In [1]:
```
# Import Numpy Library
import numpy as np
```

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.ions.

In [2]:
```
# created class Employee
class Employee():
    # create data members to keep count of employee count and list of salaries
    emp_count = 0

    # constructor to initilize the variables
    def __init__(self,name,family,salary,department):
        self.name = name,
        self.family = family,
        self.salary = salary,
        self.department = department,
        Employee.emp_count += 1

    # funtion to return average of salary
    def avg_salary(employees:list):
        # calculate salaries average
        salaries_list = [emp.salary[0] for emp in employees]
        # print(emp.salary[0] for emp in employees)
        avg_sal = sum(salaries_list)/Employee.emp_count     # method 1 : generic
        # avg_sal = np.mean(salaries_list)                      # method 2 : using numpy
        return avg_sal

# created FullTimeEmployee class and inherit the properties from Employee class
class FullTimeEmployee(Employee):
    def __init__(self,name,family,salary,department):
        # calling parent class constructor
        super().__init__(name,family,salary,department)

employees = []
employees.append(Employee("Naveen","Indluru",40000,"analytics"))
employees.append(Employee("Anvesh","Kalikiri",74000,"CTO"))

employees.append(FullTimeEmployee("Naveen Reddy","Indluru",50000,"data science"))
employees.append(FullTimeEmployee("Jagadeesh","Kethu",84000,"data analytics"))


print("using employee class",Employee.avg_salary(employees))
print("using fulltime employee class",FullTimeEmployee.avg_salary(employees))
```

```
using employee class 62000.0
using fulltime employee class 62000.0
```

1. 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 [3]:
```
arr = np.random.uniform(1,20,20) # created a random vector of size 20 in the range of 1
arr = arr.reshape(4,5) # reshaped the vector to array of (4,5) shape
```

In [4]:
```
# Task : replace the max in each row by 0 (axis = 1)

# print array before replacement
print("Array before :\n\n",arr,end="\n\n\n")

# get max elements of each row and reshape it
each_row_max = np.max(arr,axis = 1).reshape(-1,1)

arr[arr == each_row_max] = 1

# print array after replacement
print("Array after replace the max element of each row with 1: \n\n", arr)
```

```
Array before :

 [[ 9.41136559  8.29098713 12.23375169 19.59203541 18.21838439]
 [13.12865465  6.78789517 19.43663057 13.43142537  3.50417629]
 [ 8.22360626 13.26835668 19.02461584  3.60125819  2.0091133 ]
 [17.12966119  8.56089488 19.92597494  9.11390514  5.06372136]]


Array after replace the max element of each row with 1:

 [[ 9.41136559  8.29098713 12.23375169  1.          18.21838439]
 [13.12865465  6.78789517  1.         13.43142537  3.50417629]
 [ 8.22360626 13.26835668  1.          3.60125819  2.0091133 ]
 [17.12966119  8.56089488  1.          9.11390514  5.06372136]]
```

In [ ]:

In [ ]: