NEURAL NETWORK & DEEP LEARNING

ASSIGNMENT 3

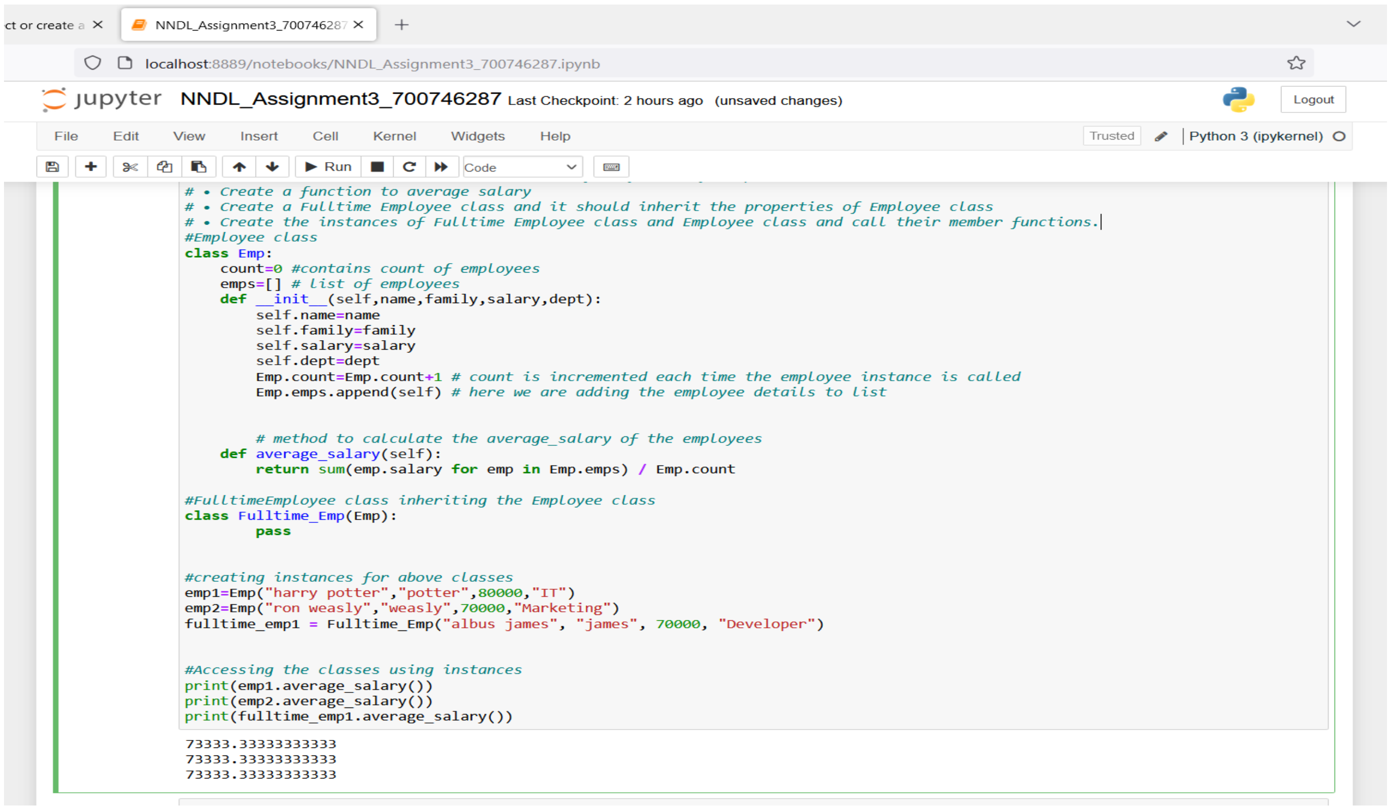
Name : SAI SNUSHA NAKKA

Student ID : 700746287

Git hub Link: https://github.com/NSnusha/NNDL_Assignment3

Video link: https://drive.google.com/file/d/1J_frq8hkfDmJ2uLBZRIf8ijHdh9Waey_/view?usp=sharing

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.



ct or create a | NNDL_Assignment3_700746287

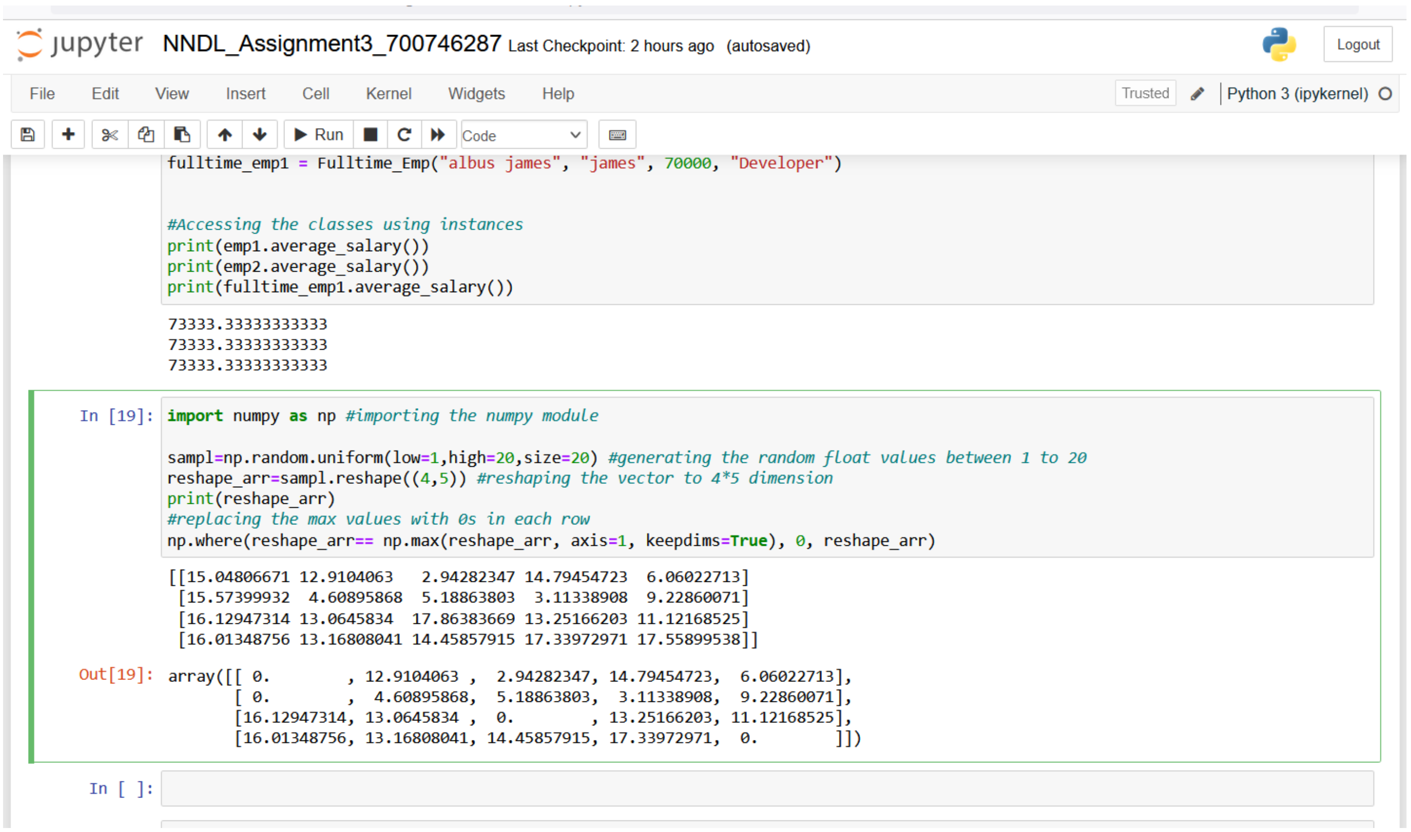
localhost:8889/notebooks/NNDL_Assignment3_700746287.ipynb

jupyter NNDL_Assignment3_700746287 Last Checkpoint: 2 hours ago (unsaved changes)

```python
# • 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.
#Employee class
class Emp:
    count=0 #contains count of employees
    emps=[] # list of employees
    def __init__(self,name,family,salary,dept):
        self.name=name
        self.family=family
        self.salary=salary
        self.dept=dept
        Emp.count=Emp.count+1 # count is incremented each time the employee instance is called
        Emp.emps.append(self) # here we are adding the employee details to list


        # method to calculate the average_salary of the employees
    def average_salary(self):
        return sum(emp.salary for emp in Emp.emps) / Emp.count

#FulltimeEmployee class inheriting the Employee class
class Fulltime_Emp(Emp):
        pass


#creating instances for above classes
emp1=Emp("harry potter","potter",80000,"IT")
emp2=Emp("ron weasly","weasly",70000,"Marketing")
fulltime_emp1 = Fulltime_Emp("albus james", "james", 70000, "Developer")


#Accessing the classes using instances
print(emp1.average_salary())
print(emp2.average_salary())
print(fulltime_emp1.average_salary())
```

```
73333.33333333333
73333.33333333333
73333.33333333333
```

2. 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)



```
fulltime_emp1 = Fulltime_Emp("albus james", "james", 70000, "Developer")

#Accessing the classes using instances
print(emp1.average_salary())
print(emp2.average_salary())
print(fulltime_emp1.average_salary())
```

```
73333.33333333333
73333.33333333333
73333.33333333333
```

In [19]:
```
import numpy as np #importing the numpy module

sampl=np.random.uniform(low=1,high=20,size=20) #generating the random float values between 1 to 20
reshape_arr=sampl.reshape((4,5)) #reshaping the vector to 4*5 dimension
print(reshape_arr)
#replacing the max values with 0s in each row
np.where(reshape_arr== np.max(reshape_arr, axis=1, keepdims=True), 0, reshape_arr)
```

```
[[15.04806671 12.9104063   2.94282347 14.79454723  6.06022713]
 [15.57399932  4.60895868  5.18863803  3.11338908  9.22860071]
 [16.12947314 13.0645834  17.86383669 13.25166203 11.12168525]
 [16.01348756 13.16808041 14.45857915 17.33972971 17.55899538]]
```

Out[19]:
```
array([[ 0.        , 12.9104063 ,  2.94282347, 14.79454723,  6.06022713],
       [ 0.        ,  4.60895868,  5.18863803,  3.11338908,  9.22860071],
       [16.12947314, 13.0645834 ,  0.        , 13.25166203, 11.12168525],
       [16.01348756, 13.16808041, 14.45857915, 17.33972971,  0.        ]])
```

In [ ]: