Git Hub Link: https://github.com/srija1609/NNDL3

Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-3

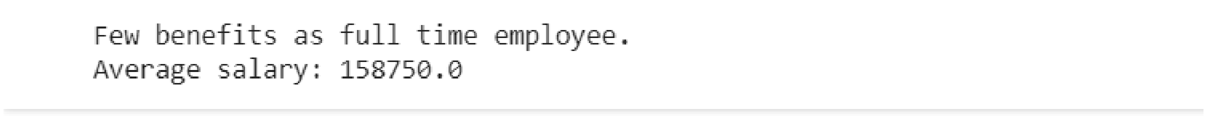
Name: Baby Srija Bitra

#700: 700755908

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:

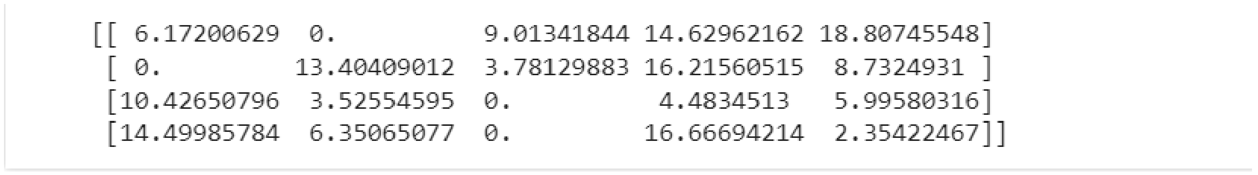


```
Few benefits as full time employee.
Average salary: 158750.0
```

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).

```
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:



```
[[ 6.17200629  0.          9.01341844 14.62962162 18.80745548]
 [ 0.         13.40409012  3.78129883 16.21560515  8.7324931 ]
 [10.42650796  3.52554595  0.          4.4834513   5.99580316]
 [14.49985784  6.35065077  0.         16.66694214  2.35422467]]
```