AI-Based Code Completion – Classes, Loops, and Conditionals

NAME:CHENCHU RAJU

H.T.NO:2303A52206

BATCH:43

AI Assisted Coding

Assignment_6.4

Task 1: Student Performance Evaluation System

AI.PY

my-app > AI.PY > ...

```python
class Student:

    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def display_details(self):
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks Obtained: {self.marks}")

    def check_performance(self, class_average):
        if self.marks > class_average:
            return f"✓ {self.name} performed ABOVE average (Class avg: {
        elif self.marks == class_average:
            return f"= {self.name} performed AT average (Class avg: {cla
        else:
            return f"X {self.name} performed BELOW average (Class avg: {
if __name__ == "__main__":
    print("=" * 60)
    print("STUDENT PERFORMANCE EVALUATION SYSTEM")
    print("=" * 60)
```

AI.PY

my-app > AI.PY > ...

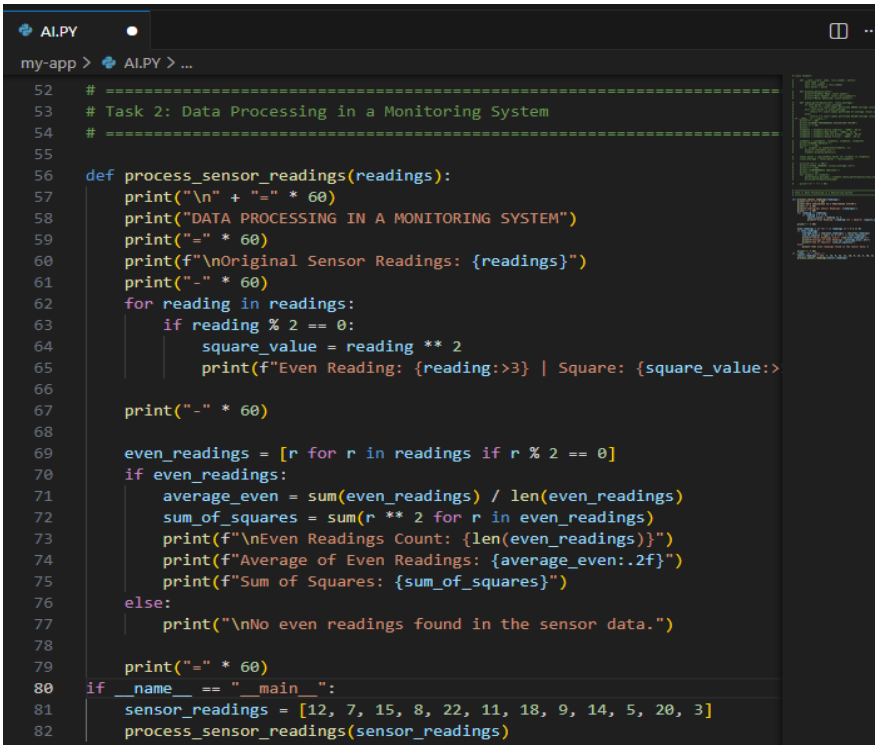
```
if __name__ == "__main__":
    print("=" * 60)
    print("STUDENT PERFORMANCE EVALUATION SYSTEM")
    print("=" * 60)
    print()
    student1 = Student("Alice Johnson", "S001", 85.5)
    student2 = Student("Bob Smith", "S002", 72.0)
    student3 = Student("Charlie Brown", "S003", 90.0)
    student4 = Student("Diana Prince", "S004", 65.5)

    students = [student1, student2, student3, student4]
    print("STUDENT DETAILS:")
    print("-" * 60)
    for i, student in enumerate(students, 1):
        print(f"\nStudent {i}:")
        student.display_details()

    total_marks = sum(student.marks for student in students)
    class_average = total_marks / len(students)

    print(f"\n{'=' * 60}")
    print(f"CLASS AVERAGE: {class_average:.2f}")
    print("=" * 60)
    print("\nPERFORMANCE ANALYSIS:")
    print("-" * 60)
    for student in students:
        performance_message = student.check_performance(class_average)
        print(performance_message)

    print("\n" + "=" * 60)
```

OUTPUT:

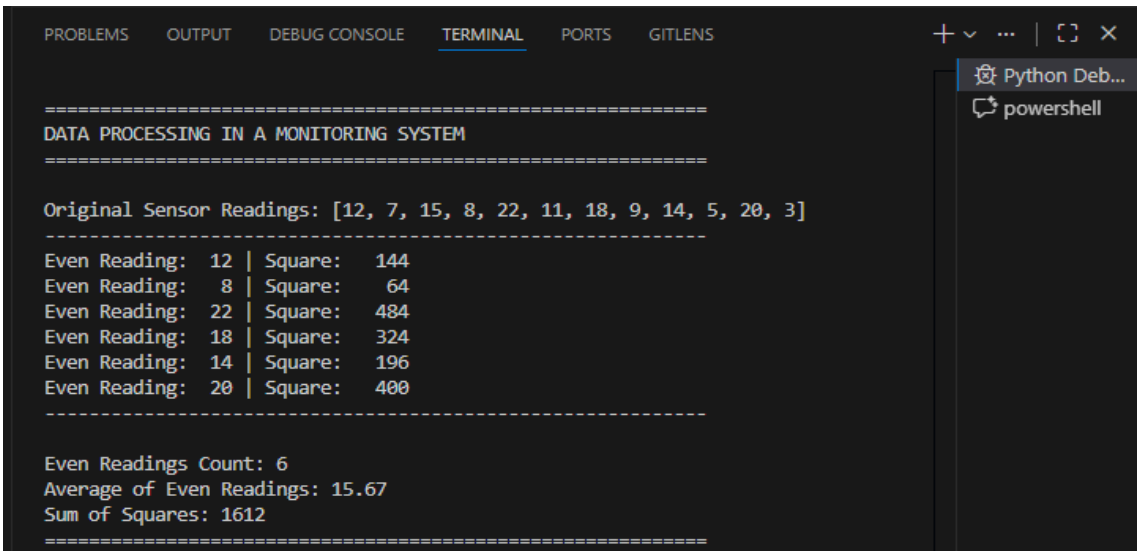
```
============================================================
STUDENT PERFORMANCE EVALUATION SYSTEM
============================================================

STUDENT DETAILS:
------------------------------------------------------------

Student 1:
Student Name: Alice Johnson
Roll Number: S001
Marks Obtained: 85.5

Student 2:
Student Name: Bob Smith
Roll Number: S002
Marks Obtained: 72.0

Student 3:
Student Name: Charlie Brown
Roll Number: S003
Marks Obtained: 90.0

Student 4:
Student Name: Diana Prince
Roll Number: S004
Marks Obtained: 65.5

============================================================
CLASS AVERAGE: 78.25
============================================================

PERFORMANCE ANALYSIS:
------------------------------------------------------------
✓ Alice Johnson performed ABOVE average (Class avg: 78.25, Student marks: 85.5)
✗ Bob Smith performed BELOW average (Class avg: 78.25, Student marks: 72.0)
✓ Charlie Brown performed ABOVE average (Class avg: 78.25, Student marks: 90.0)
------------------------------------------------------------
✓ Alice Johnson performed ABOVE average (Class avg: 78.25, Student marks: 85.5)
✗ Bob Smith performed BELOW average (Class avg: 78.25, Student marks: 72.0)
------------------------------------------------------------
✓ Alice Johnson performed ABOVE average (Class avg: 78.25, Student marks: 85.5)
------------------------------------------------------------
```

Summary: a *Student Performance Evaluation System* was created using a Python class. AI-generated methods helped display student details and evaluate performance using conditional statements. This demonstrated how AI can quickly complete class methods and reduce manual coding effort

Task 2: Data Processing in a Monitoring System

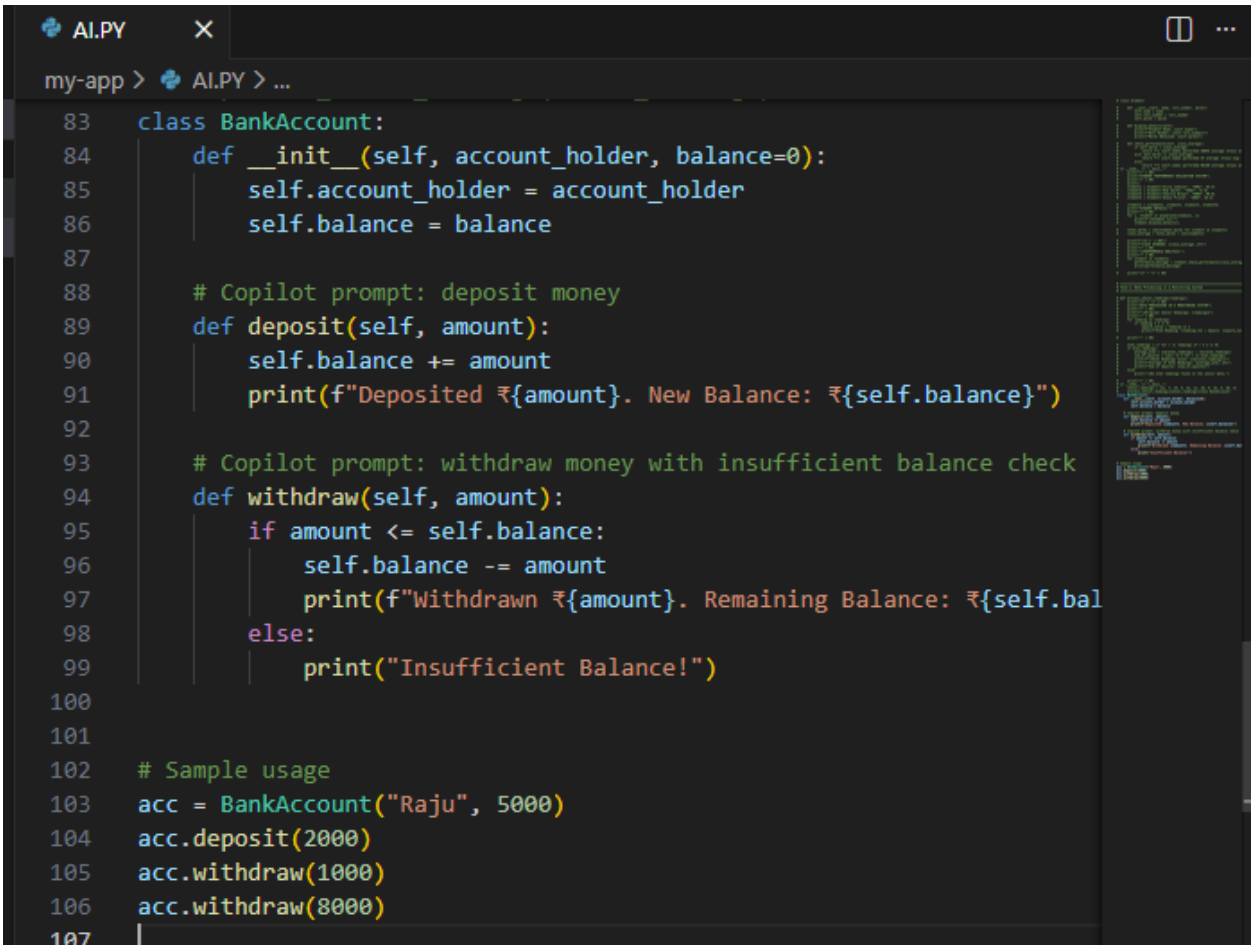


OUTPUT:



Summary: a *Monitoring System* processed sensor readings using a loop. AI completed the logic to identify even numbers and calculate their squares. This showed how AI effectively handles repetitive tasks and loop-based operations.

Task 3: Banking Transaction Simulation



Output:

```
ocuments\Desktop\Desktop\full stack\my-app\AI.PY'
Deposited ₹2000. New Balance: ₹7000
Withdrawn ₹1000. Remaining Balance: ₹6000
Insufficient Balance!
```

Summary: A *Bank Account Simulation* was implemented with deposit and withdrawal features. AI-generated methods included balance updates and

validation checks for insufficient funds. This highlighted the usefulness of AI in writing secure and logical conditional statements.

Task 4: Student Scholarship Eligibility Check

AI.PY

my-app > AI.PY > ...

```
1   students = [
2       {"name": "Nipun", "score": 80},
3       {"name": "Kiran", "score": 65},
4       {"name": "Raju", "score": 90},
5       {"name": "Neha", "score": 72}
6   ]
7
8   i = 0
9   while i < len(students):
10      if students[i]["score"] > 75:
11          print(students[i]["name"], "is eligible for scholarship")
12      i += 1
13
```

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
ython.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2025.18.0-win
32-x64\bundled\libs\debugpy\launcher' '52683' '--' 'c:\Users\Admin\OneDrive\D
ocuments\Desktop\Desktop\full stack\my-app\AI.PY'
Nipun is eligible for scholarship
Raju is eligible for scholarship
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\full stack>
```

Python Deb...

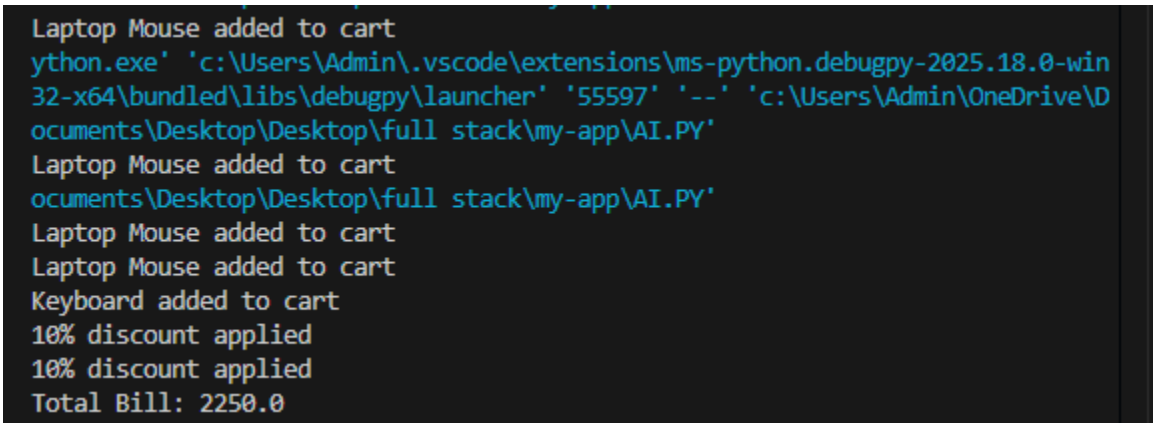
powershell

Summary: A *Scholarship Eligibility Checker* used a while loop to iterate through student records and filter eligible candidates. This task demonstrated AI assistance in list traversal and condition-based filtering

Task 5: Online Shopping Cart Module

```
my-app > AI.PY > ShoppingCart
13  class ShoppingCart:
14      def __init__(self):
15          self.items = []  # each item: {name, price, quantity}
16
17      # Copilot prompt: add items
18      def add_item(self, name, price, quantity):
19          self.items.append({"name": name, "price": price, "quantity": q
20          print(f"{name} added to cart")
21
22      # Copilot prompt: remove items
23      def remove_item(self, name):
24          for item in self.items:
25              if item["name"] == name:
26                  self.items.remove(item)
27                  print(f"{name} removed")
28                  return
29          print("Item not found")
30  # Copilot prompt: calculate total using loop + apply discount
31      def calculate_total(self):
32          total = 0
33          for item in self.items:
34              total += item["price"] * item["quantity"]
32          total = 0
33          for item in self.items:
34              total += item["price"] * item["quantity"]
35
36          if total > 2000:
37              discount = total * 0.10
38              total -= discount
39              print("10% discount applied")
40
41          return total
42  cart = ShoppingCart()
43  cart.add_item("Laptop Mouse", 500, 2)
44  cart.add_item("Keyboard", 1500, 1)
45  print("Total Bill:", cart.calculate_total())
46
```

Output:



Summary: A *Shopping Cart System* was developed where AI helped generate methods to add/remove items, compute totals using loops, and apply discounts using conditions. This showed how AI can help build complete real-world modules efficiently.