ICP – 3

Naga Phaneendra Kumara Gupta Mogili

700757977

GitHub link: https://github.com/nagaphaneendra2001/Deep_Learning_Neural_Networks.git

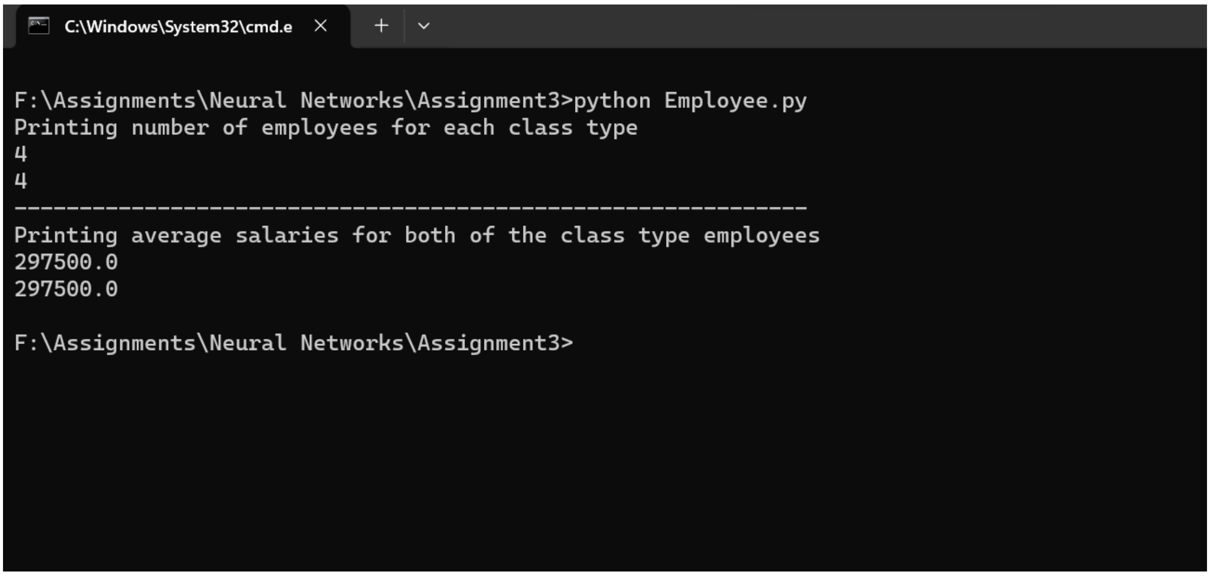
1. Create a class Employee and then do the following
   - Create a data member to count the number of Employees
   - Create a constructor to initialize name, family, salary, department
   - Create a function to average salary
   - Create a Fulltime Employee class and it should inherit the properties of Employee class
   - Create the instances of Fulltime Employee class and Employee class and call their member functions.

Source Code:

```
#Creating an Employee Class
class Employee:
  employee_count = 0

  #Constructor of Employee class which requires four parameters
  def __init__(self, name, family, salary, department):
    self.name = name
    self.family = family
    self.salary = salary
    self.department = department
```

```
        #Counting the number of employees
        Employee.employee_count = Employee.employee_count + 1

    #Method to find the average of the salaries of the employees
    def average_Salary(self,employees_list):
        sum_of_salaries = 0
        for employee in employees_list:
            sum_of_salaries += employee.salary
        print(sum_of_salaries / len(employees_list))

#Creating a class which inherits Employee class
class FullTimeEmployee(Employee):

    def __init__(self, name, family, salary, department):
        Employee.__init__(self, name, family, salary, department)

employees_list = []
full_time_employees_list = []

#Creating objects for Employee Class
employee1 = Employee("Naga Phaneendra", "Mogili", 295000, "Developer")
employee2 = Employee("Prasad", "Gaddam", 300000, "Manager")

#Creating objects for the FullTimeEmployee Class
employee3 = FullTimeEmployee("Stephen", "Kennedy", 295000, "HR")
employee4 = FullTimeEmployee("Uma", "Pichipati", 300000, "Tester")

employees_list.append(employee1)
employees_list.append(employee2)

full_time_employees_list.append(employee3)

full_time_employees_list.append(employee4)


#All the printing statements of the requirements

print("Printing number of employees for each class type")

print(Employee.employee_count)

print(FullTimeEmployee.employee_count)

print("-------------------------------------------------------------")


print("Printing average salaries for both of the class type employees")

employees_list[0].average_Salary(employees_list)

full_time_employees_list[0].average_Salary(full_time_employees_list)
```

Output:



```
F:\Assignments\Neural Networks\Assignment3>python Employee.py
Printing number of employees for each class type
4
4
-------------------------------------------------------------
Printing average salaries for both of the class type employees
297500.0
297500.0

F:\Assignments\Neural Networks\Assignment3>
```

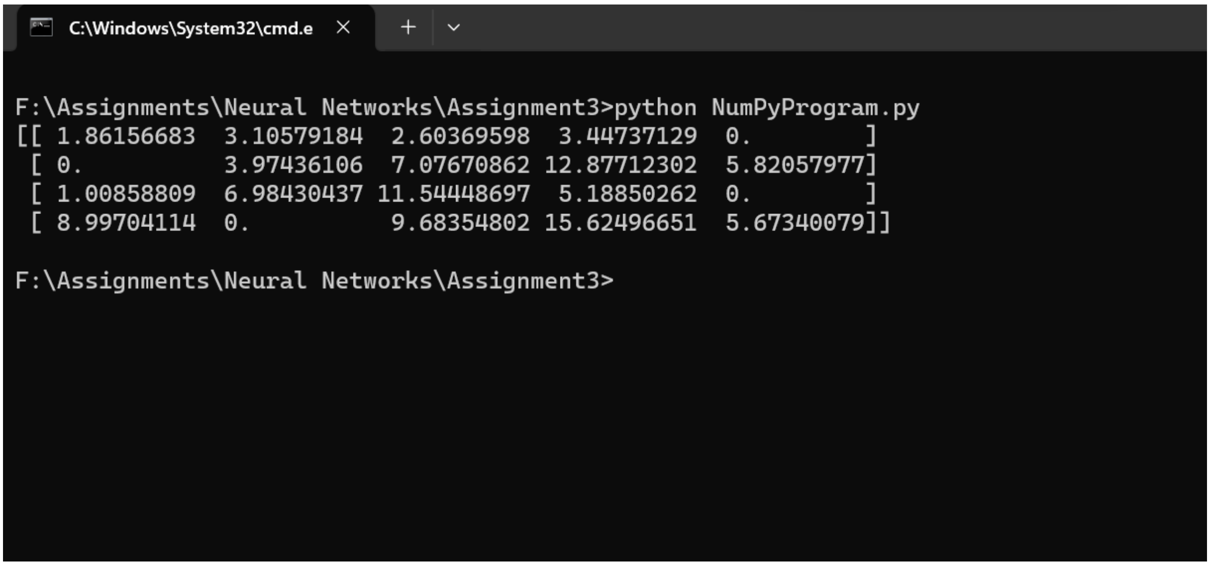
2. Numpy

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) (you can NOT implement it via for loop)

Source Code:

```
import numpy as np
from numpy import random

#Creating a random vector of size 20 with float values within the range of 1 to 20
random_vector = random.uniform(1,20,size=(20))

#Reshaping the random_vector using reshape() method
new_random_vector = random_vector.reshape(4, 5)

#Assigning '0' value to the maximum elements along the axis-1
new_random_vector[np.arange(4), new_random_vector.argmax(axis=1)] = 0.00

#Printing the final output
print(new_random_vector)
```

Output:



```
F:\Assignments\Neural Networks\Assignment3>python NumPyProgram.py
[[ 1.86156683  3.10579184  2.60369598  3.44737129  0.        ]
 [ 0.          3.97436106  7.07670862 12.87712302  5.82057977]
 [ 1.00858809  6.98430437 11.54448697  5.18850262  0.        ]
 [ 8.99704114  0.          9.68354802 15.62496651  5.67340079]]

F:\Assignments\Neural Networks\Assignment3>
```