**Python Tasks:**

1. Create an employee object with attributes of employee id, department name, and salary. Create a list with 10 employee objects. Get employee count department wise

**Code Snippet:**

""" 1. Create an employee object with attributes

of employee id, department name, and salary.

Create a list with 10 employee objects.

Get employee count department wise"""

class Employee():

    totalEmpCount = hr\_count = manager\_count = TL\_count = anl\_count = progr\_count = 0

    def \_\_init\_\_(self, eId, eDept, eSal):

        self.eId = eId

        self.eDept = eDept

        self.eSal = eSal

        Employee.totalEmpCount += 1

        if self.eDept.lower() == "hr":

            Employee.hr\_count += 1

        elif self.eDept.lower() == "manager":

            Employee.manager\_count += 1

        elif self.eDept.lower() == "tl":

            Employee.TL\_count += 1

        elif self.eDept.lower() == "analyst":

            Employee.anl\_count += 1

        elif self.eDept.lower() == "programmer":

            Employee.progr\_count += 1

    def display(self):

        print(f"Employee Id: {self.eId}, Department: {self.eDept}, Salary: {self.eSal}")

    def empCount(self):

        return(Employee.hr\_count, Employee.manager\_count, Employee.TL\_count, Employee.anl\_count, Employee.progr\_count)

empList = []

empList.append(Employee(101, "HR", 50000))

empList.append(Employee(102, "Analyst", 25000))

empList.append(Employee(103, "TL", 40000))

empList.append(Employee(104, "Manager", 70000))

empList.append(Employee(105, "Programmer", 30000))

empList.append(Employee(106, "TL", 45000))

empList.append(Employee(107, "Manager", 60000))

empList.append(Employee(108, "Programmer", 20000))

empList.append(Employee(109, "Programmer", 30000))

empList.append(Employee(110, "Analyst", 30000))

for x in range(len(empList)):

    empList[x].display()

totEmp = empList[len(empList) - 1].totalEmpCount

hr, mgr, tl, anl, prg = empList[len(empList) - 1].empCount()

print(f"Total no.of Employees: {totEmp}")

print(f"Total no.of HR:{hr}")

print(f"Total no.of Manager:{mgr}")

print(f"Total no.of TL:{tl}")

print(f"Total no.of Analyst:{anl}")

print(f"Total no.of Programmer:{prg}")

**Output:**

**Graphical user interface, text

Description automatically generated**

2. Define an employee object. Create 2 employee objects - one is a Teacher employee object and another one is a Clerk employee object. Define appropriate functionalities for the teacher & clerk and execute them.

**Code Snippet:**

""" 2. Define an employee object. Create 2 employee objects -

one is a Teacher employee object and another one is a Clerk employee object.

Define appropriate functionalities for the teacher & clerk and execute them. """

class Employee():

    def \_\_init\_\_(self, eName, eExp):

        self.eName = eName

        self.eExp = eExp

        self.salary = 0

class Teacher(Employee):

    def \_\_init\_\_(self, eName, eExp, basicPay):

        super().\_\_init\_\_(eName, eExp)

        self.basicPay = basicPay

        self.eExp = int(self.eExp)

    def calc\_salary(self):

        if self.eExp >= 7:

            self.salary = int(self.basicPay) + 10000

        elif self.eExp >= 5:

            self.salary = int(self.basicPay) + 7000

        elif self.eExp >= 3:

            self.salary = int(self.basicPay) + 5000

        elif self.eExp >= 1:

            self.salary = int(self.basicPay) + 3000

        elif self.eExp <= 1:

            self.salary = int(self.basicPay) + 1000

        return self.salary

class Clerk(Employee):

    def \_\_init\_\_(self, eName, eExp, basicPay):

        super().\_\_init\_\_(eName, eExp)

        self.basicPay = basicPay

        self.eExp = int(self.eExp)

    def calc\_salary(self):

        if self.eExp >= 7:

            self.salary = int(self.basicPay) + 7000

        elif self.eExp >= 5:

            self.salary = int(self.basicPay) + 5000

        elif self.eExp >= 3:

            self.salary = int(self.basicPay) + 3000

        elif self.eExp >= 1:

            self.salary = int(self.basicPay) + 1000

        elif self.eExp <= 1:

            self.salary = int(self.basicPay) + 0

        return self.salary

print("Choose the employee type from below: \n"

      "1 => Teacher \n"

      "2 => Clerk")

choice = input("Enter the choice: ")

try:

    int(choice)

except (TypeError, ValueError):

    print("Please enter a valid option!")

else:

    match int(choice):

        case 1:

            ename = input("Enter the Teacher's name: ")

            exp = input("Enter the no. of experience: ")

            try:

                int(exp)

            except (ValueError, TypeError):

                print("Please enter the valid no. of experience!")

            else:

                basicpay = input("Enter the basic pay of the Teacher: ")

                try:

                    int(basicpay)

                except (ValueError, TypeError):

                    print("Please enter the valid Basic Pay!")

                else:

                    salary\_Teacher = Teacher(ename,exp,basicpay).calc\_salary()

                    print(f"The salary of employee ({ename}) with {exp} years experience is Rs.{salary\_Teacher}")

        case 2:

            ename = input("Enter the Clerk's name: ")

            exp = input("Enter the no. of experience: ")

            try:

                int(exp)

            except (ValueError, TypeError):

                print("Please enter the valid no. of experience!")

            else:

                basicpay = input("Enter the basic pay of the Clerk: ")

                try:

                    int(basicpay)

                except (ValueError, TypeError):

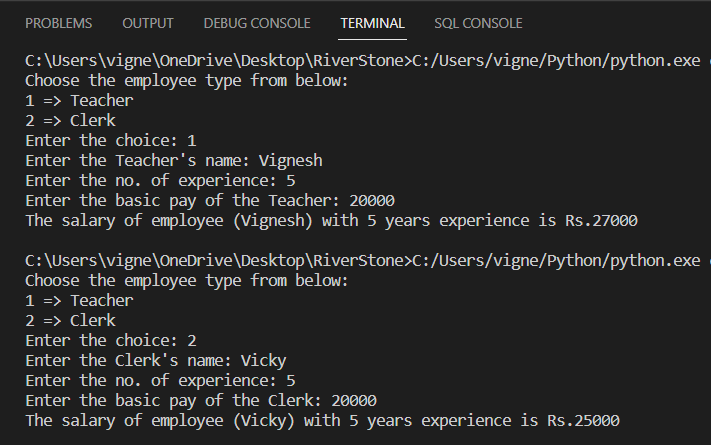
                    print("Please enter the valid Basic Pay!")

                else:

                    salary\_Clerk = Clerk(ename,exp,basicpay).calc\_salary()

                    print(f"The salary of employee ({ename}) with {exp} years experience is Rs.{salary\_Clerk}")

**Output:**

****