

Ai Assisted Coding

Assignment – 5.1 & 6

Name : G.Samshray

Batch : 23

Ht.no : 2303A51670

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.

```
Employee.py 2 ...
1 class Employee:
2     def __init__(self, empid, empname, designation, basic_salary, exp):
3         self.empid = empid
4         self.empname = empname
5         self.designation = designation
6         self.basic_salary = basic_salary
7         self.exp = exp
8     def display_details(self):
9         print(f"Employee ID: {self.empid}")
10        print(f"Employee Name: {self.empname}")
11        print(f"Designation: {self.designation}")
12        print(f"Basic Salary: {self.basic_salary}")
13        print(f"Experience: {self.exp} years")
14    def calculate_allowance(self):
15        if self.exp < 5:
16            allowance = 0.05 * self.basic_salary
17        elif 5 <= self.exp < 10:
18            allowance = 0.10 * self.basic_salary
19        else:
20            allowance = 0.20 * self.basic_salary
21        print(f"Allowance: {allowance}")
22        print(f"Total Salary: {self.basic_salary + allowance}")
23    # Example usage
24    emp = Employee(101, "John Doe", "Software Engineer", 60000, 7)
25    emp.display_details()
26    allowance = emp.calculate_allowance()

PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL  PORTS

PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:\Program Files\Python312\python.exe" c:/Users/Ganne/OneDrive/Desktop/python/Employee.py
Employee ID: 101
Employee Name: John Doe
Designation: Software Engineer
Basic Salary: 60000
Experience: 7 years
Allowance: 6000.0
Total Salary: 66000.0
PS C:\Users\Ganne\OneDrive\Desktop\python>
```

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 \leq 100 \rightarrow ₹5 per unit
- 101 to 300 units \rightarrow ₹7 per unit
- More than 300 units \rightarrow ₹10 per unit

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

```
Electricity.py > ElectricityBill > calculate_bill
1 class ElectricityBill:
2     def __init__(self, customer_id, customer_name, units_consumed):
3         self.customer_id = customer_id
4         self.customer_name = customer_name
5         self.units_consumed = units_consumed
6
7     def display_details(self):
8         print(f"Customer ID: {self.customer_id}")
9         print(f"Customer Name: {self.customer_name}")
10        print(f"Units Consumed: {self.units_consumed}")
11
12    def calculate_bill(self):
13        if self.units_consumed <= 100:
14            bill_amount = self.units_consumed * 5
15        elif self.units_consumed <= 300:
16            bill_amount = self.units_consumed * 7
17        else:
18            bill_amount = self.units_consumed * 10
19
20        print(f"Bill Amount: {bill_amount}")
21        print(f"Total Amount to be Paid: {bill_amount}")
22
23    # Example usage
24    bill = ElectricityBill(201, "Alice Smith", 250)
25    bill.display_details()
26    bill_amount = bill.calculate_bill()
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python112/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/Electricity.py
Customer ID: 201
Customer Name: Alice Smith
Units Consumed: 250
Bill Amount: 1750
Total Amount to be Paid: 1750
PS C:\Users\Ganne\OneDrive\Desktop\python>
```

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.

```
Product_Discount_Calculator.py
1 class Product:
2     def __init__(self, product_id, product_name, price, category):
3         self.product_id = product_id
4         self.product_name = product_name
5         self.price = price
6         self.category = category
7
8     def display_details(self):
9         print(f"Product ID: {self.product_id}")
10        print(f"Product Name: {self.product_name}")
11        print(f"Price: {self.price}")
12        print(f"Category: {self.category}")
13
14    def calculate_discount(self):
15        if self.category.lower() == "electronics":
16            discount = 0.10 * self.price
17        elif self.category.lower() == "clothing":
18            discount = 0.15 * self.price
19        else:
20            discount = 0.05 * self.price
21
22        print(f"Discount: {discount}")
23        print(f"Price after Discount: {self.price - discount}")
24        print(f"Total Price to be Paid: {self.price - discount}")
25
26    # Example usage
27    product = Product(301, "Smartphone", 50000, "Electronics")
28    product.display_details()
29    product.calculate_discount()
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python112/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/Product_Discount_Calc.py
Product ID: 301
Product Name: Smartphone
Price: 50000
Category: Electronics
Discount: 5000.0
Price after Discount: 45000.0
Total Price to be Paid: 45000.0
PS C:\Users\Ganne\OneDrive\Desktop\python>
```

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 $\leq 5 \rightarrow$ ₹5 per day
- 6 to 10 days late \rightarrow ₹7 per day
- More than 10 days late \rightarrow ₹10 per day

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

```
book_late_fee.py >_
1 class LibraryBook:
2     def __init__(self, BookID, Title, Author, Borrower, Days_late):
3         self.BookID = BookID
4         self.Title = Title
5         self.Author = Author
6         self.Borrower = Borrower
7         self.Days_late = Days_late
8     def display_details(self):
9         print(f"Book ID: {self.BookID}")
10        print(f"Title: {self.Title}")
11        print(f"Author: {self.Author}")
12        print(f"Borrower: {self.Borrower}")
13        print(f"Days Late: {self.Days_late}")
14    def calculate_late_fee(self):
15        if self.Days_late <= 5:
16            late_fee = self.Days_late * 5
17        elif 6 <= self.Days_late <= 10:
18            late_fee = self.Days_late * 7
19        else:
20            late_fee = self.Days_late * 10
21        print(f"Late Fee: {late_fee}")
22        print(f"Total Amount to be Paid: {late_fee}")
23    # Example usage
24    book = LibraryBook(401, "The Great Gatsby", "F. Scott Fitzgerald", "Emily Clark", 8)
25    book.display_details()
26    book.calculate_late_fee()
```

PROBLEMS DESIGN CONSOLE OUTPUT TERMINAL PORTS

```
* PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python112/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/book_late_fee.py
Book ID: 401
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Borrower: Emily Clark
Days Late: 8
Late Fee: 56
Total Amount to be Paid: 56
* PS C:\Users\Ganne\OneDrive\Desktop\python>
```

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.

```

# student_performance_report.py
1 # write a program that defines a function student_report(student_data) that accepts a dictionary containing student names and their marks. This function should
2 # calculate avg score of each student
3 # determine pass/fail status (pass if avg >= 40)
4 # return a summary report as a list of dictionaries.
5
6 def student_report(student_data):
7     report = []
8     for student, marks in student_data.items():
9         avg_score = sum(marks) / len(marks)
10        status = "Pass" if avg_score >= 40 else "Fail"
11        report.append({
12            "Student": student,
13            "Average Score": avg_score,
14            "Status": status
15        })
16    return report
17
18 # Example usage
19 students = {
20     "Alice": [45, 78, 88],
21     "Bob": [35, 40, 30],
22     "Charlie": [50, 60, 70]
23 }
24 report = student_report(students)
25 for student_summary in report:
26     print(student_summary)

```

PROBLEMS DEBUG CONSOLE SOURCE TERMINAL POKT

```

PS C:\Users\Ganmo\OneDrive\Desktop\python> & "C:/Program Files/Python112/python.exe" C:/Users/Ganmo/OneDrive/Desktop/python/student_performance_report.py
[{"Student": "Alice", "Average Score": 70.33333333333333, "Status": "Pass"}]
[{"Student": "Bob", "Average Score": 35.0, "Status": "Fail"}]
[{"Student": "Charlie", "Average Score": 60.0, "Status": "Pass"}]
PS C:\Users\Ganmo\OneDrive\Desktop\python>

```

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.

```
taxi_fee_calculation.py > ...
1 class TaxiRide:
2     def __init__(self, ride_id, driver_name, distance_km, waiting_time_min):
3         self.ride_id = ride_id
4         self.driver_name = driver_name
5         self.distance_km = distance_km
6         self.waiting_time_min = waiting_time_min
7     def display_details(self):
8         print(f"Ride ID: {self.ride_id}")
9         print(f"Driver Name: {self.driver_name}")
10        print(f"Distance (km): {self.distance_km}")
11        print(f"Waiting Time (min): {self.waiting_time_min}")
12    def calculate_fare(self):
13        if self.distance_km <= 10:
14            fare = self.distance_km * 15
15        elif 11 <= self.distance_km <= 20:
16            fare = self.distance_km * 12
17        else:
18            fare = self.distance_km * 10
19        waiting_charge = self.waiting_time_min * 2
20        total_fare = fare + waiting_charge
21        print(f"Fare: {fare}")
22        print(f"Waiting Charge: {waiting_charge}")
23        print(f"Total Fare to be Paid: {total_fare}")
24    # Example usage
25    ride = TaxiRide(501, "Michael Scott", 18, 15)
26    ride.display_details()
27    ride.calculate_fare()
```

PROBLEMS DEVOPS CONSOLE OUTPUT TERMINAL PLOTS

```
* PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python12/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/taxi_fee_calculation.py
Ride ID: 501
Driver Name: Michael Scott
Distance (km): 18
Waiting Time (min): 15
Fare: 216
Waiting Charge: 30
Total Fare to be Paid: 246
* PS C:\Users\Ganne\OneDrive\Desktop\python>
```

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.

```
# statistics_sub.py
1 # generate a code that defines a function named statistics_subject(scores_list) that accepts a list of 50 student scores and computes key performance statistics. The function should
   return:
2 # - highest score
3 # - lowest score
4 # - average score
5 # - number of students passed(scores>40)
6 # - number of students failed(scores<40)
7 def statistics_subject(scores_list):
8     highest_score = max(scores_list)
9     lowest_score = min(scores_list)
10    average_score = sum(scores_list) / len(scores_list)
11    passed_count = sum(1 for score in scores_list if score >= 40)
12    failed_count = sum(1 for score in scores_list if score < 40)
13
14    return {
15        "Highest Score": highest_score,
16        "Lowest Score": lowest_score,
17        "Average Score": average_score,
18        "Number of Students Passed": passed_count,
19        "Number of Students Failed": failed_count
20    }
21
22 # Example usage
23 scores = [55, 78, 34, 90, 67, 45, 21, 89, 100, 34, 76, 84, 61, 47, 52, 29, 60, 72, 81, 44, 39, 58, 56, 88, 92, 15, 27, 35, 41, 58, 73, 88, 95, 12, 22, 31, 49, 64, 70, 83, 97, 16, 26, 33,
42, 57, 48, 75, 60, 91, 11, 19, 38, 85, 43, 52, 46, 74, 79, 94, 13, 21, 30]
25 report = statistics_subject(scores)
26 for key, value in report.items():
27     print(f"{key}: {value}")
28
```

PROGRAM | SOURCE CODE | OUTPUT | TERMINAL | PLOTS

```
# PS C:\Users\Game\Desktop\python> . ".\Program Files\Python112\python.exe" c:\Users\Game\Desktop\python\statistics_sub_per.py
Highest Score: 100
Lowest Score: 11
Average Score: 54.8875
Number of Students Passed: 43
Number of Students Failed: 7
PS C:\Users\Game\Desktop\python>
```

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.
- Validation that explanations align with runtime behavior.


```

◆ prime.py > _
1  # Generate Python code for two prime-checking methods and explain
2  #- how the optimized version improves performance."
3  # - transparent explanation of time complexity for both methods.
4  # - comparison highlighting efficiency improvements.
5
6  def is_prime_basic(n):
7      """Check if a number is prime using basic method."""
8      if n <= 1:
9          return False
10     for i in range(2, n):
11         if n % i == 0:
12             return False
13     return True
14
15 def is_prime_optimized(n):
16     """Check if a number is prime using optimized method."""
17     if n <= 1:
18         return False
19     if n <= 3:
20         return True
21     if n % 2 == 0 or n % 3 == 0:
22         return False
23     i = 5
24     while i * i <= n:
25         if n % i == 0 or n % (i + 2) == 0:
26             return False
27         i += 6
28     return True

```

```

◆ prime.py > _
28 # Explanation of time complexity:
29 # 1. Basic Method:
30 # - Time complexity: O(n)
31 # - Explanation: In the basic method, we check all numbers from 2 to n
32 #   to see if they divide n evenly. In the worst case, we perform n-2
33 #   checks, leading to linear time complexity.
34 # 2. Optimized Method:
35 # - Time complexity: O(√n)
36 # - Explanation: The optimized method reduces the number of checks
37 #   significantly by eliminating even numbers and checking only up to
38 #   the square root of n. This is because if n is divisible by any number
39 #   greater than its square root, the corresponding factor must be less
40 #   than the square root. Thus, we only need to check up to √n,
41 #   leading to a much faster algorithm for large n.
42 # Comparison:
43 # The optimized method is significantly more efficient than the basic
44 # method, especially for large numbers. While the basic method's linear
45 # time complexity can become impractical for large n, the optimized
46 # method's square root time complexity allows it to handle much larger
47 # numbers efficiently. This makes the optimized method preferable for
48 # prime checking in real-world applications.
49 # Example usage
50 number = 29
51 print(f"Is {number} prime (basic)? {is_prime_basic(number)}")
52 print(f"Is {number} prime (optimized)? {is_prime_optimized(number)}")

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python312/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/prime.py
PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python312/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/prime.py
Is 29 prime (basic)? True
Is 29 prime (optimized)? True
PS C:\Users\Ganne\OneDrive\Desktop\python>

```

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.

```
fib.py > fibonacci
1  # write a code to generate a recursive function to calculate fibonacci numbers.
2  # - add clear comments explaining recursion.
3  # - also explain base cases and recursive calls.
4  # - verification that explanation matches actual execution.
5  def fibonacci(n):
6      """
7      Calculate the nth Fibonacci number using recursion.
8
9      The Fibonacci sequence is defined as:
10     F(0) = 0 (base case)
11     F(1) = 1 (base case)
12     F(n) = F(n-1) + F(n-2) for n > 1 (recursive case)
13
14     Parameters:
15     n (int): The position in the Fibonacci sequence to calculate.
16
17     Returns:
18     int: The nth Fibonacci number.
19     """
20     # Base cases
21     if n == 0:
22         return 0
23     elif n == 1:
24         return 1
25     else:
26         # Recursive case: sum of the two preceding numbers
27         return fibonacci(n - 1) + fibonacci(n - 2)
28 # Example usage and verification
29 n = 6
30 print(f"The {n}th Fibonacci number is: {fibonacci(n)}")
```

```

• Rupy > fibonacci
28 # Example usage and verification
29 n = 6
30 print(f"The {n}th Fibonacci number is: {fibonacci(n)}")
31 # Explanation:
32 # When we call fibonacci(6), the function checks if n is 0 or 1. Since it's neither, it proceeds to the recursive case:
33 # fibonacci(6) = fibonacci(5) + fibonacci(4)
34 # This pattern continues, breaking down each call until it reaches the base cases:
35 # fibonacci(1) = 1 and fibonacci(0) = 0.
36 # The results are then combined back up the call stack to produce the final result.
37 # The execution for fibonacci(6) would look like this:
38 # fibonacci(6)
39 # = fibonacci(5) + fibonacci(4)
40 # = (fibonacci(4) + fibonacci(3)) + (fibonacci(3) + fibonacci(2))
41 # = ((fibonacci(3) + fibonacci(2)) + (fibonacci(2) + fibonacci(1))) + ((fibonacci(2) + fibonacci(1)) + (fibonacci(1) + fibonacci(0)))
42 # = ... and so on, until all calls reach the base cases.
43 # The final result is 8, which is the 6th Fibonacci number.

PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL  PORTS
Python

PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python312/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/Fib.py
The 6th Fibonacci number is: 8
PS C:\Users\Ganne\OneDrive\Desktop\python>
```

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.

```

1  # error_handling.py > process_file
2  # Generate a Python program that reads and processes data from a file using try-except blocks. Include meaningful exception handling and clear
  # comments explaining each possible error scenario.
3
4  def process_file(filename):
5      """
6      Reads a file and processes its data.
7      Processing: counts number of lines and words.
8      """
9      try:
10         # Tries to open the file in read mode
11         with open(filename, "r") as file:
12             content = file.read()
13             # Data processing
14             lines = content.splitlines()
15             words = content.split()
16
17             print("File processed successfully.")
18             print("Number of lines:", len(lines))
19             print("Number of words:", len(words))
20             print(lines)
21             print(words)
22
23     except FileNotFoundError:
24         # Raised when the file does not exist at the given path
25         print("Error: The file was not found. Please check the file name or path.")
26

```

```

1  # error_handling.py > process_file
2  def process_file(filename):
3
4      # Raised when the file does not exist at the given path
5      print("Error: The file was not found. Please check the file name or path.")
6
7      except PermissionError:
8          # Raised when the program does not have permission to read the file
9          print("Error: Permission denied. You do not have access to this file.")
10
11     except Exception as e:
12         # Handles any other unexpected runtime errors
13         print("Unexpected error occurred:", e)
14
15     else:
16         # Executes only if no exception occurs
17         print("File reading and processing completed without errors.")
18
19     finally:
20         # Always executes, useful for cleanup actions
21         print("Program execution finished.")
22
23 # Example usage
24 process_file("sample.txt")
25

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\Users\Ganne\OneDrive\Desktop\python> & "C:/Program Files/Python312/python.exe" c:/Users/Ganne/OneDrive/Desktop/python/error_handling.py
File processed successfully.
Number of lines: 3
Number of words: 9
['AI Assisted Coding', 'SR University', 'Roll no : 2303A51670']
['AI', 'Assisted', 'Coding', 'SR', 'University', 'Roll', 'no', ':', '2303A51670']
File reading and processing completed without errors.
Program execution finished.
PS C:\Users\Ganne\OneDrive\Desktop\python>

```