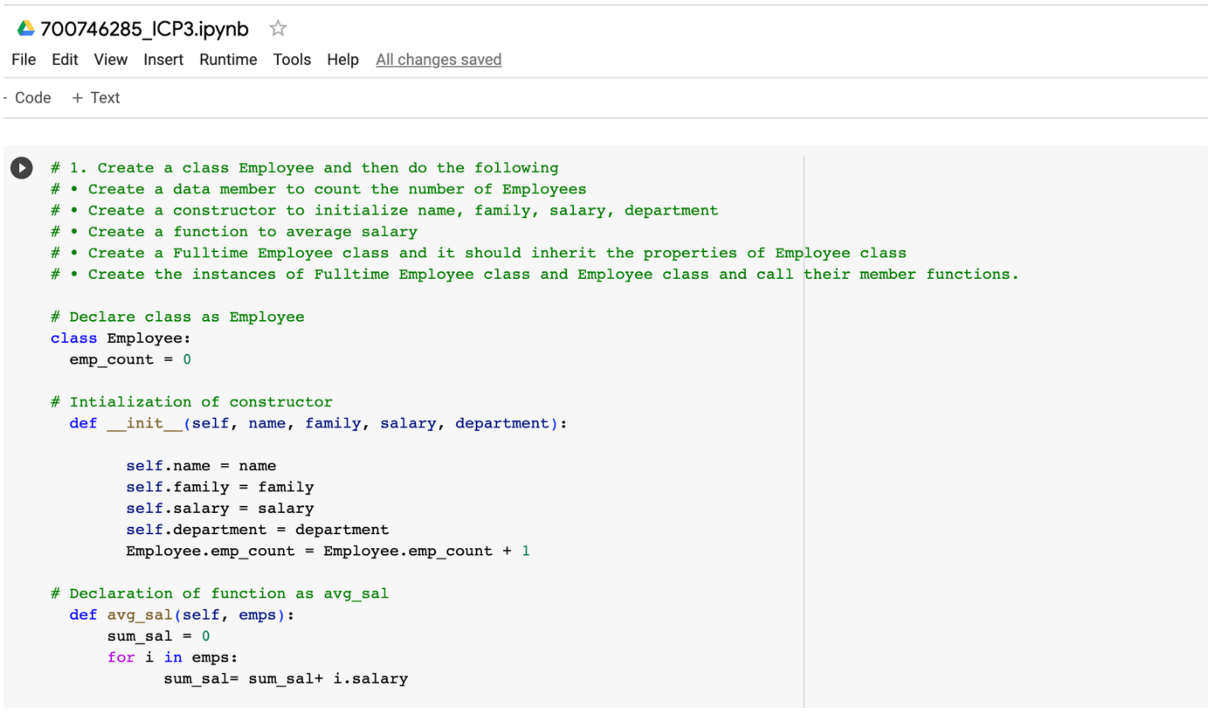
Neural Networks and Deep Learning
ICP3

Student Name: Sravani Lankala
Student Id: 700746285

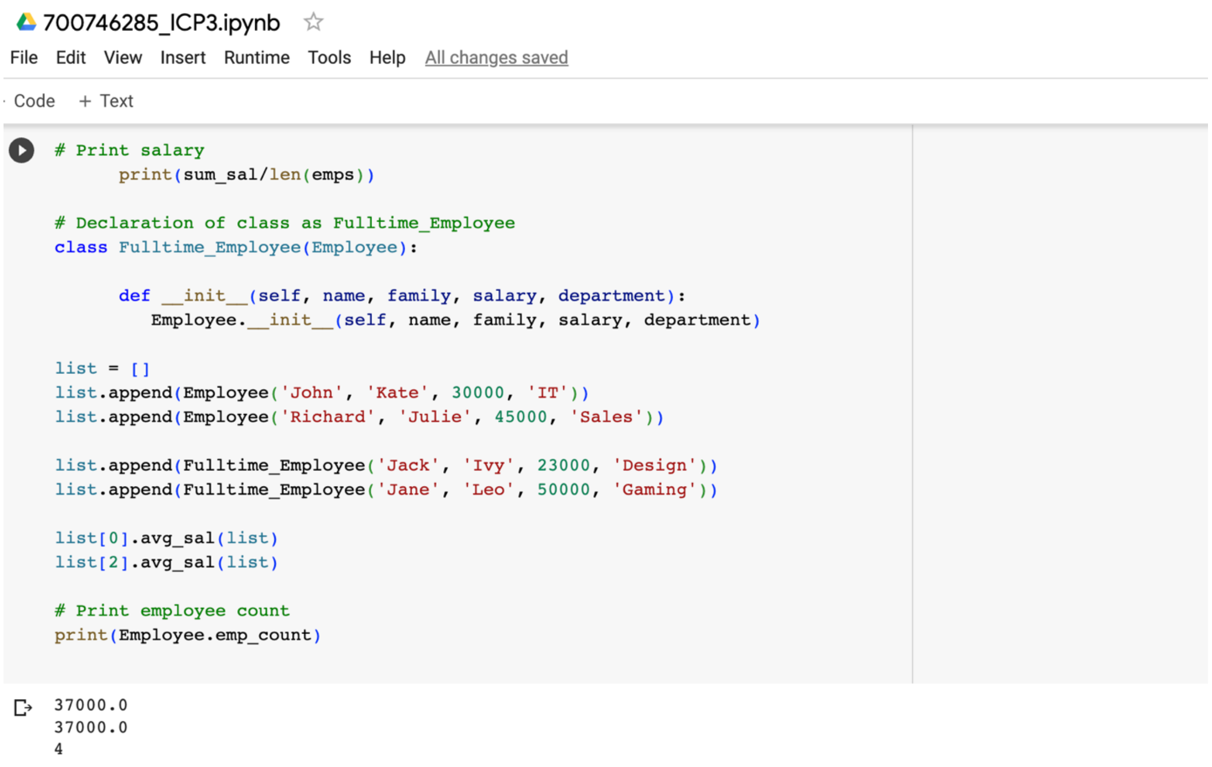
GitHub Link: https://github.com/sravanilankala/NNDL_ICP3_Fall2023

Video Link: https://drive.google.com/file/d/1M26O6sBgl33967C1TfbK0Spga-pGnFW5/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.



```
# 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.

# Declare class as Employee
class Employee:
  emp_count = 0

# Intialization of constructor
  def __init__(self, name, family, salary, department):

        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.emp_count = Employee.emp_count + 1

# Declaration of function as avg_sal
  def avg_sal(self, emps):
      sum_sal = 0
      for i in emps:
            sum_sal= sum_sal+ i.salary
```

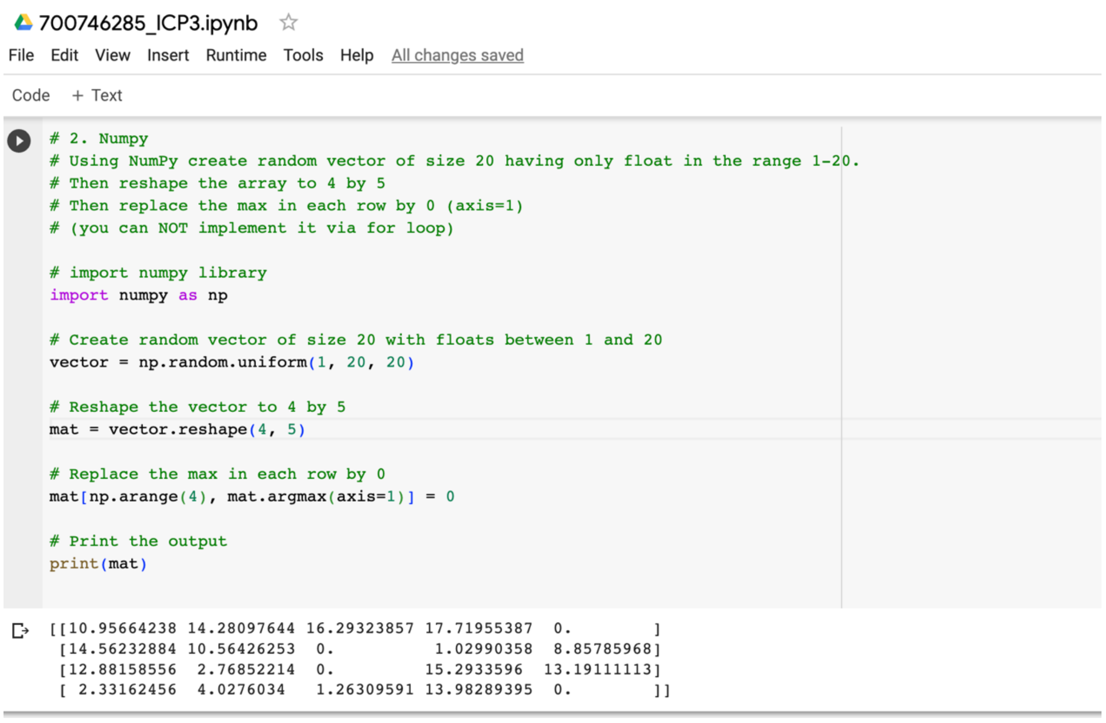


```
# Print salary
      print(sum_sal/len(emps))

# Declaration of class as Fulltime_Employee
class Fulltime_Employee(Employee):

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

list = []
list.append(Employee('John', 'Kate', 30000, 'IT'))
list.append(Employee('Richard', 'Julie', 45000, 'Sales'))

list.append(Fulltime_Employee('Jack', 'Ivy', 23000, 'Design'))
list.append(Fulltime_Employee('Jane', 'Leo', 50000, 'Gaming'))

list[0].avg_sal(list)
list[2].avg_sal(list)

# Print employee count
print(Employee.emp_count)
```

```
37000.0
37000.0
4
```

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)



```
# 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 library
import numpy as np

# Create random vector of size 20 with floats between 1 and 20
vector = np.random.uniform(1, 20, 20)

# Reshape the vector to 4 by 5
mat = vector.reshape(4, 5)

# Replace the max in each row by 0
mat[np.arange(4), mat.argmax(axis=1)] = 0

# Print the output
print(mat)
```

```
[[10.95664238 14.28097644 16.29323857 17.71955387  0.        ]
 [14.56232884 10.56426253  0.          1.02990358  8.85785968]
 [12.88158556  2.76852214  0.         15.2933596  13.19111113]
 [ 2.33162456  4.0276034   1.26309591 13.98289395  0.        ]]
```