|  |  |  |
| --- | --- | --- |
| **Ex.No.8** | **Classes and Objects in Python** | **Reg.No: URK23CS1261** |
| **20.3.24** |
| **8 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.**   * **r3 = r1 + r2, where r3 is an another Rectangle object** * **r3’s length = r1’s length + r2’s length** * **r3’s breadth = r1’s breadth + r2’s breadth**   **Obtain a user-friendly string representation of the Rectangle object as Length is 30 and Breadth is 11 by overriding \_\_str\_\_( ).**   * **print(r3)** * **Compare the dimensions of r1 and r2 => r1 == r2, r1 < r2, r1 > r2, r1 >= r2, r1 <= r2**   **Aim:** The objective of this program is to design a class to represent a rectangle with length and breadth as instance attributes.  **Algorithm:**  Step 1: Start the program.  Step 2: Define a class named Rectangle with appropriate methods for initialization, string representation, equality comparison, addition, and comparison operators.  Step 3: Create an instance of Rectangle called r1 with length 20 and breadth 6.  Step 4: Create an instance of Rectangle called r2 with length 10 and breadth 5.  Step 5: Add r1 and r2 and store the result in r3.  Step 6: Print the string representation of r3.  Step 7: Compare if r1 is equal to r2 and print whether r1 is equal to r2.  Step 8: Compare if r1 is greater than r2 and print whether r1 is greater than r2.  Step 9: Compare if r1 is less than r2 and print whether r1 is less than r2.  Step 10: End the program.  **Program:**  class Rectangle():  def \_\_init\_\_(self: object, length: int, breadth: int) -> None:  self.length = length  self.breadth = breadth  def \_\_str\_\_(self: object) -> str:  return f"Length: {self.length} Breadth: {self.breadth}"  def \_\_eq\_\_(self, other: object) -> bool:  return self.length == other.length, self.breadth == other.breadth  def \_\_add\_\_(self:object, other: object) -> int:  return Rectangle(self.length + other.length, self.breadth + other.breadth)  def \_\_lt\_\_(self, other: object) -> bool:  return self.length < other.length, self.breadth < other.breadth  def \_\_gt\_\_(self, other: object) -> bool:  return self.length > other.length, self.breadth > other.breadth  def \_\_ge\_\_(self, other: object) -> bool:  return self.length >= other.length, self.breadth >= other.breadth  def \_\_le\_\_(self, other: object) -> bool:  return self.length <= other.length, self.breadth <= other.breadth  r1 = Rectangle(length=20, breadth=6)  r2 = Rectangle(length=10, breadth=5)  r3 = r1 + r2  print(r3)  r3 = r1 == r2  print(f"Equal: {r3}")  r3 = r1 > r2  print(f"Greater than: {r3}")  r3 = r1 < r2  print(f"Less than: {r3}")  r3 = r1 >= r2  print(f"Greater than or equal: {r3}")  r3 = r1 <= r2  print(f"Less than or equal: {r3}")  print("╔═════════════╗\n║   Tanvik    ║\n║ URK23CS1261 ║\n╚═════════════╝")  **Output:**        **Result:** Thus, The program has successfully produced the desired output. | | |
| **8 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.**  **a. For each employee your application must have the details such as name, empid, department, designation, experience, basicPay, DA(10% BP), HRA(5%BP),EPF(5%BP), Tax(10% of BP), Net salary= BP+DA+HRA-EPF-Tax**  **b. Get the employee details from user(admin)**  **c. In the menu give the user options to add, edit, delete or display the employee details.**  **Aim:** The objective of this program is to build a menu driven application to maintain the employee payroll details.  **Algorithm:**  Step 1: Start the program.  Step 2: Define a class named Employee with attributes for employee details and methods for calculating salary and displaying employee information.  Step 3: Inside the class, define an \_\_init\_\_ method to initialize the attributes to None.  Step 4: Define a method named other\_data to calculate additional salary components based on the basic pay.  Step 5: Define a method named set\_data to set the employee details by taking user input.  Step 6: Define a method named display to print the employee information along with calculated salary components.  Step 7: Create an empty list named Emp to store instances of Employee class.  Step 8: Display a menu with options to add, edit, delete, display employees, or exit the program.  Step 9: Based on the user's choice, perform the corresponding action such as adding, editing, deleting, displaying employee information, or exiting the program.  Step 10: End the program if the user chooses to exit.  **Program:**  class Employee:  def \_\_init\_\_(self):  self.name = None  self.empid = None  self.department = None  self.designation = None  self.experience = None  self.basicPay = None  self.da = None  self.hra = None  self.epf = None  self.Tax = None  self.NetSalary = None  def other\_data(self, basicPay):  try:  self.da = 10 / 100 \* basicPay  self.hra = 5 / 100 \* basicPay  self.epf = 5 / 100 \* basicPay  self.Tax = 10 / 100 \* basicPay  self.NetSalary = basicPay + self.da + self.hra - self.epf - self.Tax  except Exception as e:  print(f"Error in Em \_\_dataset : {e}")  def set\_data(self):  self.name = input("name: ")  self.empid = int(input("empid: "))  self.department = input("department: ")  self.designation = input("designation: ")  self.experience = float(input("experience: "))  self.basicPay = float(input("basicPay: "))  def display(self):  self.other\_data(self.basicPay)  print(f"""  Employee Name: {self.name}  Employee ID: {self.empid}  Department: {self.department}  Designation: {self.designation}  Experience: {self.experience}  Basic pay: {self.basicPay}  DA: {self.da}  HRA: {self.hra}  EPF: {self.epf}  Tax: {self.Tax}  Net Salary: {self.NetSalary}  """)  Emp = []  while True:  print("""Menu:  1. Add  2. Edit  3. Delete  4. Display  5. Exit""")  option = int(input("Enter your choice: "))  if option == 1:  print("Enter the new employee details: ")  emp = Employee()  emp.set\_data()  Emp.append(emp)  print("Employee successfully added!")  elif option == 2:  empid = int(input("Enter the employee empid: "))  found = False  for emp in Emp:  if emp.empid == empid:  emp.set\_data()  found = True  print("Employee successfully Edited!")  break  if not found:  print("ID not found")  elif option == 3:  empid = int(input("Enter the employee empid: "))  for emp in Emp:  if emp.empid == empid:  Emp.remove(emp)  print("Employee successfully Deleted!")  break  else:  print("ID not found")  elif option == 4:  empid = int(input("Enter the employee empid: "))  for emp in Emp:  if emp.empid == empid:  emp.display()  break  else:  print("ID not found")  elif option == 5:  quit()  print("╔═════════════╗\n║ Tanvik ║\n║ URK23CS1261 ║\n╚═════════════╝")  **Output:**    **Result:** Thus, The program has successfully produced the desired output. | | | |