In class Programming Assignment - 2

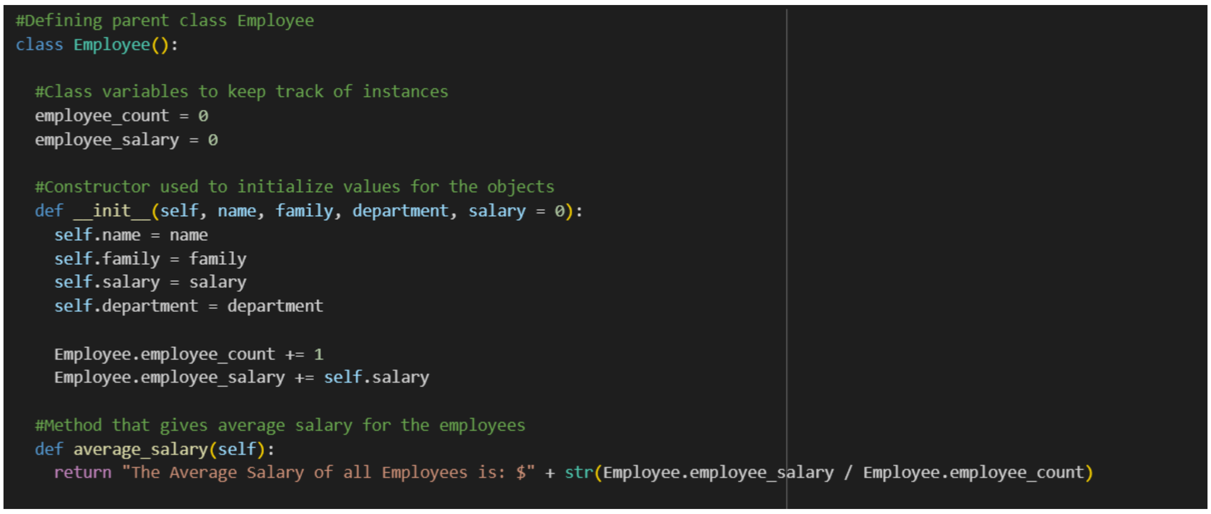
GitHub Link: https://github.com/HarshithaBoyapati2002/nn-icp2

Problem 1: To demonstrate inheritance in Python

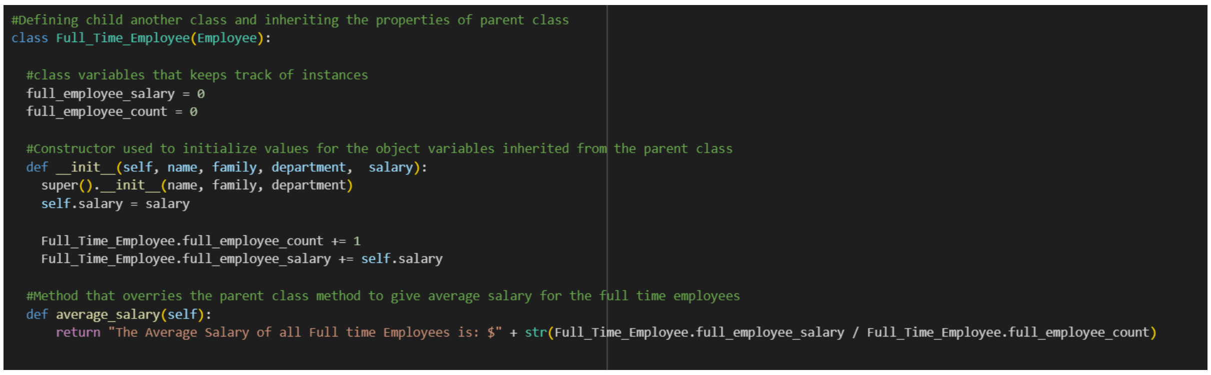
Solution:

1. Created a class "Employee" and data members to count the no. of instances created for the class Employee and its child class.
2. A constructor is created to initialize the values for the objects and a function "average_salary" is created to calculate average for the employees.
3. Child class "Full_Time_Employee" is created that inherits all the properties of the parent class "Employee" and data members are created to keep track of all the objects.
4. Constructor is defined to initialize the values for the objects and method "average_salary" is created to calculate average salary for the full-time employees.
5. Driver code contains the initialization of objects for both the classes and member functions are called using the objects.

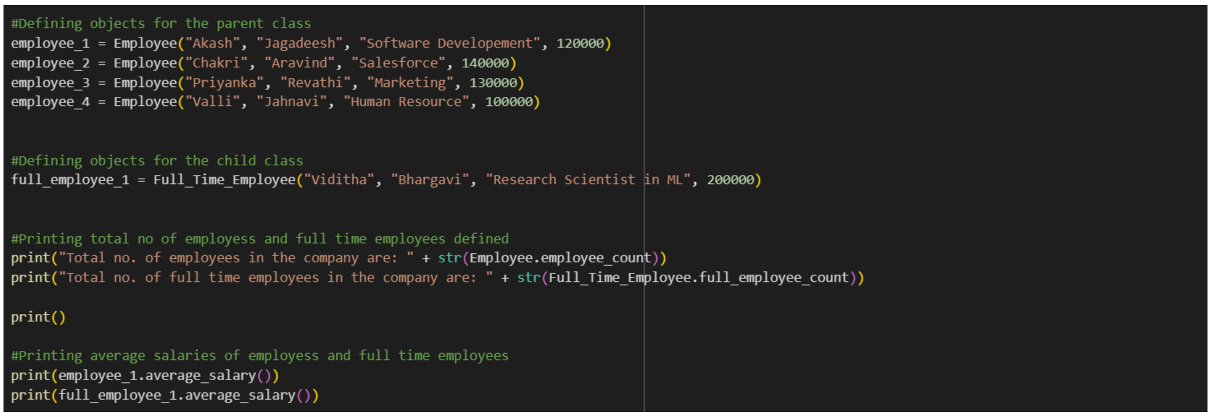
Code:



```python
#Defining parent class Employee
class Employee():

  #Class variables to keep track of instances
  employee_count = 0
  employee_salary = 0

  #Constructor used to initialize values for the objects
  def __init__(self, name, family, department, salary = 0):
    self.name = name
    self.family = family
    self.salary = salary
    self.department = department

    Employee.employee_count += 1
    Employee.employee_salary += self.salary

  #Method that gives average salary for the employees
  def average_salary(self):
    return "The Average Salary of all Employees is: $" + str(Employee.employee_salary / Employee.employee_count)
```

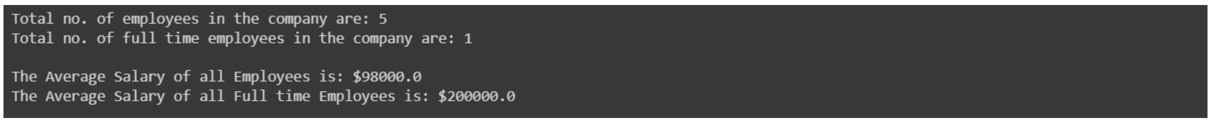


```python
#Defining child another class and inheriting the properties of parent class
class Full_Time_Employee(Employee):

  #class variables that keeps track of instances
  full_employee_salary = 0
  full_employee_count = 0

  #Constructor used to initialize values for the object variables inherited from the parent class
  def __init__(self, name, family, department,  salary):
    super().__init__(name, family, department)
    self.salary = salary

    Full_Time_Employee.full_employee_count += 1
    Full_Time_Employee.full_employee_salary += self.salary

  #Method that overries the parent class method to give average salary for the full time employees
  def average_salary(self):
      return "The Average Salary of all Full time Employees is: $" + str(Full_Time_Employee.full_employee_salary / Full_Time_Employee.full_employee_count)
```



```
#Defining objects for the parent class
employee_1 = Employee("Akash", "Jagadeesh", "Software Developement", 120000)
employee_2 = Employee("Chakri", "Aravind", "Salesforce", 140000)
employee_3 = Employee("Priyanka", "Revathi", "Marketing", 130000)
employee_4 = Employee("Valli", "Jahnavi", "Human Resource", 100000)


#Defining objects for the child class
full_employee_1 = Full_Time_Employee("Viditha", "Bhargavi", "Research Scientist in ML", 200000)


#Printing total no of employess and full time employees defined
print("Total no. of employees in the company are: " + str(Employee.employee_count))
print("Total no. of full time employees in the company are: " + str(Full_Time_Employee.full_employee_count))

print()

#Printing average salaries of employess and full time employees
print(employee_1.average_salary())
print(full_employee_1.average_salary())
```

Output:



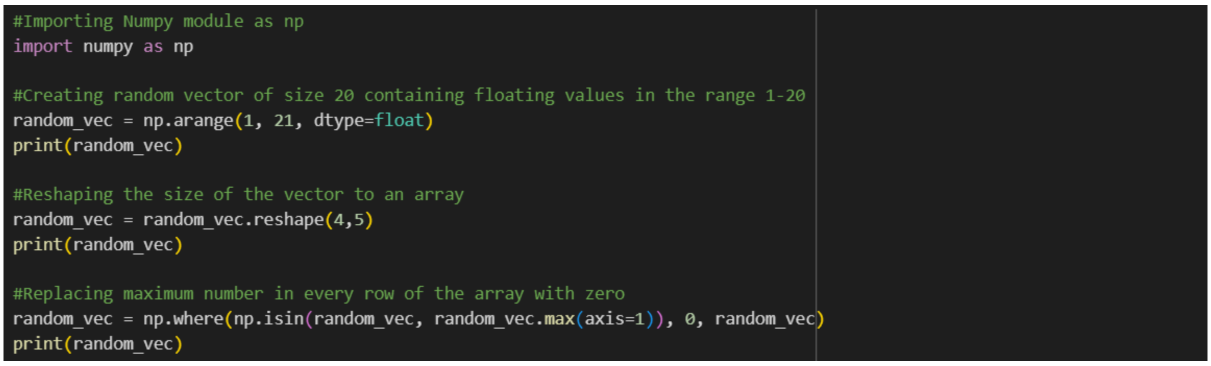
```
Total no. of employees in the company are: 5
Total no. of full time employees in the company are: 1

The Average Salary of all Employees is: $98000.0
The Average Salary of all Full time Employees is: $200000.0
```

Problem 2: To create a random vector of size 20 that contains only floating-point values in the range 1-20, reshaping the vector size from 1x20 to 4x5 and then replacing maximum value in each row to zero.
Input:
Output: NumPy array

Solution:

1. Created a random vector of size 20 with floating-point values using np.arange() method with arguments, range 1-21, type of value (i.e. dtype) as float.
2. Vector is resized using np.reshape() method that takes size of new array as arguments.
3. Maximum value in each row is replaced with 0 using functions np.where() that returns 0 if value is true and remaining elements that has false value, and np.isin() that return Boolean values that is set to true for max value and false for remaining values.

Code:



```
#Importing Numpy module as np
import numpy as np

#Creating random vector of size 20 containing floating values in the range 1-20
random_vec = np.arange(1, 21, dtype=float)
print(random_vec)

#Reshaping the size of the vector to an array
random_vec = random_vec.reshape(4,5)
print(random_vec)

#Replacing maximum number in every row of the array with zero
random_vec = np.where(np.isin(random_vec, random_vec.max(axis=1)), 0, random_vec)
print(random_vec)
```

Output:

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
 19. 20.]
[[ 1.  2.  3.  4.  5.]
 [ 6.  7.  8.  9. 10.]
 [11. 12. 13. 14. 15.]
 [16. 17. 18. 19. 20.]]
[[ 1.  2.  3.  4.  0.]
 [ 6.  7.  8.  9.  0.]
 [11. 12. 13. 14.  0.]
 [16. 17. 18. 19.  0.]]
```