Ex.No.: 7 13.03.2025	Class and Object	Register Number: URK24CS1189

7 A. Design a class to represent a rectangle with length and breadth as instance attributes. Create two rectangle objects, r1 and r2. Initialize the attributes using the constructor and do the following operations.

Aim: -

To write a Python program that reads the entire content of a file, reverses the content, and stores it back in another file.

Algorithm: -

Step 1: Start.
Step 2: Define Rectangle Class
Step 3: Create Rectangle Objects
- r1 with length=10, breadth=5
- r2 with length=20, breadth=6

Step 4: Add Rectangles
- Create r3 with:
  - length = r1.length + r2.length
  - breadth = r1.breadth + r2.breadth

Step 5: Display Result
- Print r3's dimensions

Step 6: Compare Rectangles
- Check equality (both dimensions equal)
- Check less than (both dimensions smaller)
- Check greater than (both dimensions larger)
- Check less than or equal
- Check greater than or equal

.**Step 7**: End.

Program: -

exp7.py > ...

```python
#URK24CS1189

class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth
r1 = Rectangle(10, 5)
r2 = Rectangle(20, 6)
r3 = Rectangle(r1.length + r2.length, r1.breadth + r2.breadth)
print(f"Length is {r3.length} and Breadth is {r3.breadth}")
print("r1 == r2:", r1.length == r2.length and r1.breadth == r2.breadth)
print("r1 < r2:", r1.length < r2.length and r1.breadth < r2.breadth)
print("r1 > r2:", r1.length > r2.length and r1.breadth > r2.breadth)
print("r1 <= r2:", r1.length <= r2.length and r1.breadth <= r2.breadth)
print("r1 >= r2:", r1.length >= r2.length and r1.breadth >= r2.breadth)
```

Output: -

```
Length is 30 and Breadth is 11
r1 == r2: False
r1 < r2: True
r1 > r2: False
r1 <= r2: True
r1 >= r2: False
```

7 B. Write a menu driven application to maintain the employee payroll details using Python. Your application must contain the following functionalities.
Use constructors, getter and setter functions.

Aim: -

To create a simple payroll system that stores employee details, calculates net salary and allows adding/viewing employees via a menu.

Algorithm: -

Step 1: Start.
Step 2: Open Input File (og.txt) for reading
Step 3: Create Employee class with:
- Constructor (name, ID, basic pay)
- net_salary() method (BP + DA + HRA - EPF - Tax)

Step 4: Create empty employees dictionary
Step 3: Add Employee
- Take input for name, ID
- basic pay
- Create new Employee object Store in dictionary using ID as key
- Print confirmation

Step 5: Show Employee
- Take input for employee ID. Look up in dictionary .
- If found: display name and formatted net salary If not found: display error

Step 6: Main Loop
- Display menu options:
  1. Add employee
  2. Show employee
  3. Exit
- Get user choice
- Call corresponding function
- Repeat until exit chosen

Step 7: end

Program: -

exp7.py > ...

```python
#URK24CS1189

class Employee:
    def __init__(self, name, empid, bp):
        self.name = name
        self.empid = empid
        self.bp = bp
    def net_salary(self):
        return self.bp * 1.10 * 1.05 - (self.bp * 0.15)
employees = {}
def add_emp():
    e = Employee(input("Name: "), input("ID: "), float(input("Basic Pay: ")))
    employees[e.empid] = e
    print("Added!")
def show_emp():
    e = employees.get(input("ID: "))
    if e:
        print(f"{e.name}: {e.net_salary():.2f}")
    else:
        print("Not found")
while True:
    print("\n1. Add\n2. Show\n3. Exit")
    c = input("Choice: ")
    if c == '1':
        add_emp()
    elif c == '2':
        show_emp()
    elif c == '3':
        break
```

Output: -

```
1. Add
2. Show
3. Exit
Choice: 1
Name: Leo
ID: 101
Basic Pay: 2
Added!

1. Add
2. Show
3. Exit
Choice: 2
ID: 100
Not found

1. Add
2. Show
3. Exit
Choice: 4

1. Add
2. Show
3. Exit
Choice: 3
```

Result: -

Thus the all programs has been done successfully.