Spring 2024: CS5720 Neural Networks & Deep Learning – ICP3

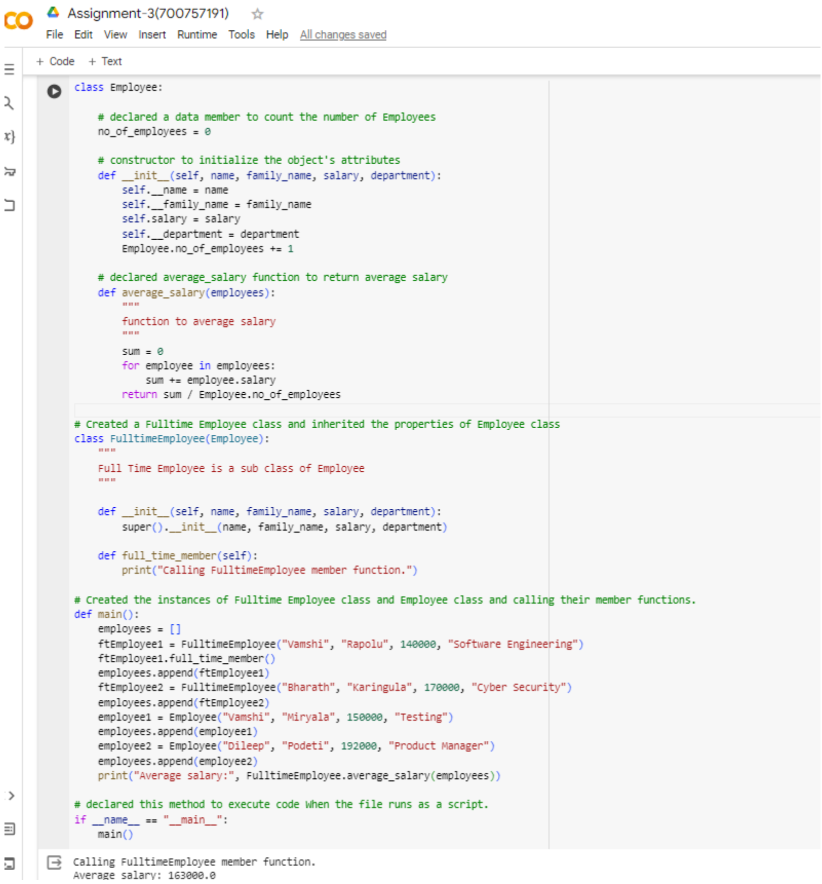
Assignment-3
NAME: Sangeetha Baddam
STUDENT ID:700757191

Video Link:
https://drive.google.com/file/d/1XX5VwWpgvLcSca6CRqLYnEJigEF96eAH/view?usp=sharing

GitHub Link: https://github.com/Sangeetha-Baddam/Assignment_3

1. Create a class Employee and then do the following

- Create a data member to count the number of Employee
- 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.

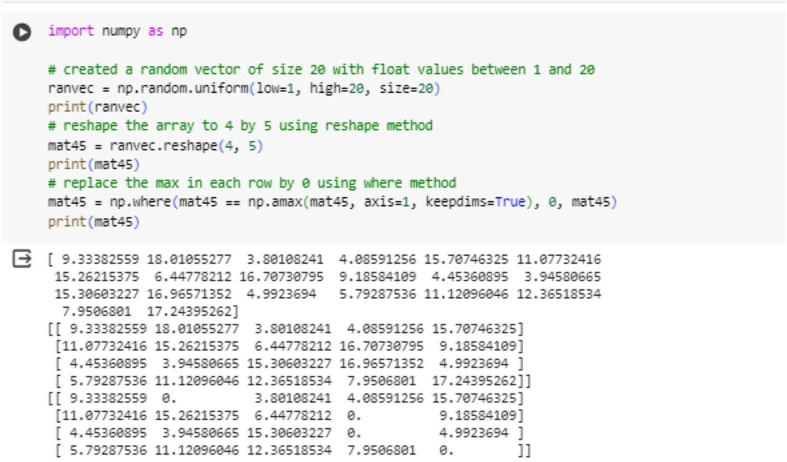


```
class Employee:

    # declared a data member to count the number of Employees
    no_of_employees = 0

    # constructor to initialize the object's attributes
    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

    # declared average_salary function to return average salary
    def average_salary(employees):
        """
        function to average salary
        """
        sum = 0
        for employee in employees:
            sum += employee.salary
        return sum / Employee.no_of_employees

# Created a Fulltime Employee class and inherited the properties of Employee class
class FulltimeEmployee(Employee):
    """
    Full Time Employee is a sub class of Employee
    """

    def __init__(self, name, family_name, salary, department):
        super().__init__(name, family_name, salary, department)

    def full_time_member(self):
        print("Calling FulltimeEmployee member function.")

# Created the instances of Fulltime Employee class and Employee class and calling their member functions.
def main():
    employees = []
    ftEmployee1 = FulltimeEmployee("Vamshi", "Rapolu", 140000, "Software Engineering")
    ftEmployee1.full_time_member()
    employees.append(ftEmployee1)
    ftEmployee2 = FulltimeEmployee("Bharath", "Karingula", 170000, "Cyber Security")
    employees.append(ftEmployee2)
    employee1 = Employee("Vamshi", "Miryala", 150000, "Testing")
    employees.append(employee1)
    employee2 = Employee("Dileep", "Podeti", 192000, "Product Manager")
    employees.append(employee2)
    print("Average salary:", FulltimeEmployee.average_salary(employees))

# declared this method to execute code When the file runs as a script.
if __name__ == "__main__":
    main()
```

```
Calling FulltimeEmployee member function.
Average salary: 163000.0
```

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)



```
import numpy as np

# created a random vector of size 20 with float values between 1 and 20
ranvec = np.random.uniform(low=1, high=20, size=20)
print(ranvec)
# reshape the array to 4 by 5 using reshape method
mat45 = ranvec.reshape(4, 5)
print(mat45)
# replace the max in each row by 0 using where method
mat45 = np.where(mat45 == np.amax(mat45, axis=1, keepdims=True), 0, mat45)
print(mat45)
```

```
[ 9.33382559 18.01055277  3.80108241  4.08591256 15.70746325 11.07732416
 15.26215375  6.44778212 16.70730795  9.18584109  4.45360895  3.94580665
 15.30603227 16.96571352  4.9923694   5.79287536 11.12096046 12.36518534
  7.9506801  17.24395262]
[[ 9.33382559 18.01055277  3.80108241  4.08591256 15.70746325]
 [11.07732416 15.26215375  6.44778212 16.70730795  9.18584109]
 [ 4.45360895  3.94580665 15.30603227 16.96571352  4.9923694 ]
 [ 5.79287536 11.12096046 12.36518534  7.9506801  17.24395262]]
[[ 9.33382559  0.          3.80108241  4.08591256 15.70746325]
 [11.07732416 15.26215375  6.44778212  0.          9.18584109]
 [ 4.45360895  3.94580665 15.30603227  0.          4.9923694 ]
 [ 5.79287536 11.12096046 12.36518534  7.9506801   0.        ]]
```