|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOLOFCOMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENTOFCOMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **AssignmentType: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | 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) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **DateandDay**  **of Assignment** | | | Week5 - WednesDay | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:10.1**(Presentassignmentnumber)/**24**(Totalnumberofassignments) | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***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. * Apply prompt engineering for targeted improvements. * Evaluate AI-generated suggestions against PEP 8 standards and software engineering best practices   **Task Description #1 – Refactor Nested Conditionals**  Task: Provide AI with the following nested conditional code and ask it to simplify and refactor for readability.  **Python script.**  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  **Expected Output:**   * Refactored code using cleaner logic, possibly a dictionary or separate helper functions.   **Task Description #2 – Optimize Redundant Loops**  Task: Give AI this messy loop and ask it to refactor and optimize.  **Python script**  def find\_common(a, b):  res = []  for i in a:  for j in b:  if i == j:  res.append(i)  return res  **Expected Output:**  Cleaner version using Python sets (set(a) & set(b))  **Task Description #3 – Improve Class Design**  **Task:** Provide 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)  **Expected Output:**   * Employee class with meaningful methods (increase\_salary, display\_info), formatted output, and added docstrings.   **Task Description #4 – Modularize Long Function**  Task: Give AI this long unstructured function and let it modularize into smaller helper functions.  **Python Script**  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)  **Expected Output:**   * Split into functions: calculate\_average, find\_highest, find\_lowest. * Clean main process\_scores() using helper functions.   **Task Description #5 – Code Review on Error Handling**  Task: Provide AI with this faulty code and ask it to improve error handling, naming, and readability.  **Python Script**  def div(a,b):  return a/b  print(div(10,0))  **Expected Output:**   * Function with proper error handling using try-except. * Better naming (divide\_numbers). * AI-generated docstring explaining error handling.   **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:  if score >= 70:  return "C"  else:  if score >= 60:  return "D"  else:  return "F"  Expected Output:   * Cleaner logic using elif or dictionary mapping. | | | | | | Week5 - Monday |  |

**Task Description #1 – Refactor Nested Conditionals**

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

**Python script.**

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

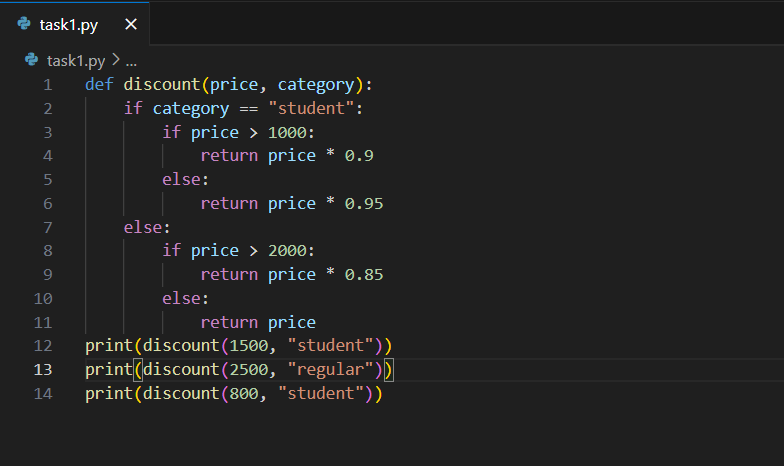
else:

return price

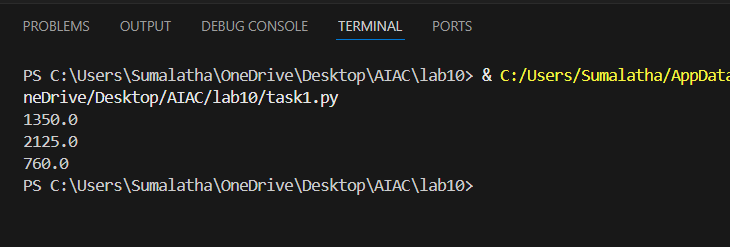
**Expected Output:**

* Refactored code using cleaner logic, possibly a dictionary or separate helper functions.

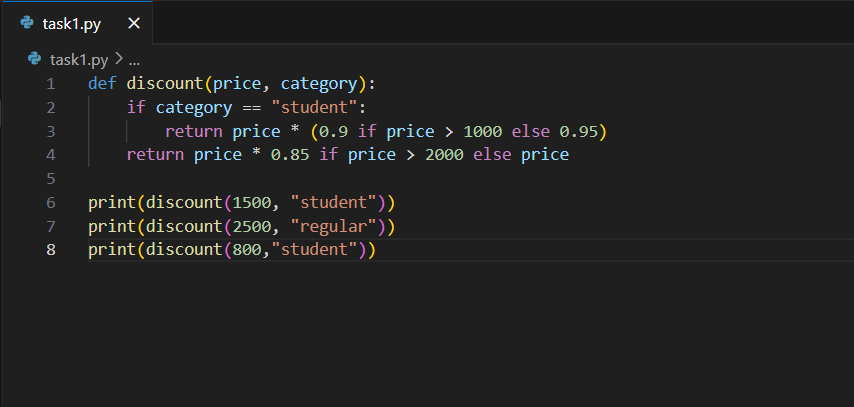
Given code:



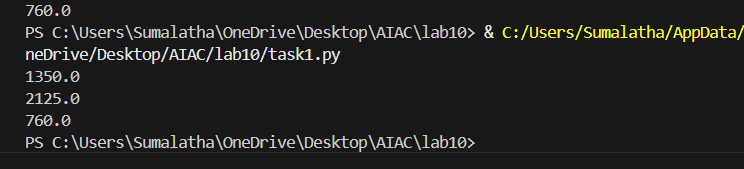
Output:



Reduced code:



Output:



**Task Description #2 – Optimize Redundant Loops**

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

**Python script**

def find\_common(a, b):

res = []

for i in a:

for j in b:

if i == j:

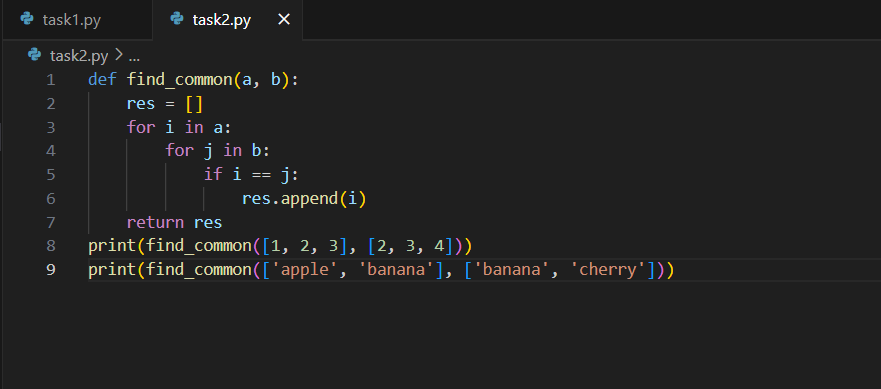
res.append(i)

return res

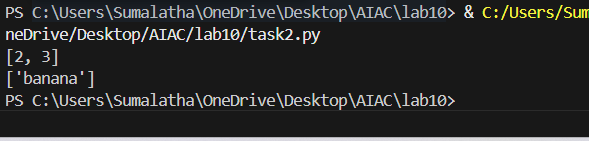
**Expected Output:**

Cleaner version using Python sets (set(a) & set(b)

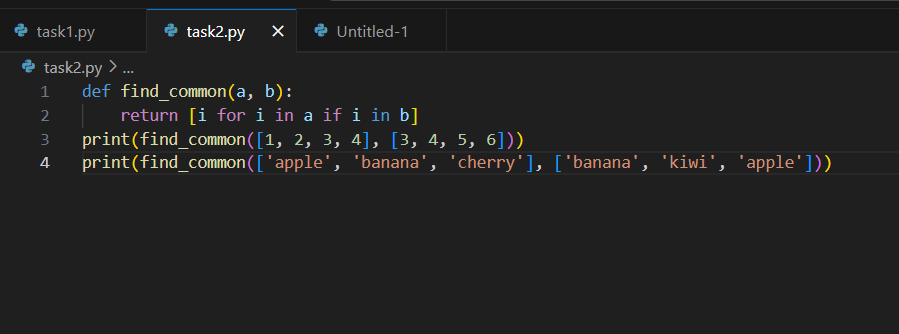
Given code:



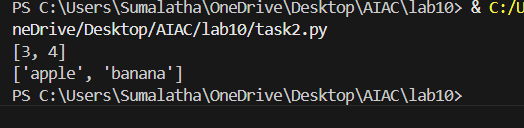
Output:



Reduced code:



Output:



**Task Description #3 – Improve Class Design**

**Task:** Provide 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)

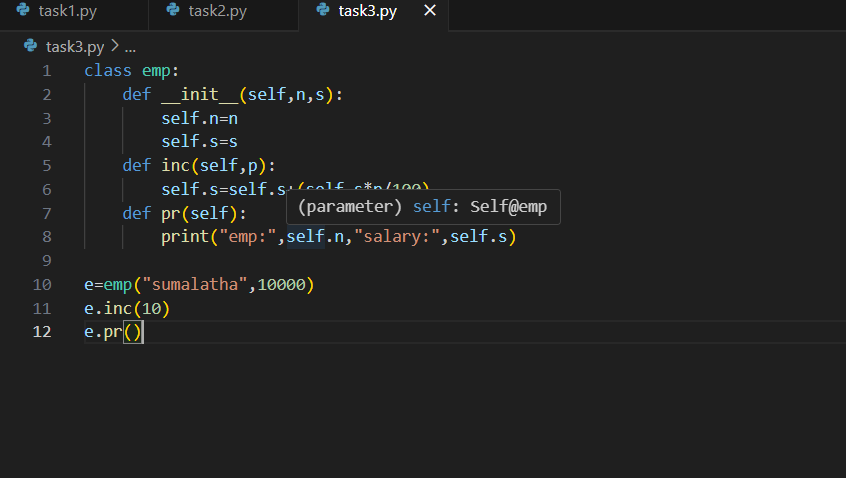
def pr(self):

print("emp:",self.n,"salary:",self.s)

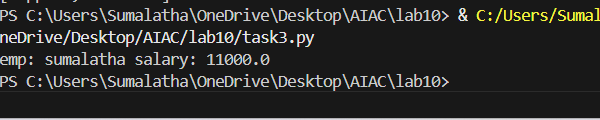
**Expected Output:**

Employee class with meaningful methods (increase\_salary, display\_info), formatted output, and added docstrings.

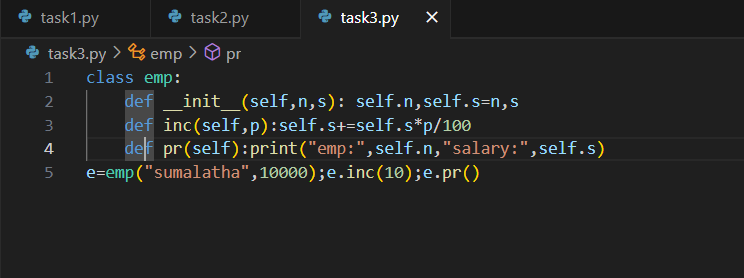
Given code:



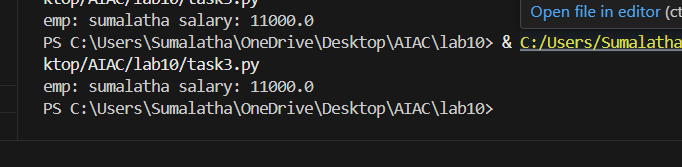
Output:



Reduced code:



Output:



**Task Description #4 – Modularize Long Function**

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

**Python Script**

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)

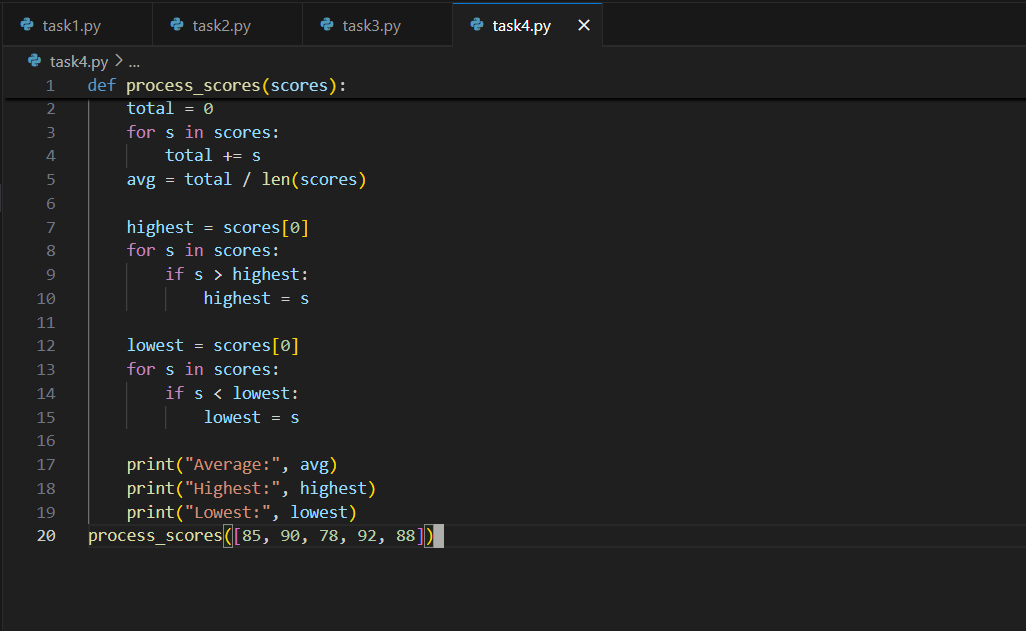
print("Lowest:", lowest)

**Expected Output:**

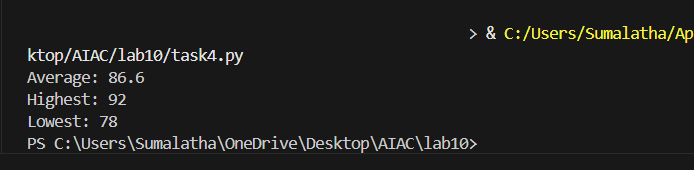
* Split into functions: calculate\_average, find\_highest, find\_lowest.

Clean main process\_scores() using helper functions

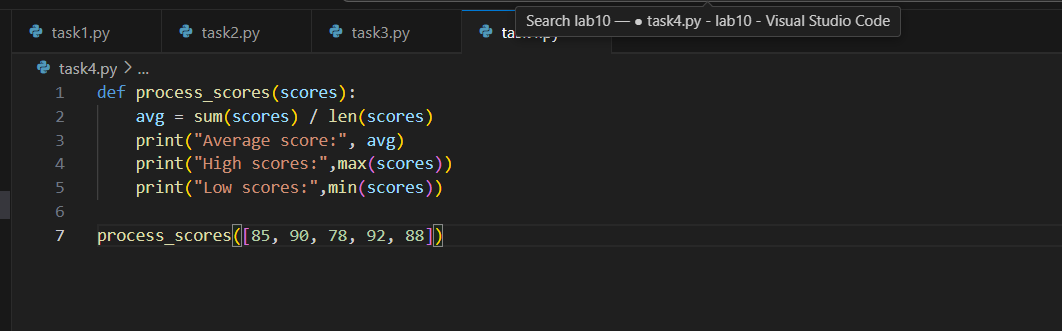
Given code:



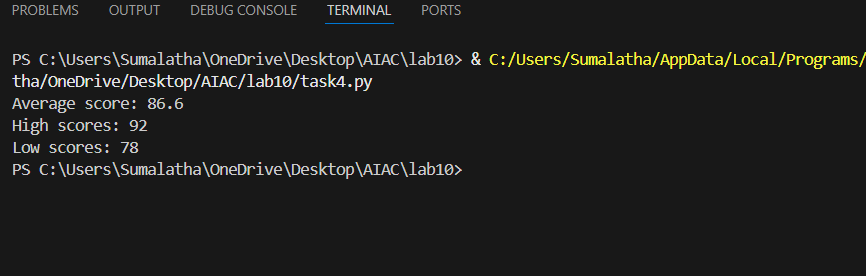
Output:



Reduced code:



Output:



**Task Description #5 – Code Review on Error Handling**

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

**Python Script**

def div(a,b):

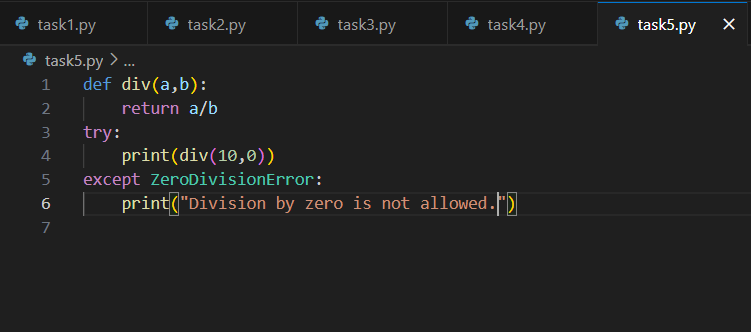
return a/b

print(div(10,0))

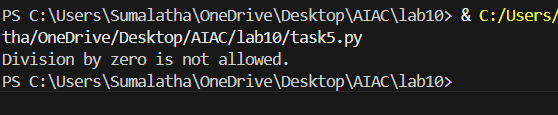
**Expected Output:**

* Function with proper error handling using try-except.
* Better naming (divide\_numbers).
* AI-generated docstring explaining error handling.

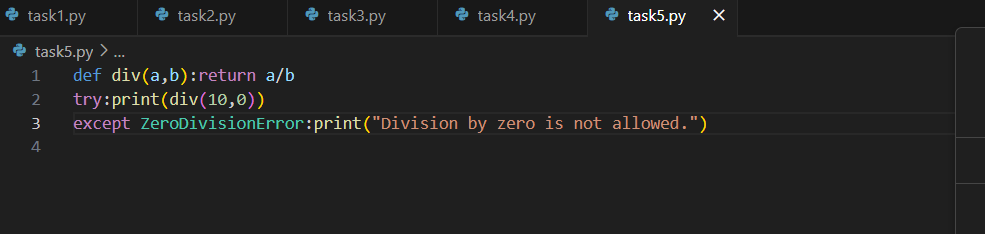
Given code:



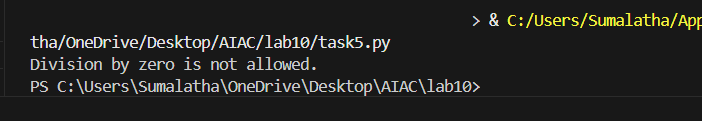
Output:



Reduced code:



Output:



**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:

if score >= 70:

return "C"

else:

if score >= 60:

return "D"

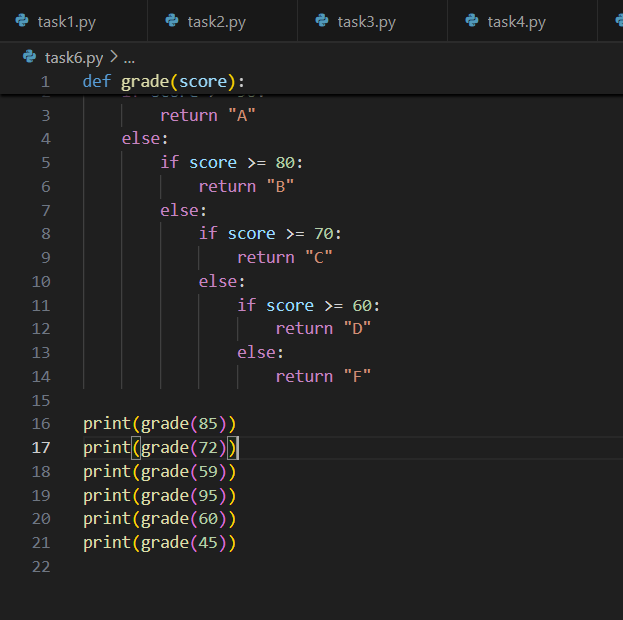
else:

return "F"

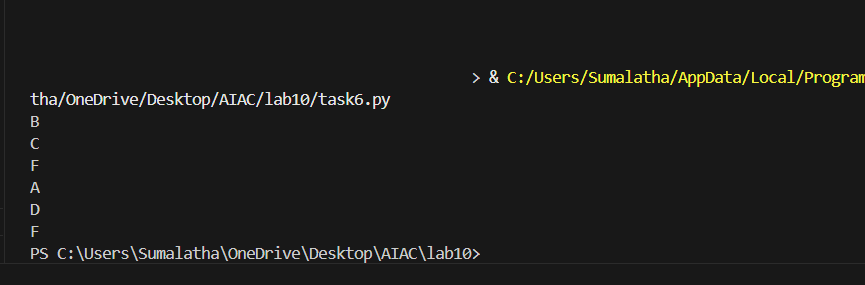
Expected Output:

Cleaner logic using elif or dictionary mapping

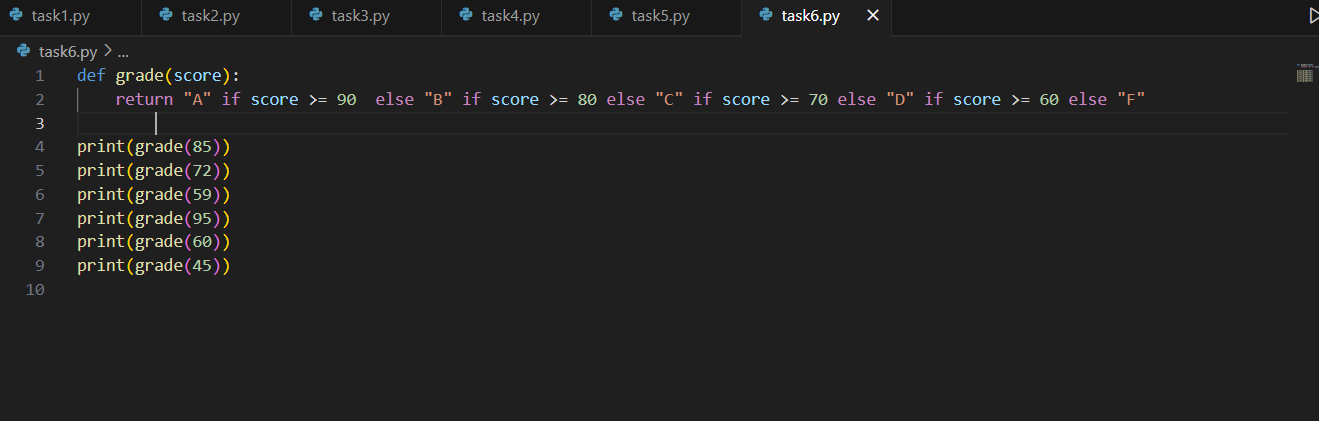
Given code:



Output:



Reduced code:



Output:

