AI assisted coding

ASSIGNMENT-10.3

2403A51266

Yashwanth

B-12

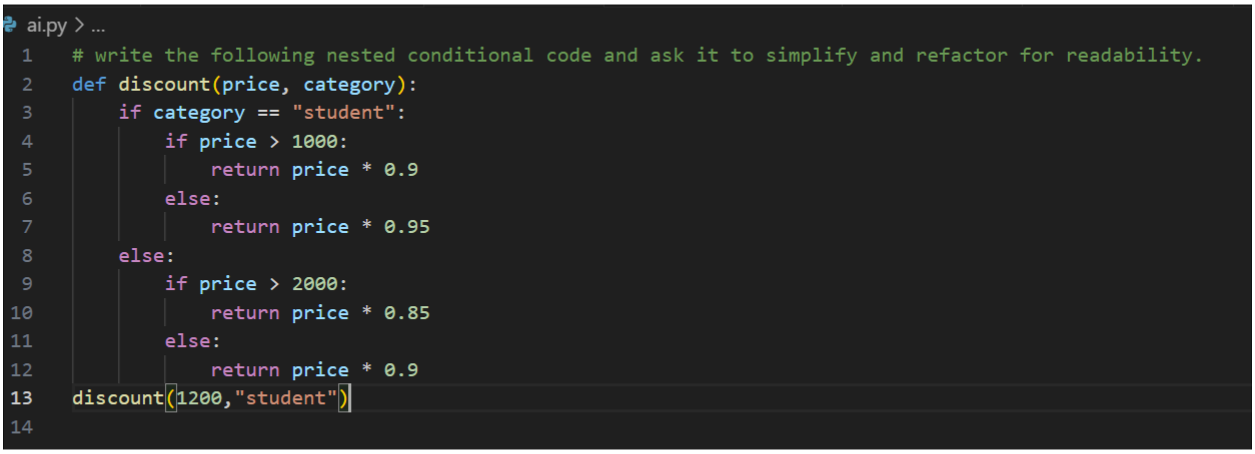
TASK-1:

Provide AI with the following nested conditional code and ask it to simplify and refactor for readability

PROMPT

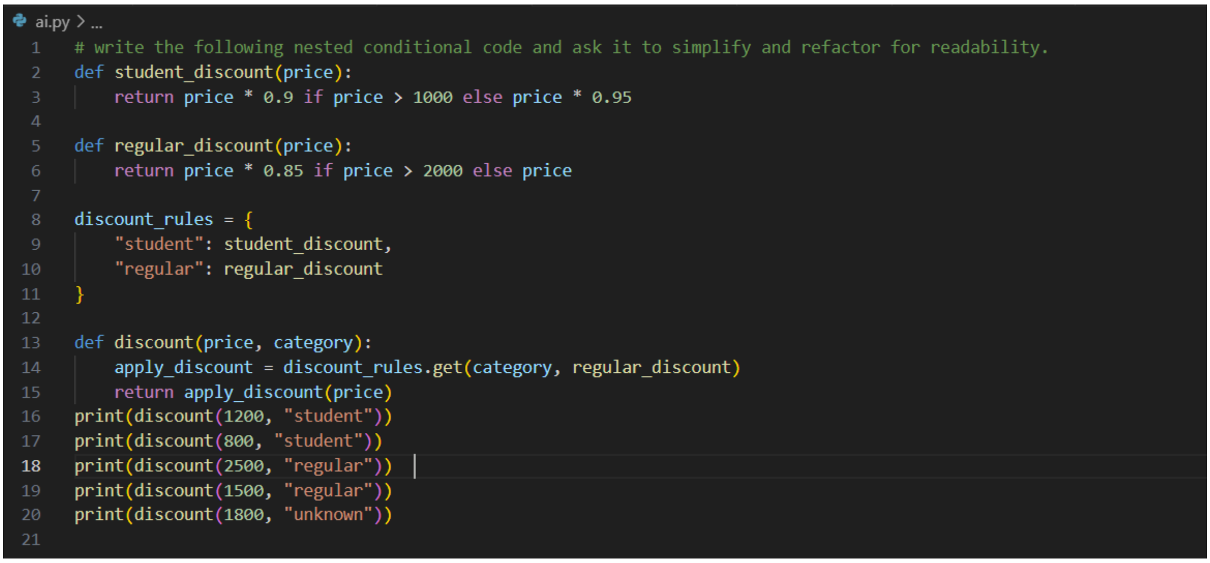
write the following nested conditional code and ask it to simplify and refactor for readability.

Nested code:



```python
# write the following nested conditional code and ask it to simplify and refactor for readability.
def discount(price, category):
    if category == "student":
        if price > 1000:
            return price * 0.9
        else:
            return price * 0.95
    else:
        if price > 2000:
            return price * 0.85
        else:
            return price * 0.9
discount(1200,"student")
```

Output code:



```python
# write the following nested conditional code and ask it to simplify and refactor for readability.
def student_discount(price):
    return price * 0.9 if price > 1000 else price * 0.95

def regular_discount(price):
    return price * 0.85 if price > 2000 else price

discount_rules = {
    "student": student_discount,
    "regular": regular_discount
}

def discount(price, category):
    apply_discount = discount_rules.get(category, regular_discount)
    return apply_discount(price)
print(discount(1200, "student"))
print(discount(800, "student"))
print(discount(2500, "regular"))
print(discount(1500, "regular"))
print(discount(1800, "unknown"))
```

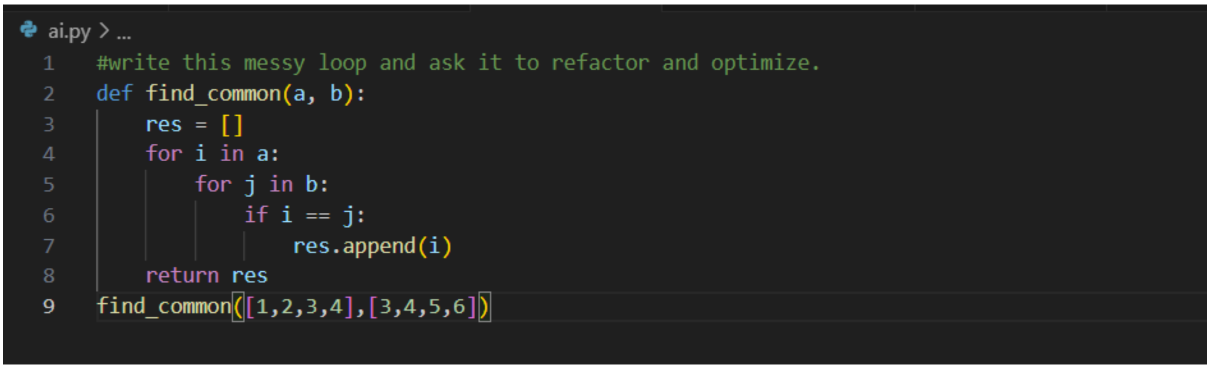
TASK-2 :

Give AI this messy loop and ask it to refactor and optimize.

PROMPT

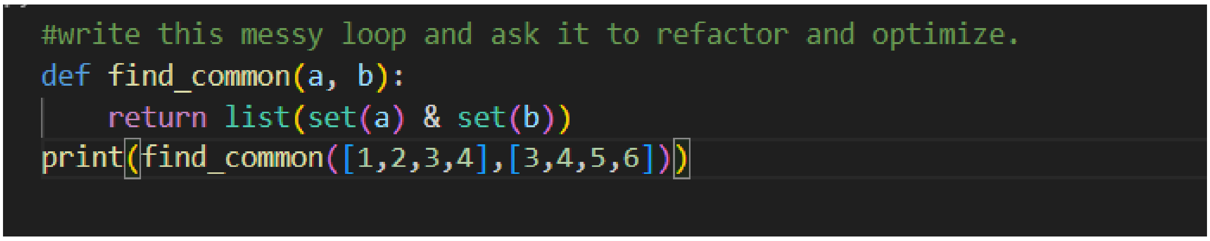
#write this messy loop and ask it to refactor and optimize.

Messy code:



```python
#write this messy loop and ask it to refactor and optimize.
def find_common(a, b):
    res = []
    for i in a:
        for j in b:
            if i == j:
                res.append(i)
    return res
find_common([1,2,3,4],[3,4,5,6])
```

Output code :



```
#write this messy loop and ask it to refactor and optimize.
def find_common(a, b):
    return list(set(a) & set(b))
print(find_common([1,2,3,4],[3,4,5,6]))
```

TASK-3:

Provide this class with poor readability and ask AI to improve:

- Naming conventions
- Encapsulation
- Readability & maintainability

PROMPT

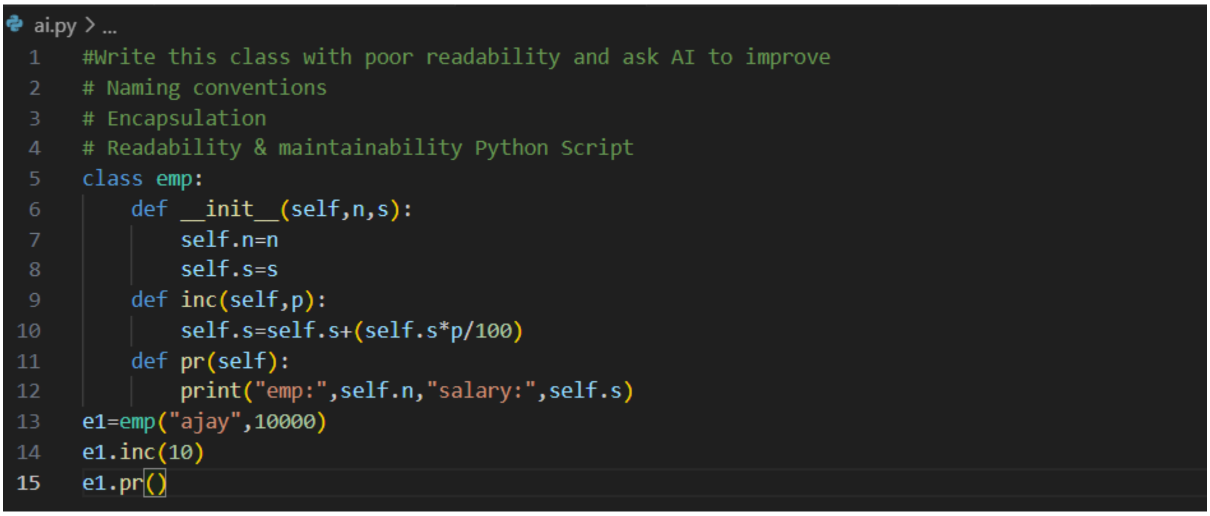
#Write this class with poor readability and ask AI to improve

# Naming conventions

# Encapsulation

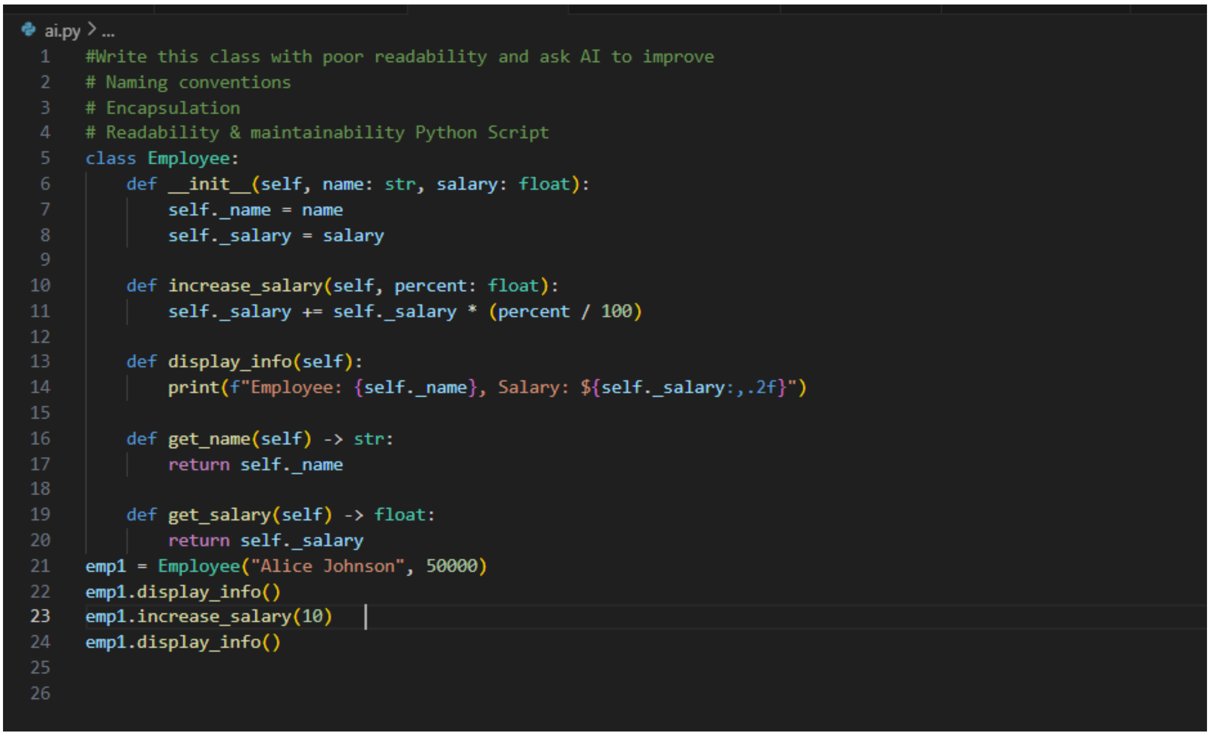
# Readability & maintainability

Poor readability code:



```python
#Write this class with poor readability and ask AI to improve
# Naming conventions
# Encapsulation
# Readability & maintainability Python Script
class emp:
    def __init__(self,n,s):
        self.n=n
        self.s=s
    def inc(self,p):
        self.s=self.s+(self.s*p/100)
    def pr(self):
        print("emp:",self.n,"salary:",self.s)
e1=emp("ajay",10000)
e1.inc(10)
e1.pr()
```

Output code :



```python
#Write this class with poor readability and ask AI to improve
# Naming conventions
# Encapsulation
# Readability & maintainability Python Script
class Employee:
    def __init__(self, name: str, salary: float):
        self._name = name
        self._salary = salary

    def increase_salary(self, percent: float):
        self._salary += self._salary * (percent / 100)

    def display_info(self):
        print(f"Employee: {self._name}, Salary: ${self._salary:,.2f}")

    def get_name(self) -> str:
        return self._name

    def get_salary(self) -> float:
        return self._salary
emp1 = Employee("Alice Johnson", 50000)
emp1.display_info()
emp1.increase_salary(10)
emp1.display_info()


```

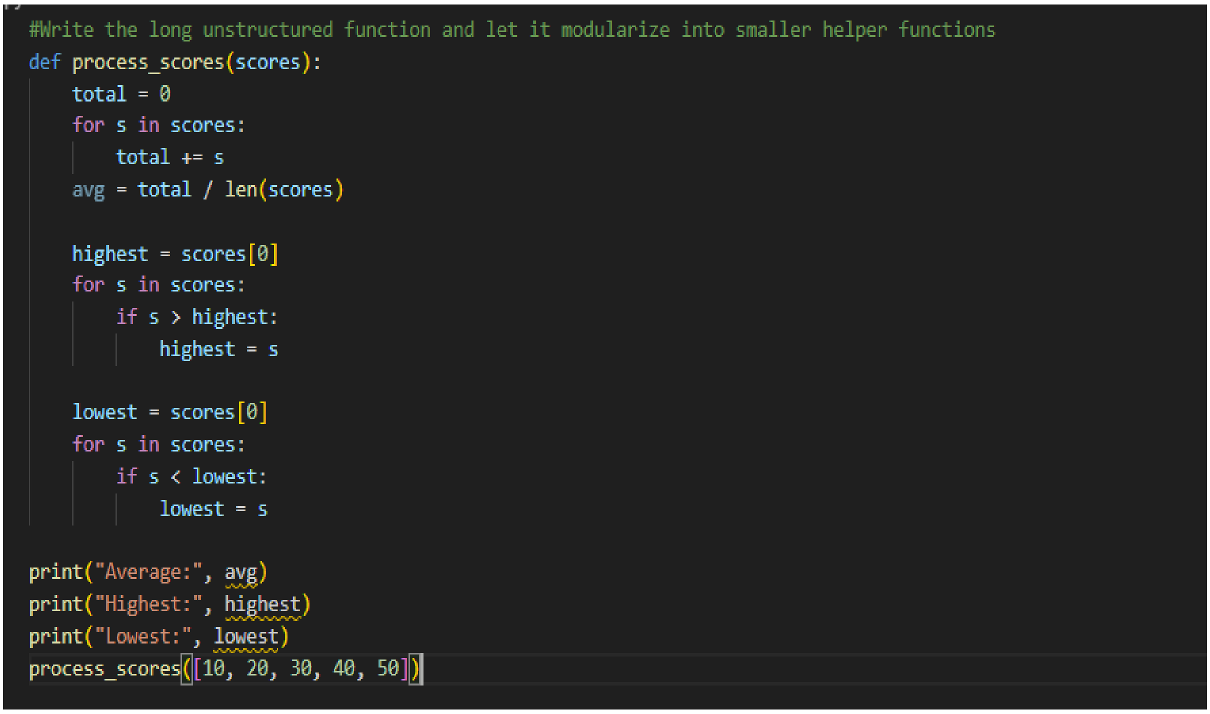
TASK-4 :

Give AI this long unstructured function and let it modularize into smaller helper functions

Prompt

Write the long unstructured function and let it modularize into smaller helper functions

Unstructured function :



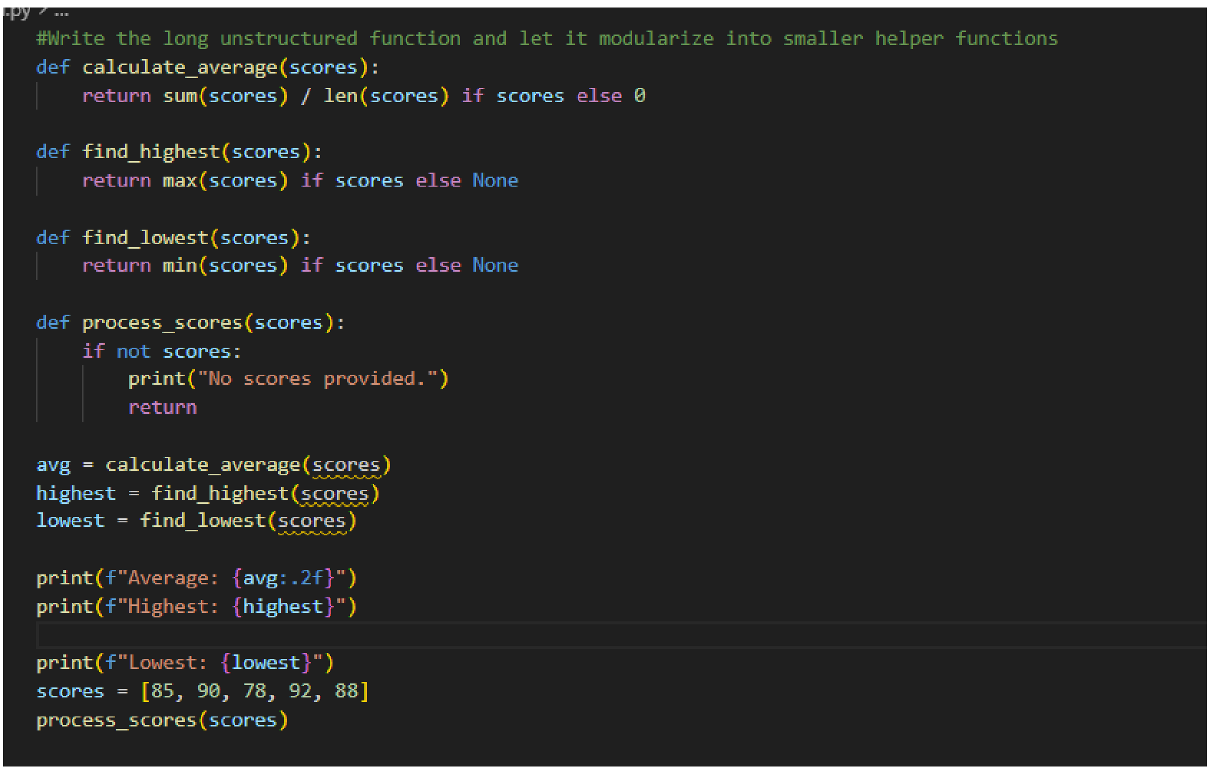
```
#Write the long unstructured function and let it modularize into smaller helper functions
def process_scores(scores):
    total = 0
    for s in scores:
        total += s
    avg = total / len(scores)

    highest = scores[0]
    for s in scores:
        if s > highest:
            highest = s

    lowest = scores[0]
    for s in scores:
        if s < lowest:
            lowest = s

print("Average:", avg)
print("Highest:", highest)
print("Lowest:", lowest)
process_scores([10, 20, 30, 40, 50])
```

Output code :



```python
#Write the long unstructured function and let it modularize into smaller helper functions
def calculate_average(scores):
    return sum(scores) / len(scores) if scores else 0

def find_highest(scores):
    return max(scores) if scores else None

def find_lowest(scores):
    return min(scores) if scores else None

def process_scores(scores):
    if not scores:
        print("No scores provided.")
        return

avg = calculate_average(scores)
highest = find_highest(scores)
lowest = find_lowest(scores)

print(f"Average: {avg:.2f}")
print(f"Highest: {highest}")

print(f"Lowest: {lowest}")
scores = [85, 90, 78, 92, 88]
process_scores(scores)
```

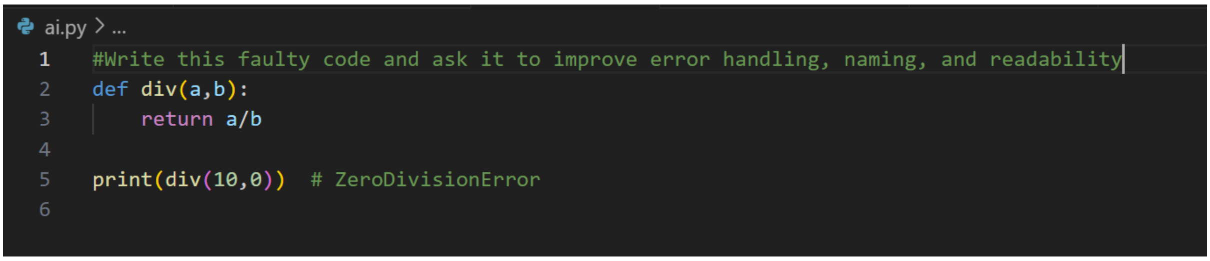
TASK-5 :

Provide AI with this faulty code and ask it to improve error handling, naming, and readability

PROMPT

#Write this faulty code and ask it to improve error handling, naming, and readability

Faulty code :

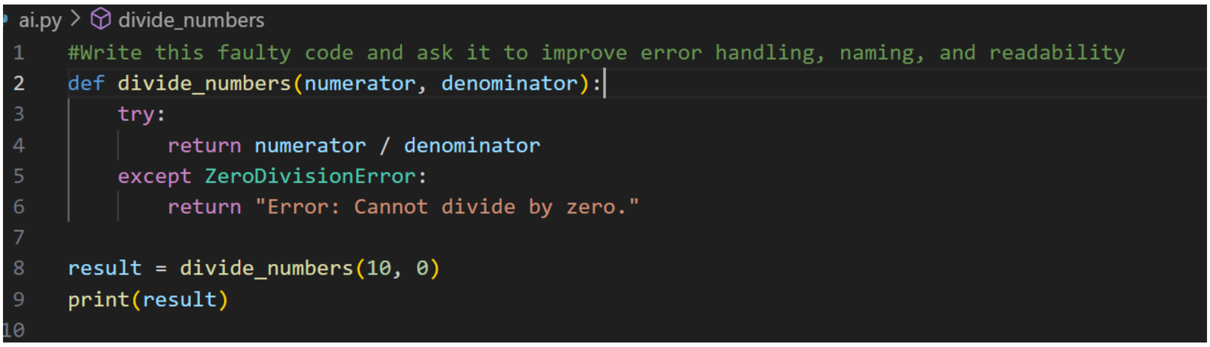


```python
#Write this faulty code and ask it to improve error handling, naming, and readability
def div(a,b):
    return a/b

print(div(10,0))  # ZeroDivisionError

```

Output code :



```python
#Write this faulty code and ask it to improve error handling, naming, and readability
def divide_numbers(numerator, denominator):
    try:
        return numerator / denominator
    except ZeroDivisionError:
        return "Error: Cannot divide by zero."

result = divide_numbers(10, 0)
print(result)

```

TASK-6:

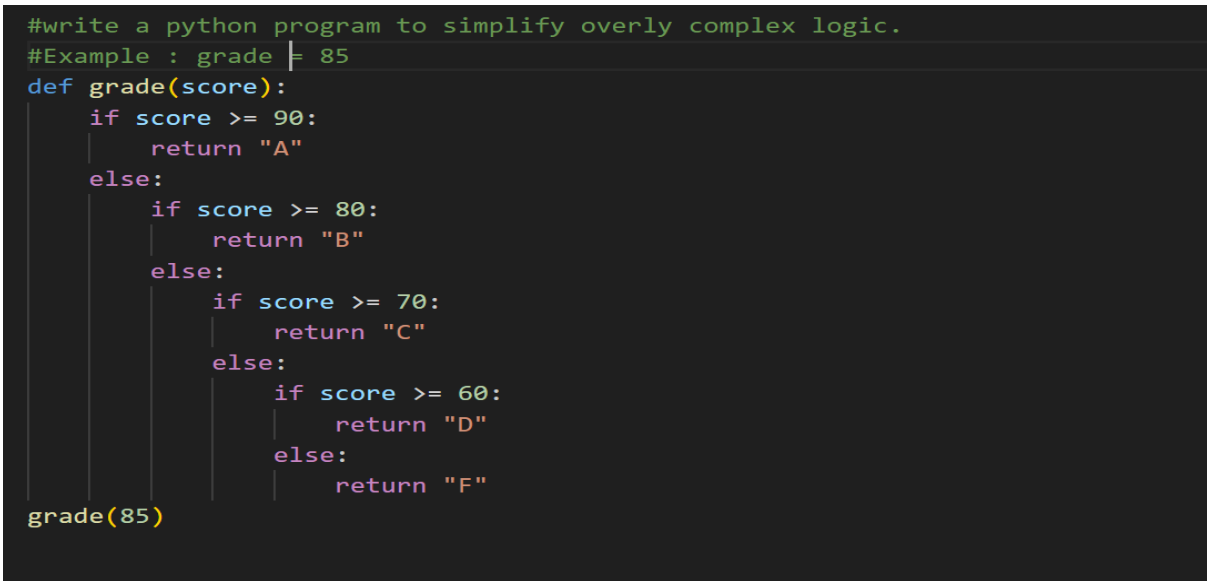
Use AI to simplify overly complex logic.
Sample Input Code

PROMPT

# write a python program to simplify overly complex logic.
Sample Input Code

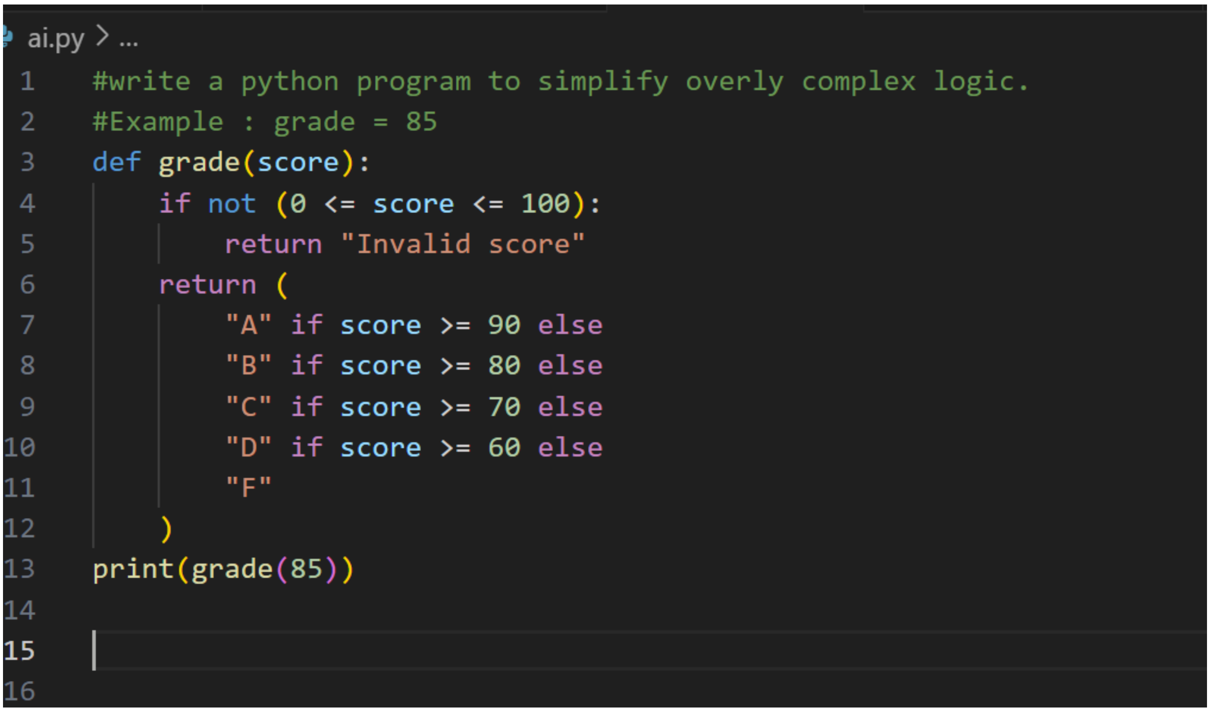
#Example : grade = 85

Complex logic :



```python
#write a python program to simplify overly complex logic.
#Example : grade = 85
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"
grade(85)
```

Output code :



```python
#write a python program to simplify overly complex logic.
#Example : grade = 85
def grade(score):
    if not (0 <= score <= 100):
        return "Invalid score"
    return (
        "A" if score >= 90 else
        "B" if score >= 80 else
        "C" if score >= 70 else
        "D" if score >= 60 else
        "F"
    )
print(grade(85))


```