**Spring 2024: CS5720 Neural Networks & Deep Learning - ICP-3**
**Bhanu Chandrika Lakkimsetti (700747439)**

GitHub Link: https://github.com/bhanuchandrika99/NNDL_ICP_3

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.

**Solution:**

jupyter ICP_3_Spring24 Last Checkpoint: 38 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [1]:

```python
class Employee:
    num_employees = 0

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

    @staticmethod
    def average_salary(salary_list):
        if len(salary_list) == 0:
            return 0
        return sum(salary_list) / len(salary_list)


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


# Creating instances of Employee and FulltimeEmployee classes
employee1 = Employee("Jimmy Carter", "Carter", 40000, "Human Resources")
employee2 = Employee("Ryan Gosling", "Gosling", 60000, "Finance")
```

```python
fulltime_employee1 = FulltimeEmployee("Tom Hanks", "Hanks", 70000, "Marketing")
fulltime_employee2 = FulltimeEmployee("Bobby Brown", "Brown", 80000, "Engineering")

# Calling member functions
print("Number of employees:", Employee.num_employees)

salaries = [employee1.salary, employee2.salary, fulltime_employee1.salary, fulltime_employee2.salary]
print("Average salary:", Employee.average_salary(salaries))
```

```
Number of employees: 4
Average salary: 62500.0
```

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)

In [5]:

```python
import numpy as np

# Create a random vector of size 20 with floats in the range 1-20
random_vector = np.random.uniform(1, 20, 20)

# Reshape the array to 4 by 5
reshaped_array = random_vector.reshape(4, 5)

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

print(reshaped_array)
```

```
[[ 3.82742505  0.         17.61506028  5.21816866  6.57518762]
 [ 5.72294301  4.08979337  0.         12.8257235   9.54239631]
 [ 2.91408116  3.1310031   0.          5.82377181  1.07556005]
 [ 0.          2.33555267  9.64495385 14.15770628  2.23730037]]
```