

Assignment- 5.1 & 6

Name: A. Srivani

Ht.No: 2303A51272

Batch: 23

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.

◆ Assignment-1.5 and 6.py > ...

```
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
15    def calculate_allowance(self):
16        if self.exp > 10:
17            allowance = 0.20 * self.basic_salary
18        elif 5 <= self.exp <= 10:
19            allowance = 0.10 * self.basic_salary
20        else:
21            allowance = 0.05 * self.basic_salary
22        print(f"Allowance:{allowance}")
23        print(f"Total Salary: {self.basic_salary+allowance}")
24 empobj1=Employee(101,"Alice","Manger",80000,12)
25 empobj1.display_details()
26 empobj1.calculate_allowance
27 print()
28 empobj2=Employee(102,"vicky","HR",160000,20)
29 empobj2.display_details()
30 empobj2.calculate_allowance
```

Employee ID:101
Employee Name: Alice
Designation: Manger
Basic Salary:80000
Experience:12 years

Employee ID:102
Employee Name: vicky
Designation: HR
Basic Salary:160000
Experience:20 years

PS C:\Users\SRAVANI\Documents\AI Assist>

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.

```
task2.py > ...
1 class ElectricityBill():
2     def __init__(self, customer_id, name, units_consumed):
3         self.customer_id = customer_id
4         self.name = name
5         self.units_consumed = units_consumed
6     def display_details(self):
7         print("Customer ID:", self.customer_id)
8         print("Customer Name:", self.name)
9         print("Units Consumed:", self.units_consumed)
10    def calculate_bill(self):
11        if self.units_consumed <= 100:
12            bill_amount = self.units_consumed * 5
13        elif 101 <= self.units_consumed <= 300:
14            bill_amount = (100 * 5) + (self.units_consumed - 100) * 7
15        else:
16            bill_amount = (100 * 5) + (200 * 7) + (self.units_consumed - 300) * 10
17        return bill_amount
18    bill=ElectricityBill(1, "Alice", 350)
19    bill.display (method) def calculate_bill() -> Any
20    amount=bill.calculate_bill()
21    print(f"Total Bill Amount: {amount}")
```

```
Customer ID: 1
Customer Name: Alice
Units Consumed: 350
Total Bill Amount: 2400
PS C:\Users\SRAVANI\Documents\AI Assist>
```

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.

```
task3.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     def display_details(self):
8         print(f"Product ID:{self.product_id}")
9         print(f"Product Name:{self.product_name}")
10        print(f"Price:{self.price}")
11        print(f"Category:{self.category}")
12    def calculate_discount(self):
13        if self.category.lower() == "electronics":
14            discount = 0.10 * self.price
15        elif self.category.lower() == "clothing":
16            discount = 0.15 * self.price
17        else:
18            discount = 0.05 * self.price
19        print(f"Discount Amount: {discount}")
20        print(f"Price after Discount: {self.price - discount}")
21    prod1=Product(301,"Laptop",60000,"Electronics")
22    prod1.display_details()
23    prod1.calculate_discount()
```

```
Product ID:301
Product Name:Laptop
Price:60000
Category:Electronics
Discount Amount: 6000.0
Price after Discount: 54000.0
PS C:\Users\SRAVANI\Documents\AI Assist>
```

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.

```
task-4.py > ...
1 class LibraryBook:
2     def __init__(self, book_id, title, author, borrower, days_late):
3         self.book_id = book_id
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.book_id}")
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 librarybook1 = LibraryBook("8003", "To Kill a Mockingbird", "Harper Lee", "Sarah Smith", 4)
23 librarybook1.display_details()
24 librarybook1.calculate_late_fee()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
task-4.py
Book ID: 8003
Title: To Kill a Mockingbird
Author: Harper Lee
Borrower: Sarah Smith
Days Late: 4
Late Fee: 20
PS C:\Users\SRAVANI\Documents\AI Assist>
```

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.

```
Assignment-5&6 > task-5.py > ...
1  def student_performance_report(students):
2      report = {}
3      for name, marks in students.items():
4          if marks >= 40:
5              report[name] = "Pass"
6          else:
7              report[name] = "Fail"
8  def calculate_average_marks(students):
9      averages = {}
10     for name, marks_list in students.items():
11         averages[name] = sum(marks_list) / len(marks_list)
12     return averages
13     students={
14         "Alice": [85, 90, 78],
15         "Bob": [35, 40, 50],
16         "Charlie": [60, 70, 80]
17     }
18     average_marks = calculate_average_marks(students)
19     print("Average_marks:")
20     for name, avg in average_marks.items():
21         print(f"{name}: {avg:.2f}")
22     performance_report = {
23         name: "Pass" if all(mark >= 40 for mark in marks) else "Fail"
24         for name, marks in students.items()
25     }
26     }
27     print("Student performance report:")
28     for name, status in performance_report.items():
29         print(f"{name}: {status}"]
```

```
Average_marks:
Alice: 84.33
Bob: 41.67
Charlie: 70.00
Student performance report:
Alice: Pass
Bob: Fail
Charlie: Pass
PS C:\Users\SRAVANI\Documents\AI Assist>
```

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.

```
Assignment-5&6 > task-6.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
8     def display_details(self):
9         print(f"Ride ID: {self.ride_id}")
10        print(f"Driver Name: {self.driver_name}")
11        print(f"Distance (km): {self.distance_km}")
12        print(f"Waiting Time (min): {self.waiting_time_min}")
13
14    def calculate_fare(self):
15        if self.distance_km <= 10:
16            fare = self.distance_km * 15
17        elif 11 <= self.distance_km <= 30:
18            fare = (10 * 15) + (self.distance_km - 10) * 12
19        else:
20            fare = (10 * 15) + (20 * 12) + (self.distance_km - 30) * 10
21
22        fare += self.waiting_time_min * 2
23        return fare
24
25
26    ride = TaxiRide(501, "Charlie Brown", 25, 10)
27    ride.display_details()
28    fare = ride.calculate_fare()
29    print(f"Total Fare: {fare}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Ride ID: 501
Driver Name: Charlie Brown
Distance (km): 25
Waiting Time (min): 10
Total Fare: 350
PS C:\Users\SRAVANI\Documents\AI Assist>
```

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

```
Assignment-586 > task-7.py > ...
1 def statistics_subject(score_list):
2     total = sum(score_list)
3     average = total / len(score_list)
4     highest = max(score_list)
5     lowest = min(score_list)
6     passed = 0
7     failed = 0
8     for i in score_list:
9         if i >= 40:
10             passed += 1
11         else:
12             failed += 1
13     print(f"Number of Students Passed: {passed}")
14     print(f"Number of Students Failed: {failed}")
15     return {
16         "average": average,
17         "highest": highest,
18         "lowest": lowest
19     }
20 scores = [
21     28, 49, 33, 72, 15, 60, 95, 40, 53, 81, 22, 47, 68, 79, 34, 91, 44, 58, 73, 38, 66, 84, 29, 50, 77, 92, 41,
22     36, 65, 80, 54, 87, 30, 69, 45, 71, 39, 83, 59, 74
23 ]
24 stats = statistics_subject(scores)
25 print(stats)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAVANI/docu
Assignment-586/Task-7.py"
Number of Students Passed: 30
Number of Students Failed: 10
{'average': 57.775, 'highest': 95, 'lowest': 15}
PS C:\Users\SRAVANI\Documents\AI Assist>
```

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.

```

1  # generate a program that reads a file and process the data
2  # Generate code with proper error handling and clear explanations for each exception.
3
4  def read_file(file_path):
5      try:
6          # Attempt to open the file
7          with open(file_path, 'r') as file:
8              data = file.read()
9              print("File content successfully read.")
10             return data
11     except FileNotFoundError:
12         # Handle the case where the file does not exist
13         print(f"Error: The file at {file_path} was not found.")
14     except PermissionError:
15         # Handle the case where there are permission issues
16         print(f"Error: You do not have permission to read the file at {file_path}.")
17     except Exception as e:
18         # Handle any other exceptions that may occur
19         print(f"An unexpected error occurred: {e}")
20 file_path = 'example.txt' # Specify the path to your file here
21 file_content = read_file(file_path)
22 if file_content:
23     print("File Content:")
24     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

```

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.

```

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: use of the two preceding numbers
27        return fibonacci(n - 1) + fibonacci(n - 2)
28
29 # Example usage and verification
30 n = 6
31 print(f"The {n}th Fibonacci number is: {fibonacci(n)}")
32 # Explanation:
33 # When we call fibonacci(6), the function checks if n is 0 or 1. Since it's neither, it proceeds to the recursive case:
34 # fibonacci(6) = fibonacci(5) + fibonacci(4)
35 # This pattern continues, breaking down each call until it reaches the base cases:
36 # fibonacci(1) = 1 and fibonacci(0) = 0

```

```

The 6th Fibonacci number is: 8
PS C:\Users\SRAVANI\Documents\AI Assist>

```

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.

Assignment-5&6 > task-10.py > ...

```
1  # generate two programs naive approach and optimized approach to check if given number is prime or not
2  # also calculate time and space complexities of both programs
3  import time
4
5  # Naive Approach
6  def is_prime_naive(n):
7      if n <= 1:
8          return False
9      for i in range(2, n):
10         return True
11
12
13
14
15 start_time = time.time()
16 number = 29
17 result_naive = is_prime_naive(number)
18 end_time = time.time()
19
20 print(f"Naive Approach: Is {number} prime? {result_naive}")
21 print(f"Time taken (Naive): {end_time - start_time} seconds")
22
23 # Time Complexity: O(n)
24 # Space Complexity: O(1)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/SRAVANI/Documents/Assignment-5&6/task-10.py"
Naive Approach: Is 29 prime? True
Time taken (Naive): 7.3989759521484375e-06 seconds
PS C:\Users\SRAVANI\Documents\AI Assist>
```