ASSIGNMENT-3

STUDENT ID: 700744730

GITHUB LINK:

https://github.com/vamsikrishnaremala/700744730_NNDL_ICP3/blob/main/README.md

Drive link:

https://drive.google.com/drive/folders/1kqM7sETurasNA27UmR8aUNewPsTLUMhh

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:
    employees_count = 0

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

    @classmethod
    def avg_salary(cls, employees):
        total_sal = sum(employee.salary for employee in employees)
        if len(employees) > 0:
          return total_sal / len(employees)
        else:
          return 0
```

```
# • Create a Fulltime Employee class
class FulltimeEmployee(Employee):
    def __init__(self, name, family, salary, department):
        super().__init__(name, family, salary, department)
```

```
# Creating the instances for Employee class
employee1 = Employee("vamsi", "remala", 50000, "HR")
employee2 = Employee("ramesh", "singh", 60000, "Finance")
employee3 = Employee("zaheer", "khan", 55000, "IT")
```

```
# Creating the instances for FulltimeEmployee class
fulltime_employee1 = FulltimeEmployee("Rakesh", "prana", 70000, "Marketing")
fulltime_employee2 = FulltimeEmployee("iqbal", "hussain", 75000, "Sales")
```

```
# Calling the respective member functions
employees = [employee1, employee2, employee3, fulltime_employee1, fulltime_employee2]
avg_salary = Employee.avg_salary(employees)

print(f"Employees Count: {Employee.employees_count}")
print(f"Average salary: ${avg_salary:.2f}")
```

Input and Output:

```
Employees Count: 5
Average salary: $62000.00
```

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 as np

# Creating a random vector here
random_vector = np.random.uniform(1, 20, 20)
print(random_vector)
```

```
[ 9.12006235 17.51146296 16.54937968  4.84705882  2.04472335  3.47237538
 16.23077195 11.08015899  4.69350611 13.4158621  13.55879993 10.15306925
 11.18957902  8.66530486 13.3313942  15.46472363 14.64905792  5.05153076
 16.59219325 15.56020471]
```

```
# Reshaping the generated array to 4 by 5
reshape_array = random_vector.reshape(4, 5)
print(reshape_array)
```

```
[[ 9.12006235 17.51146296 16.54937968  4.84705882  2.04472335]
 [ 3.47237538 16.23077195 11.08015899  4.69350611 13.4158621 ]
 [13.55879993 10.15306925 11.18957902  8.66530486 13.3313942 ]
 [15.46472363 14.64905792  5.05153076 16.59219325 15.56020471]]
```

```
# Insteading of looping, we are getting the indices of the maximum values in each row of array

indices_position = np.argmax(reshape_array, axis=1)
print(indices_position)
```

```
[1 1 0 3]
```

```
# Replace the maximum values with 0
# Using np.arrange function, we are generating an array of values from 0 to 3
# Using advanced indexing, here we are combing the postion indices and our newly generated indices and assigning 0:

reshape_array[np.arange(4), indices_position] = 0

print(reshape_array)
```

```
[[ 9.12006235  0.         16.54937968  4.84705882  2.04472335]
 [ 3.47237538  0.         11.08015899  4.69350611 13.4158621 ]
 [ 0.         10.15306925 11.18957902  8.66530486 13.3313942 ]
 [15.46472363 14.64905792  5.05153076  0.         15.56020471]]
```