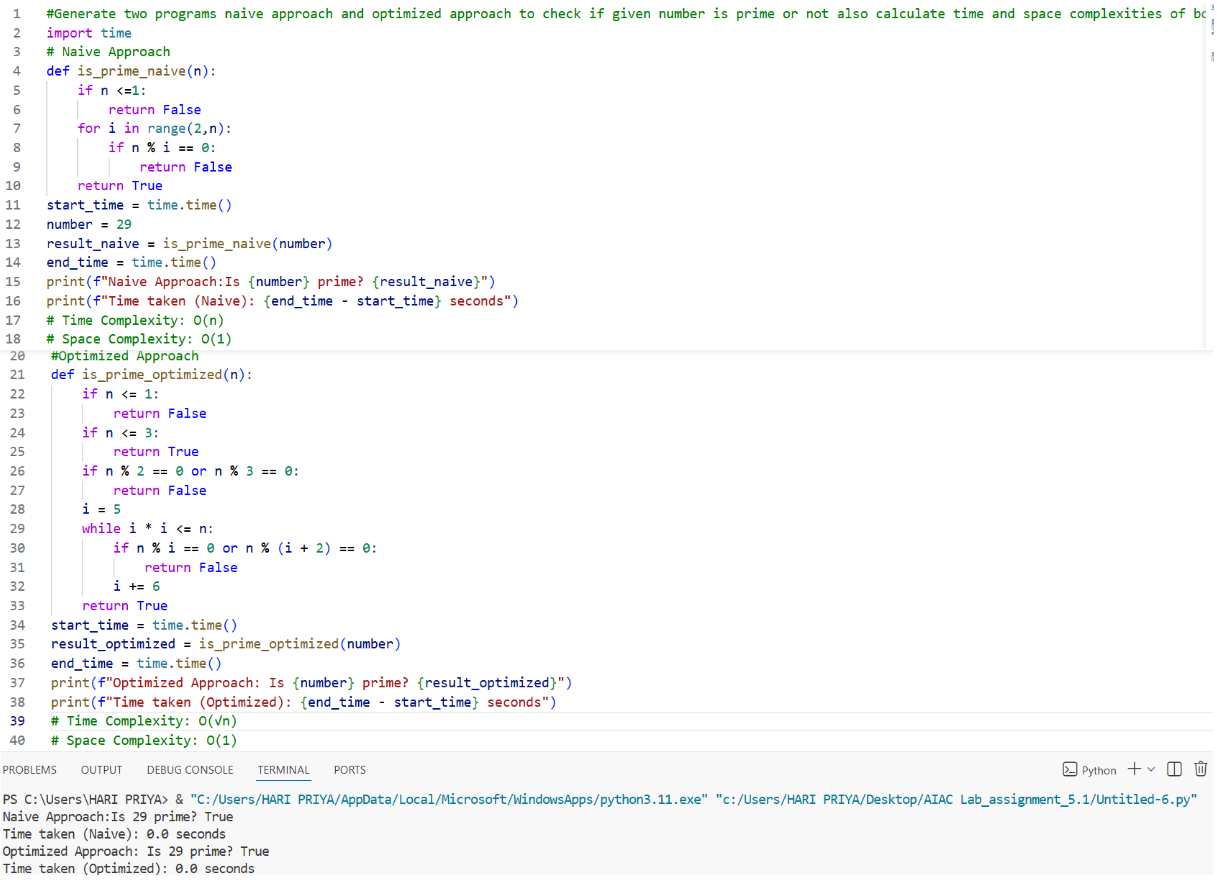
- Optimized approach

Prompt:

“Generate Python code for two prime-checking methods and explain how the optimized version improves performance.”

Expected Output:

- Code for both methods.
- Transparent explanation of time complexity.
- Comparison highlighting efficiency improvements.



```python
#Generate two programs naive approach and optimized approach to check if given number is prime or not also calculate time and space complexities of bo
import time
# Naive Approach
def is_prime_naive(n):
    if n <=1:
        return False
    for i in range(2,n):
        if n % i == 0:
            return False
    return True
start_time = time.time()
number = 29
result_naive = is_prime_naive(number)
end_time = time.time()
print(f"Naive Approach:Is {number} prime? {result_naive}")
print(f"Time taken (Naive): {end_time - start_time} seconds")
# Time Complexity: O(n)
# Space Complexity: O(1)
#Optimized Approach
def is_prime_optimized(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True
start_time = time.time()
result_optimized = is_prime_optimized(number)
end_time = time.time()
print(f"Optimized Approach: Is {number} prime? {result_optimized}")
print(f"Time taken (Optimized): {end_time - start_time} seconds")
# Time Complexity: O(√n)
# Space Complexity: O(1)
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.1/Untitled-6.py"
Naive Approach:Is 29 prime? True
Time taken (Naive): 0.0 seconds
Optimized Approach: Is 29 prime? True
Time taken (Optimized): 0.0 seconds
```

Task Description #9 (Transparency in Recursive Algorithms)

Objective: Use AI to generate a recursive function to calculate Fibonacci numbers.

Instructions:

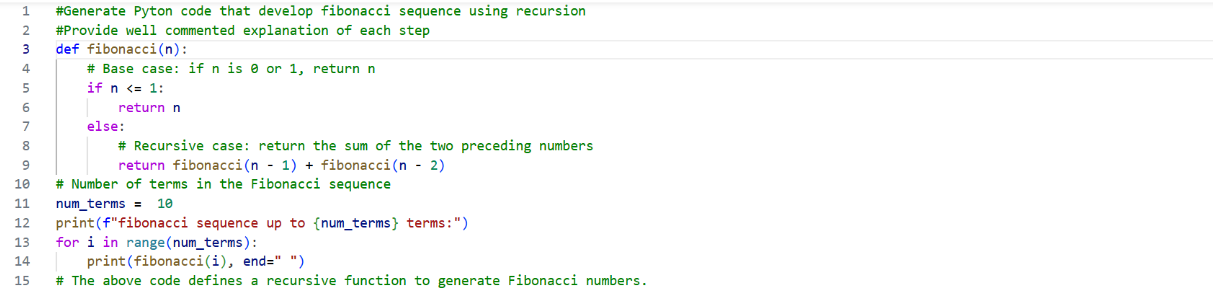
1. Ask AI to add clear comments explaining recursion.
2. Ask AI to explain base cases and recursive calls.

Expected Output:

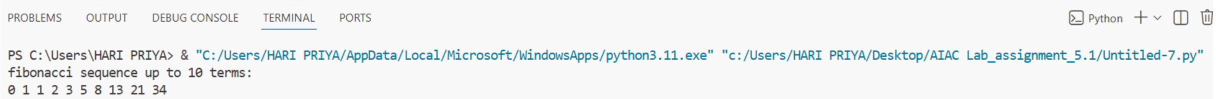
- Well-commented recursive code.

- Clear explanation of how recursion works.

- Verification that explanation matches actual execution.



```python
#Generate Pyton code that develop fibonacci sequence using recursion
#Provide well commented explanation of each step
def fibonacci(n):
    # Base case: if n is 0 or 1, return n
    if n <= 1:
        return n
    else:
        # Recursive case: return the sum of the two preceding numbers
        return fibonacci(n - 1) + fibonacci(n - 2)
# Number of terms in the Fibonacci sequence
num_terms =  10
print(f"fibonacci sequence up to {num_terms} terms:")
for i in range(num_terms):
    print(fibonacci(i), end=" ")
# The above code defines a recursive function to generate Fibonacci numbers.
```



```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignment_5.1/Untitled-7.py"
fibonacci sequence up to 10 terms:
0 1 1 2 3 5 8 13 21 34
```

Task Description #10 (Transparency in Error Handling)

Task: Use AI to generate a Python program that reads a file and processes data.

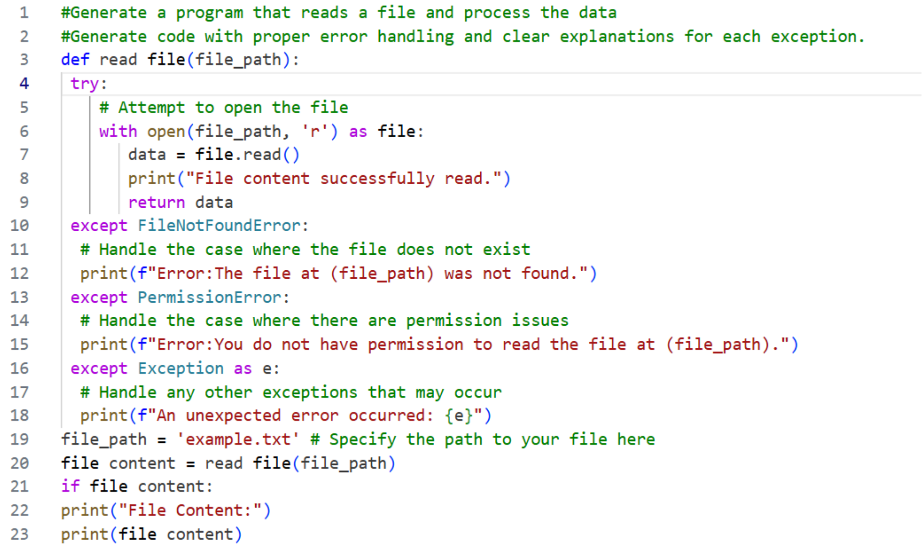
Prompt:

"Generate code with proper error handling and clear explanations for each exception."
Expected Output:

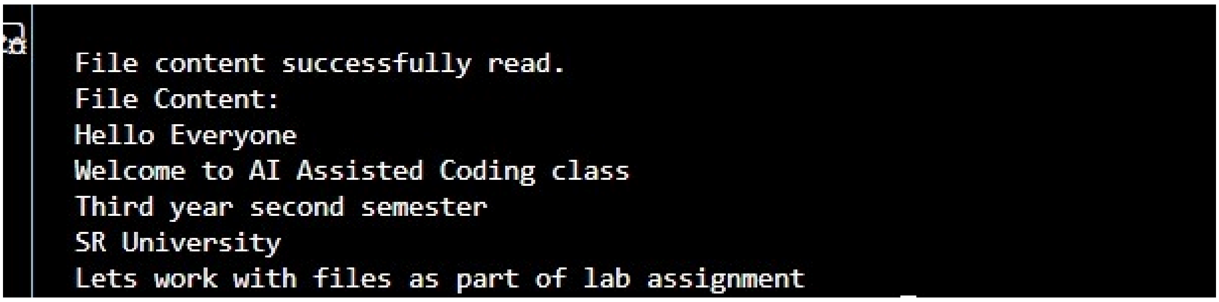
- Code with meaningful exception handling.

- Clear comments explaining each error scenario.

- Validation that explanations align with runtime behavior.



```python
#Generate a program that reads a file and process the data
#Generate code with proper error handling and clear explanations for each exception.
def read file(file_path):
 try:
    # Attempt to open the file
    with open(file_path, 'r') as file:
       data = file.read()
       print("File content successfully read.")
       return data
 except FileNotFoundError:
  # Handle the case where the file does not exist
  print(f"Error:The file at (file_path) was not found.")
 except PermissionError:
  # Handle the case where there are permission issues
  print(f"Error:You do not have permission to read the file at (file_path).")
 except Exception as e:
  # Handle any other exceptions that may occur
  print(f"An unexpected error occurred: {e}")
file_path = 'example.txt' # Specify the path to your file here
file content = read file(file_path)
if file content:
print("File Content:")
print(file content)
```



```
File content successfully read.
File Content:
Hello Everyone
Welcome to AI Assisted Coding class
Third year second semester
SR University
Lets work with files as part of lab assignment
```