ASSIGNMENT-5.1 & 6

M.ADARSH REDDY

2303A51687

Task 1:

Employee Data: Create Python code that defines a class name Employee` with the following attributes: `empid`, `empname`,designation`, `basic_salary`, and `exp`. Implement a method display_details()` to print all employee details. Implement another`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.

CODE:

```
class Employee:
    def __init__(self,empid,empname,basic_salary,exp):
        self.empid = empid
        self.empname = empname
        self.basic_salary = basic_salary
        self.exp = exp
def display_details(emp):
    print(f"Employee ID: {emp.empid}")
    print(f"Employee Name: {emp.empname}")
    print(f"Basic Salary: {emp.basic_salary}")
    print(f"Experience: {emp.exp} years")
def calculate_Allowance(emp):
    if emp.exp > 10:
        allowance = emp.basic_salary * 0.20
    elif 5 <= emp.exp < 10:
        allowance = emp.basic_salary * 0.10
    else:
        allowance = emp.basic_salary * 0.05
    return allowance
emp1 = Employee(101, "John Doe", 50000, 12)
display_details(emp1)
print(f"Allowance: {calculate_Allowance(emp1)}")
```

OUTPUT:

```
PS C:\Users\adars> & C:/Users/adars/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/adars/assign -5.1.py"
Employee ID: 101
Employee Name: John Doe
Basic Salary: 50000
Experience: 12 years
Allowance: 10000.0
```

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 ≤ 100 → ₹5 per unit

- 101 to 300 units → ₹7 per unit

- More than 300 units → ₹10 per unit

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

CODE:

```python
##2
class ElictricityBill:
    def __init__(self,customer_id,customer_name,units_consumed):
        self.customer_id = customer_id
        self.customer_name = customer_name
        self.units_consumed = units_consumed

def display_details(bill):
    print(f"Customer ID: {bill.customer_id}")
    print(f"Customer Name: {bill.customer_name}")
    print(f"Units Consumed: {bill.units_consumed}")

def calculate_bill(self):
        if self.units_consumed <= 100:
            bill_amount = self.units_consumed * 5
        elif 101 <= self.units_consumed <= 300:
            bill_amount = self.units_consumed * 7
        else:
            bill_amount = self.units_consumed * 10
        return bill_amount
bill1 = ElictricityBill(201, "Alice Smith", 150)
display_details(bill1)
print(f"Total Bill Amount: ₹{calculate_bill(bill1)}")
```

OUTPUT:

```
PS C:\Users\adars> & C:/Users/adars/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/adars/assign -5.1.py"
Customer ID: 201
Customer Name: Alice Smith
Units Consumed: 150
Total Bill Amount: ₹1050
```

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.

CODE:

```python
##3
class product:
    def __init__(self,product_id,product_name,price,category):
        self.product_id = product_id
        self.product_name = product_name
        self.price = price
        self.category = category

def display_details(product):
    print(f"Product ID: {product.product_id}")
    print(f"Product Name: {product.product_name}")
    print(f"Price: ₹{product.price}")
    print(f"Category: {product.category}")

def calculate_discount(product):
    if product.category == "Electronics":
        discount = product.price * 0.10
    elif product.category == "Clothing":
        discount = product.price * 0.15
    else:
        discount = product.price * 0.05
    return discount
prod1 = product(1, "Smartphone", 17500, "Electronics")
display_details(prod1)
print(f"Discount: ₹{calculate_discount(prod1)}")
```

OUTPUT:

```
PS C:\Users\adars> & C:/Users/adars/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/adars/a
Product Name: Smartphone
Price: ₹17500
Category: Electronics
Discount: ₹1750.0
```

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 ≤ 5 → ₹5 per day

- 6 to 10 days late → ₹7 per day

- More than 10 days late → ₹10 per day

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

CODE:

```
class LibraryBook:
    def __init__(self,book_id,book_title,author,borrower,days_late):
        self.book_id = book_id
        self.book_title = book_title
        self.author = author
        self.borrower = borrower
        self.days_late = days_late
    def display_details(book):
        print(f"Book ID: {book.book_id}")
        print(f"Book Title: {book.book_title}")
        print(f"Author: {book.author}")
        print(f"Borrower: {book.borrower}")
        print(f"Days Late: {book.days_late}")

    def calculate_late_fee(self):
        if self.days_late <= 5:
            fee = self.days_late * 5
        elif 6 <= self.days_late <= 10:
            fee = self.days_late * 7
        else:
            fee = self.days_late * 10
        return fee
book1 = LibraryBook(11, "The Great Gatsby", "F. Scott Fitzgerald", "Bob Johnson", 8)
LibraryBook.display_details(book1)
print(f"Late Fee: ₹{book1.calculate_late_fee()}")
```

OUTPUT:

```
PS C:\Users\adars> & C:/Users/adars/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/adars/assign -5.1.py"
Book ID: 11
Book Title: The Great Gatsby
Author: F. Scott Fitzgerald
Borrower: Bob Johnson
Days Late: 8
Late Fee: ₹56
```

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.

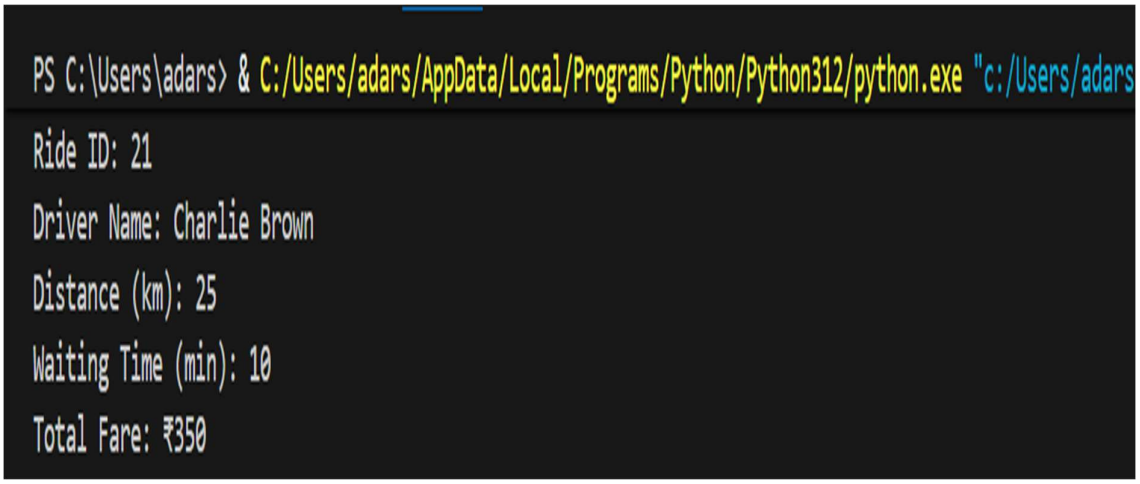
CODE:

```python
class TaxiRide:
    def __init__(self,ride_id,driver_name,distance_km,waiting_time_min):
        self.ride_id = ride_id
        self.driver_name = driver_name
        self.distance_km = distance_km
        self.waiting_time_min = waiting_time_min
    def display_details(ride):
        print(f"Ride ID: {ride.ride_id}")
        print(f"Driver Name: {ride.driver_name}")
        print(f"Distance (km): {ride.distance_km}")
        print(f"Waiting Time (min): {ride.waiting_time_min}")

    def calculate_fare(self):
        fare = 0
        if self.distance_km <= 10:
            fare = self.distance_km * 15
        elif self.distance_km <= 30:
            fare = (10 * 15) + (self.distance_km - 10) * 12
        else:
            fare = (10 * 15) + (20 * 12) + (self.distance_km - 30) * 10
        fare += self.waiting_time_min * 2

        return fare
ride1 = TaxiRide(21, "Charlie Brown", 25, 10)
TaxiRide.display_details(ride1)
print(f"Total Fare: ₹{ride1.calculate_fare()}")
```

OUTPUT:



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

CODE:

```python
##8 gnerate a prime number series using optimization by well commented and well documented code.
def is_prime(num):
    """
    This function checks if a number is prime.

    Parameters:
    num (int): The number to check for primality.

    Returns:
    bool: True if the number is prime, False otherwise.
    """
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
def prime_series(n):
    """
    This function generates a list of prime numbers up to n.
    Parameters:
    n (int): The upper limit for generating prime numbers.
    Returns:
    list: A list of prime numbers up to n.
    """
    primes = []
    for num in range(2, n + 1):
        if is_prime(num):
            primes.append(num)
    return primes
# Example usage:
n = 10  # Change this value to generate prime numbers up to a different limit
print(f"Prime numbers up to {n}: {prime_series(n)}")
```

OUTPUT:

```
PS C:\Users\adars> & C:/Users/adars/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/adars/assign -5.1.py"
Prime numbers up to 10: [2, 3, 5, 7]
```

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.

CODE:

```python
###9 Generate a fibonacci series usin ng recursion by well commented and well documented code.

def fibonacci(n):
    """
    This function returns the nth Fibonacci number using recursion.

    Parameters:
    n (int): The position in the Fibonacci series to retrieve.

    Returns:
    int: The nth Fibonacci number.
    """
    # Base case: the first two Fibonacci numbers are 0 and 1
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        # Recursive case: sum of the two preceding numbers
        return fibonacci(n - 1) + fibonacci(n - 2)
# Example usage:
n = 12  # Change this value to get a different Fibonacci number
print(f"The {n}th Fibonacci number is: {fibonacci(n)}")
```

OUTPUT:

```
PS C:\Users\adars> & C:/Users/adars/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/adars/assign -5.1.py"
The 12th Fibonacci number is: 144
```

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.

CODE:

```python
# generate  a python program to read a file and proccesses data using well commented and well documented code.
def read_and_process_file(file_path):
    """
    This function reads a file and processes its data.

    Parameters:
    file_path (str): The path to the file to be read.

    Returns:
    None
    """
    try:
        # Open the file in read mode
        with open(file_path, 'r') as file:
            # Read the contents of the file
            data = file.readlines()

            # Process each line in the file
            for line in data:
                # Example processing: print each line after stripping whitespace
                print(line.strip())

    except FileNotFoundError:
        print(f"The file at {file_path} was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")
# Example usage:
file_path = 'example.txt'  # Change this to the path of your file
read_and_process_file(file_path)
```

OUTPUT:

```
PS C:\Users\adars> & C:/Users/adars/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/adars/assign -5.1.py"
The file at example.txt was not found.
```