Sucharitha Aeluri

700744961

NEURAL NETWORK AND DEEP LEARNING ASSIGNMENT-3

GITHUB LINK: - https://github.com/SucharithaAeluri/NNAssignment3.git

RECORDINGLINK:
https://drive.google.com/file/d/1T0_ygjtDKO3lcBBM6CwGd15eGHfc4JRY/view?usp=drive_link

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:
    num_employees = 0              #Create a data member to count the number of Employees.

    def __init__(self, name, family, salary, department):   #Create a constructor to initialize name, family, salary, department
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.num_employees += 1     #Count the employees

    @staticmethod
    def average_salary(employees):      #Create a function to average salary.
        total_salary = sum(employee.salary for employee in employees)
        return total_salary / len(employees) if len(employees) > 0 else 0

    def display_employee_info(self):
      print(f"Name: {self.name}")
      print(f"Family: {self.family}")
      print(f"Salary: ${self.salary}")
      print(f"Department: {self.department}")
```

```
class FulltimeEmployee(Employee):       #Create a Fulltime Employee class and it should inherit the properties of Employee class
    def __init__(self, name, family, salary, department):
        super().__init__(name, family, salary, department)

    def display_employee_info(self):
        super().display_employee_info()

# Create instances of Employee class
employee1 = Employee("John", "Doe", 10000, "HR")
employee2 = Employee("Jane", "Smith", 10000, "Finance")

# Create instances of FulltimeEmployee class
fulltime_employee1 = FulltimeEmployee("Alice", "Brown", 10000, "Marketing")
fulltime_employee2 = FulltimeEmployee("Charlie", "Wilson", 10000, "Sales")

# Calculate the average salary of all employees
all_employees = [employee1, employee2, fulltime_employee1, fulltime_employee2]
avg_salary = Employee.average_salary(all_employees)

# Print the number of employees and the average salary
print(f"Total number of employees: {Employee.num_employees}\n")
print(f"Average salary of all employees: ${avg_salary:.2f}\n")
```

```
# Call member functions to display employee information
print("Employee Information:")
employee1.display_employee_info()
print("\n")
employee2.display_employee_info()
print("\n")
fulltime_employee1.display_employee_info()
print("\n")
fulltime_employee2.display_employee_info()
```

Output:-

```
Total number of employees: 4

Average salary of all employees: $10000.00

Employee Information:
Name: John
Family: Doe
Salary: $10000
Department: HR


Name: Jane
Family: Smith
Salary: $10000
Department: Finance


Name: Alice
Family: Brown
Salary: $10000
Department: Marketing


Name: Charlie
Family: Wilson
Salary: $10000
Department: Sales
```

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)

```
import numpy as np           #importing numpy library

x=np.arange(1,21,dtype=float)   #Using NumPy create random vector of size 20 having only float in the range 1-20
print("Numbers range from 1 to 20:- \n", x)

y=x.reshape(4,5)            #reshape the array to 4 by 5
print("\nReshape:- \n", y)

z=np.where(np.isin(y, y.max(axis=1)), 0, y)      #replace the max in each row by 0 (axis=1)
print("\nReplace max value with 0:- \n", z)
```

Output:-

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

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

Replace max value with 0:-
 [[ 1.  2.  3.  4.  0.]
 [ 6.  7.  8.  9.  0.]
 [11. 12. 13. 14.  0.]
 [16. 17. 18. 19.  0.]]
```