

Fall 2023: CS5720 Neural Networks & Deep Learning - ICP-3  
Assignment-3  
NAME:RAJYALAKSHMI GOTTIPATI  
STUDENT ID:700745186

Github Link:<https://github.com/rajigottipati/icp-3.git>

Video Link:

[https://drive.google.com/file/d/1vs83QomMrdi4jcFutWsK-wlxdGUhRy0w/view?usp=drive\\_link](https://drive.google.com/file/d/1vs83QomMrdi4jcFutWsK-wlxdGUhRy0w/view?usp=drive_link)

1. In class programming: 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.

```
#. 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:#Employee class
    empCount=0 ##employeeConut is used to keep track of numner of employee objects
    salSum=0 #salSum is used to take the average of the salaries of the Employee object
    def __init__(self,name,family,salary,department): #constructor for Employee as per the requirements
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.empCount += 1
        Employee.salSum+=self.salary
    def avg(self):
        return Employee.salSum/Employee.empCount
    def displayCount(self): #One of the functions in Employee class
        print ("Total Employee %d" % Employee.empCount)
    def displayEmployee(self):#one more function for employee class
        print("Name : ", self.name, "Family: ", self.family , "Salary: ", self.salary, "department:",self.department)

emp1 = Employee("MANIDEEP","BHEEMANATHI",30000,"DELL")#Employee objects creation with intialization
emp2 = Employee("MANASA","MANNEPALLI",60000,"HP")
emp3 = Employee("RAM","BANURI",45000,"TRUNOTE")
print("avg:",emp2.avg())#calling average function
class FulltimeEmployee(Employee):#FulltimeEmployee class with Employee as its parent class
```

```

class FulltimeEmployee(Employee):#FulltimeEmployee class with Employee as its parent class
    def __init__(self,name,family,salary,department,c_name):#constructor for FulltimeEmployee
        super().__init__(name,family,salary,department)
        self.c_name=c_name
    def setCompany(self,c_name):#one of the FulltimeEmployee class function
        self.c_name=c_name
    def getCompany(self):
        if self.c_name not in self.c_name:
            return ""
        return self.c_name
raj=FulltimeEmployee("RAJYALAKSHMI","GOTTIPATI",4000,"DEVELOPMENT","FACEBOOK")#FulltimeEmployee object
raj.setCompany("GUCCI")#calling the member functions of FulltimeEmployee and Employee
raj.getCompany(),raj.displayEmployee(),emp1.displayCount()

```

```

avg: 45000.0

```

```

Name : RAJYALAKSHMI Family: GOTTIPATI Salary: 4000 department: DEVELOPMENT
Total Employee 4
('GUCCI', None, None)

```

## 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)

```

#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 #importing numpy library
vector=np.arange(1,21,dtype=float) #creating a numpy vector with arange function
print(vector)
vector=vector.reshape(4,5)#reshaping the vector with reshape() function
print(vector)
vector=np.where(np.isin(vector,vector.max(axis=1)),0,vector)#finding the max values in row and replacing them with zero using np.where() and np.max()
vector

```

---

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
 19. 20.]
```

```
[[ 1.  2.  3.  4.  5.]
```

```
 [ 6.  7.  8.  9. 10.]
```

```
 [11. 12. 13. 14. 15.]
```

```
 [16. 17. 18. 19. 20.]]
```

```
array([[ 1.,  2.,  3.,  4.,  0.],
       [ 6.,  7.,  8.,  9.,  0.],
       [11., 12., 13., 14.,  0.],
       [16., 17., 18., 19.,  0.]])
```