Employee-Wage-Computation

UC1: Added Employee Attendance Check

print("Welcome to Employee Wage Computation Program")

import random

print("Welcome to Employee Wage Computation Program")

UC1: Check Employee Attendance

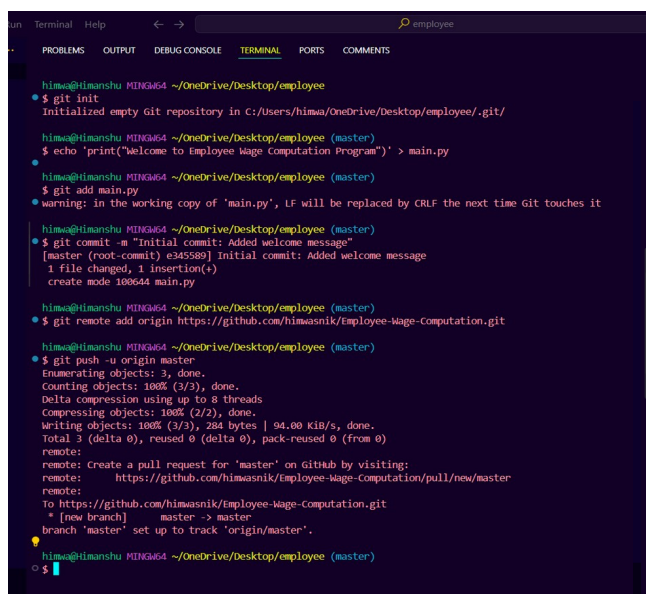attendance = random.choice([0, 1])

if attendance == 1:

    print("Employee is Present")

else:

    print("Employee is Absent")

Explanation:-

- Class **Variables**: Constants such as WAGE_PER_HOUR, FULL_DAY_HOURS, PART_TIME_HOURS, etc., are class variables used throughout the calculations.

- Attendance **Check (UC 1)**: The is_employee_present() method uses random.choice() to randomly assign whether an employee is present or absent.
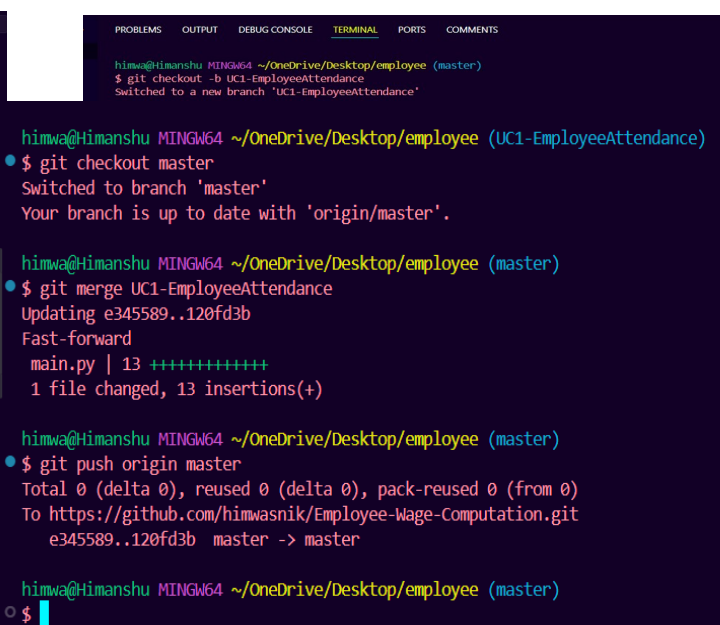
UC2: Added Daily Employee Wage Calculation
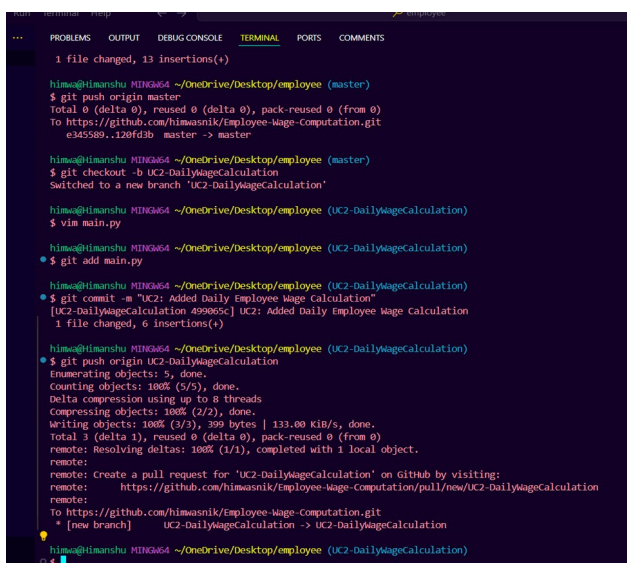
UC2: Daily Employee Wage

wage_per_hour = 20

full_day_hour = 8

daily_wage = wage_per_hour * full_day_hour

print(f"Daily Employee Wage: {daily_wage}")

Explanation:-

Daily Wage Calculation (UC 2): calculate_daily_wage() takes the number of hours worked as input and calculates the daily wage based on WAGE_PER_HOUR

UC3: Part-Time Employee Wage

part_time_hour = 4

part_time_wage = wage_per_hour * part_time_hour

print(f"Part-time Employee Wage: {part_time_wage}")

Explanation:-

Part-Time Wage (UC 3): part_time_employee_wage() method calculates the wage based on part-time hours.

UC4: Solve using Switch Case

```
def employee_wage(employee_type):

    switcher = {

        1: full_day_hour * wage_per_hour,  # Full-time employee

        2: part_time_hour * wage_per_hour  # Part-time employee

    }

    return switcher.get(employee_type, 0)  # Default for absent
```

employee_type = random.choice([0, 1, 2]) # 0: Absent, 1: Full-time, 2: Part-time

print(f"Employee Wage (Switch Case): {employee_wage(employee_type)}")

Explanation:-

Monthly Wage Calculation (UC 4): monthly_wage() calculates the wage for 20 working days in a month.

UC5: Calculate Wages for a Month

working_days_per_month = 20

monthly_wage = working_days_per_month * daily_wage

print(f"Monthly Employee Wage: {monthly_wage}")

Explanation:-

Conditional Monthly Wage Calculation (UC 5): conditional_monthly_wage() keeps track of both total days and hours worked, stopping when either condition exceeds the monthly limits.

```python
max_hours_per_month = 100

total_working_days = 0

total_working_hours = 0


while total_working_days < 20 and total_working_hours < max_hours_per_month:

    hours_worked = random.choice([0, 4, 8])  # 0: Absent, 4: Part-time, 8: Full-time

    total_working_hours += hours_worked

    total_working_days += 1


total_wage = total_working_hours * wage_per_hour

print(f"Total Employee Wage for the Month: {total_wage}")
```

Explanation:-

Main Method (UC 6): compute_employee_wage() is the main entry point, calling conditional_monthly_wage() and printing the total monthly wage based on the conditions.

UC7: Refactor code using class method

```python
class EmployeeWageComputation:

    wage_per_hour = 20

    full_day_hour = 8

    part_time_hour = 4


    @classmethod
    def calculate_wage(cls, employee_type):
        if employee_type == 1:  # Full-time
            return cls.full_day_hour * cls.wage_per_hour
        elif employee_type == 2:  # Part-time
            return cls.part_time_hour * cls.wage_per_hour
        else:
            return 0  # Absent


employee_type = random.choice([0, 1, 2])  # 0: Absent, 1: Full-time, 2: Part-time
```
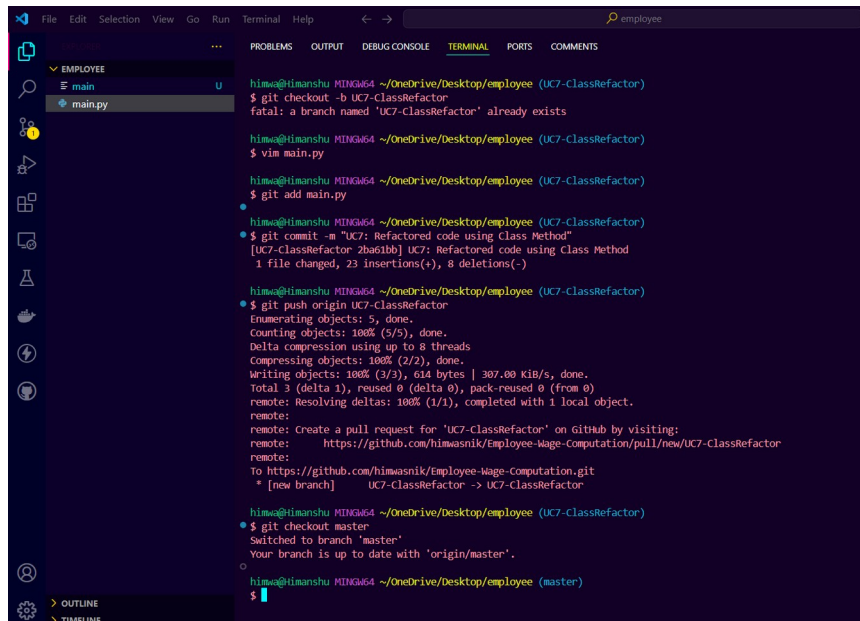
wage = EmployeeWageComputation.calculate_wage(employee_type)

print(f"Employee Wage (Class Method): {wage}")