**Git Hub Link:** [**https://github.com/Upender12/https-github.com-Upender12-Assignment3\_Neural**](https://github.com/Upender12/https-github.com-Upender12-Assignment3_Neural)

**Spring 2024: CS5720**

**Neural Networks & Deep Learning - ICP-3**

**Name: Upender Reddy Bokka**

**#700: 700746118**

1. **Create a class Employee and then do the following**

**• Create a data member to count the number of Employees**

**• Create a constructor to initialize name, family, salary, department**

**• Create a function to average salary**

**• Create a Fulltime Employee class and it should inherit the properties of Employee class**

**• Create the instances of Fulltime Employee class and Employee class and call their member functions**

class Employee:

    """    class with Employee name, family, salary and department    """

    no\_of\_employees = 0

    def \_\_init\_\_(self, name, family\_name, salary, department):

        self.\_\_name = name

        self.\_\_family\_name = family\_name

        self.salary = salary

        self.\_\_department = department

        Employee.no\_of\_employees += 1

    @staticmethod

    def average\_salary(employees):

        """        function for the avg salary        """

        sum = 0

        for employee in employees:

            sum += employee.salary

        return sum / Employee.no\_of\_employees

class FulltimeEmployee(Employee):

    """    Full Time Employee is the sub class of the Employee    """

    def \_\_init\_\_(self, name, family\_name, salary, department):

        super().\_\_init\_\_(name, family\_name, salary, department)

    def full\_time\_benefits(self):

        print("Few benefits as full time employee.")

def main():

    employees = []

    fte1 = FulltimeEmployee("Employee1", "FamilyName1", 130000, "Management")

    fte1.full\_time\_benefits()

    employees.append(fte1)

    fte2 = FulltimeEmployee("Employee2", "FamilyName2", 190000, "RnD")

    employees.append(fte2)

    emp1 = Employee("Employee3", "FamilyName3", 170000, "Marketing")

    employees.append(emp1)

    emp2 = Employee("Employee4", "FamilyName4", 145000, "HR")

    employees.append(emp2)

    print("Average salary:", FulltimeEmployee.average\_salary(employees))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**OUTPUT:**

****

1. Numpy

Using NumPy create random vector of size 20 having only float in the range 1-20.

Then reshape the array to 4 by 5.

Then replace the max in each row by 0 (axis=1).

import numpy as np

def replace\_maxmium(array, replace\_value, axis):

    """ choose from x or y depending on condition: np.where(condition, x, y) """

    output = np.where(array == np.max(

        array, axis=1).reshape(-1, 1), 0 \* array, array)

    print(output)

def main():

    random20 = np.random.random\_sample(20) \* 20 + 1

    random20\_4by5 = random20.reshape((4, 5))

    replace\_maxmium(random20\_4by5, 0, 1)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**OUTPUT:**

