https://github.com/jxc84430/NeuralNetworks/tree/main/ICP_3

Spring 2023 : CS5720

Neural Networks & Deep Learning ICP_3 : Jahnavi Chadalavada (700728443)

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.

In [1]:
```
class Employee:
    __no_of_employees=0
    __total_salary=0

    def __init__(self, name,family,salary,department):
        Employee.__no_of_employees=Employee.__no_of_employees+1
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.__total_salary = Employee.__total_salary+self.salary

    @staticmethod
    def employee_count():
        return print("\nTotal Number of Employees : ",Employee.__no_of_employees)

    @staticmethod
    def average_salary():
        return print("Average salary of All Employees",Employee.__total_salary / Employee.__no_of_employees)


    def display_employee_info(self):
        print(f"\n****Employee Details**** :\nName : {self.name} \nFamily : {self.family} \nSalary : {self.salary} \
```

In [2]:
```
class FulltimeEmployee(Employee):

    __no_of_fulltime_employees=0
    __total_salary=0

    def __init__(self,name,family,salary,department):
        FulltimeEmployee.__no_of_fulltime_employees=FulltimeEmployee.__no_of_fulltime_employees+1
        FulltimeEmployee.__total_salary = FulltimeEmployee.__total_salary+salary
        super().__init__(name,family,salary,department)

    def is_fulltime_employee(self):
        print(f"{self.name} {self.family} is FullTime Employee")

    @staticmethod
    def employee_count():
        return print("\nTotal Number of FullTime Employees : ",FulltimeEmployee.__no_of_fulltime_employees)

    @staticmethod
    def average_salary():
        return print("Average salary of FullTime Employees",FulltimeEmployee.__total_salary / FulltimeEmployee.__no_
```

In [3]:
```
john = Employee("John","Williams",5000,"IT")
tom = Employee("Tom","Smith",8000,"HR")
jerry = Employee("Jerry","Johnson",6000,"Accounts")
david = Employee("David","Richards",9000,"Finance")
julie = Employee("Julie","Martin",15000,"IT")

john.display_employee_info()

kevin = FulltimeEmployee("Kevin","Williams",9000,"IT")
natasha = FulltimeEmployee("Natasha","Smith",7000,"HR")
sophie = FulltimeEmployee("Sophie","Johnson",10000,"Accounts")
amy = FulltimeEmployee("Amy","Richards",4000,"Finance")
destiny = FulltimeEmployee("Destiny","Martin",7500,"IT")

destiny.display_employee_info()
destiny.is_fulltime_employee()

Employee.employee_count()
Employee.average_salary()

FulltimeEmployee.employee_count()
FulltimeEmployee.average_salary()
```

```
****Employee Details**** :
Name : John
Family : Williams
Salary : 5000
Department : IT

****Employee Details**** :
Name : Destiny
Family : Martin
Salary : 7500
Department : IT
Destiny Martin is FullTime Employee

Total Number of Employees :  10
Average salary of All Employees 8050.0

Total Number of FullTime Employees :  5
Average salary of FullTime Employees 7500.0
```

2. Numpy

Using NumPy create random vector of size 20 having only float in the range 1-20.

In [4]:
```
import numpy as np

random_vector = np.random.uniform(1,20,20)

print(random_vector)

random_vector.shape
```

```
[18.78445616  8.68894756 11.56510356  2.72486691  4.25105668 17.36096306
  2.45370507 17.13850476 15.41810492 14.13017637  1.5799388  14.03257593
  2.66705713 19.62284672 19.00587859 15.6525134  19.65967879  4.2423449
  5.93219885 17.79027902]
```

Out[4]: (20,)

Then reshape the array to 4 by 5

In [5]:
```
reshape_array=random_vector.reshape(4,5)

print(reshape_array)

reshape_array.shape
```

```
[[18.78445616  8.68894756 11.56510356  2.72486691  4.25105668]
 [17.36096306  2.45370507 17.13850476 15.41810492 14.13017637]
 [ 1.5799388  14.03257593  2.66705713 19.62284672 19.00587859]
 [15.6525134  19.65967879  4.2423449   5.93219885 17.79027902]]
```

Out[5]: (4, 5)

Then replace the max in each row by 0 (axis=1)

In [6]:
```
replace_max=np.where(np.isin(reshape_array,np.amax(reshape_array, axis=1)), 0, reshape_array)

print(replace_max)
```

```
[[ 0.          8.68894756 11.56510356  2.72486691  4.25105668]
 [ 0.          2.45370507 17.13850476 15.41810492 14.13017637]
 [ 1.5799388  14.03257593  2.66705713  0.         19.00587859]
 [15.6525134   0.          4.2423449   5.93219885 17.79027902]]
```