

# ICP-2

NAME: GOWTHAM SAI REDDY MADDIREDDY

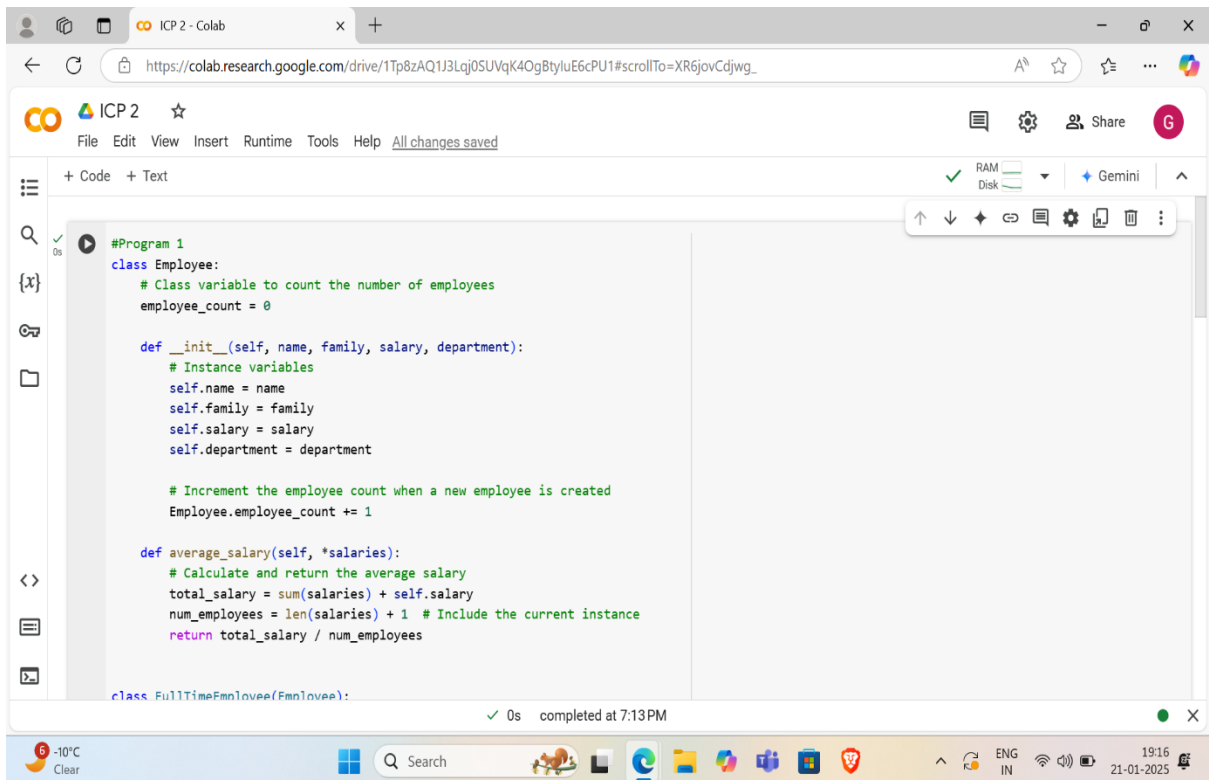
ID: 700759075

GITHUB LINK: [ICP-2/ICP\\_2.ipynb at main · Gowthamsai08/ICP-2](https://github.com/Gowthamsai08/ICP-2)

VIDEO LINK:

[https://drive.google.com/file/d/16BFxj0e0FXvzzfLHL8sYcJ91SPzOzhuk/view?usp=drive\\_link](https://drive.google.com/file/d/16BFxj0e0FXvzzfLHL8sYcJ91SPzOzhuk/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.



```
#Program 1
class Employee:
    # Class variable to count the number of employees
    employee_count = 0

    def __init__(self, name, family, salary, department):
        # Instance variables
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department

        # Increment the employee count when a new employee is created
        Employee.employee_count += 1

    def average_salary(self, *salaries):
        # Calculate and return the average salary
        total_salary = sum(salaries) + self.salary
        num_employees = len(salaries) + 1 # Include the current instance
        return total_salary / num_employees

class FullTimeEmployee(FullTimeEmployee):
```

The screenshot shows a Google Colab notebook interface. The top bar includes the Colab logo, a star icon, and a menu with options like File, Edit, View, Insert, Runtime, Tools, and Help. Below the menu is a toolbar with icons for RAM, Disk, and Gemini. The main area displays a Python code editor with the following code:   
  
#Program 1  
class Employee:  
 # Class variable to count the number of employees  
 employee\_count = 0  
  
 def \_\_init\_\_(self, name, family, salary, department):  
 # Instance variables  
 self.name = name  
 self.family = family  
 self.salary = salary  
 self.department = department  
  
 # Increment the employee count when a new employee is created  
 Employee.employee\_count += 1  
  
 def average\_salary(self, \*salaries):  
 # Calculate and return the average salary  
 total\_salary = sum(salaries) + self.salary  
 num\_employees = len(salaries) + 1 # Include the current instance  
 return total\_salary / num\_employees  
  
class FullTimeEmployee(FullTimeEmployee):

ICP 2 - Colab

https://colab.research.google.com/drive/1Tp8zAQ1J3Lqj0SUVqK4OgBtyluE6cPU1#scrollTo=XR6jovCdjwg\_

ICP 2

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
class FullTimeEmployee(Employee):
    # Additional properties for FullTimeEmployee can be added here

    def __init__(self, name, family, salary, department, fulltime_property):
        # Call the constructor of the base class (Employee)
        super().__init__(name, family, salary, department)

        # Additional property specific to FullTimeEmployee
        self.fulltime_property = fulltime_property

# Create instances of Employee class
employee1 = Employee("Gowtham sai M", "Family1", 60000, "DM")
employee2 = Employee("Teju A", "Family2", 70000, "HR")

# Call the average_salary function for Employee class
average_salary_employee = employee1.average_salary(employee2.salary)
print(f"Average Salary for Employees: {average_salary_employee}")

# Create instances of FullTimeEmployee class
fulltime_employee = FullTimeEmployee("keerthi", "Family3", 80000, "Software", "FullTimeProperty")

# Call the average_salary function for FullTimeEmployee class
average_salary_fulltime_employee = fulltime_employee.average_salary()
```

0s completed at 7:13 PM

-10°C Clear

Search

ENG IN

19:16 21-01-2025

ICP 2 - Colab

https://colab.research.google.com/drive/1Tp8zAQ1J3Lqj0SUVqK4OgBtyluE6cPU1#scrollTo=XR6jovCdjwg\_

ICP 2

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
# Create instances of Employee class
employee1 = Employee("Gowtham sai M", "Family1", 60000, "DM")
employee2 = Employee("Teju A", "Family2", 70000, "HR")

# Call the average_salary function for Employee class
average_salary_employee = employee1.average_salary(employee2.salary)
print(f"Average Salary for Employees: {average_salary_employee}")

# Create instances of FullTimeEmployee class
fulltime_employee = FullTimeEmployee("keerthi", "Family3", 80000, "Software", "FullTimeProperty")

# Call the average_salary function for FullTimeEmployee class
average_salary_fulltime_employee = fulltime_employee.average_salary()
print(f"Average Salary for FullTimeEmployee: {average_salary_fulltime_employee}")

# Print the total count of employees
print(f"Total Employees: {Employee.employee_count}")
```

Average Salary for Employees: 65000.0  
Average Salary for FullTimeEmployee: 80000.0  
Total Employees: 3

[10] import numpy as np

0s completed at 7:13 PM

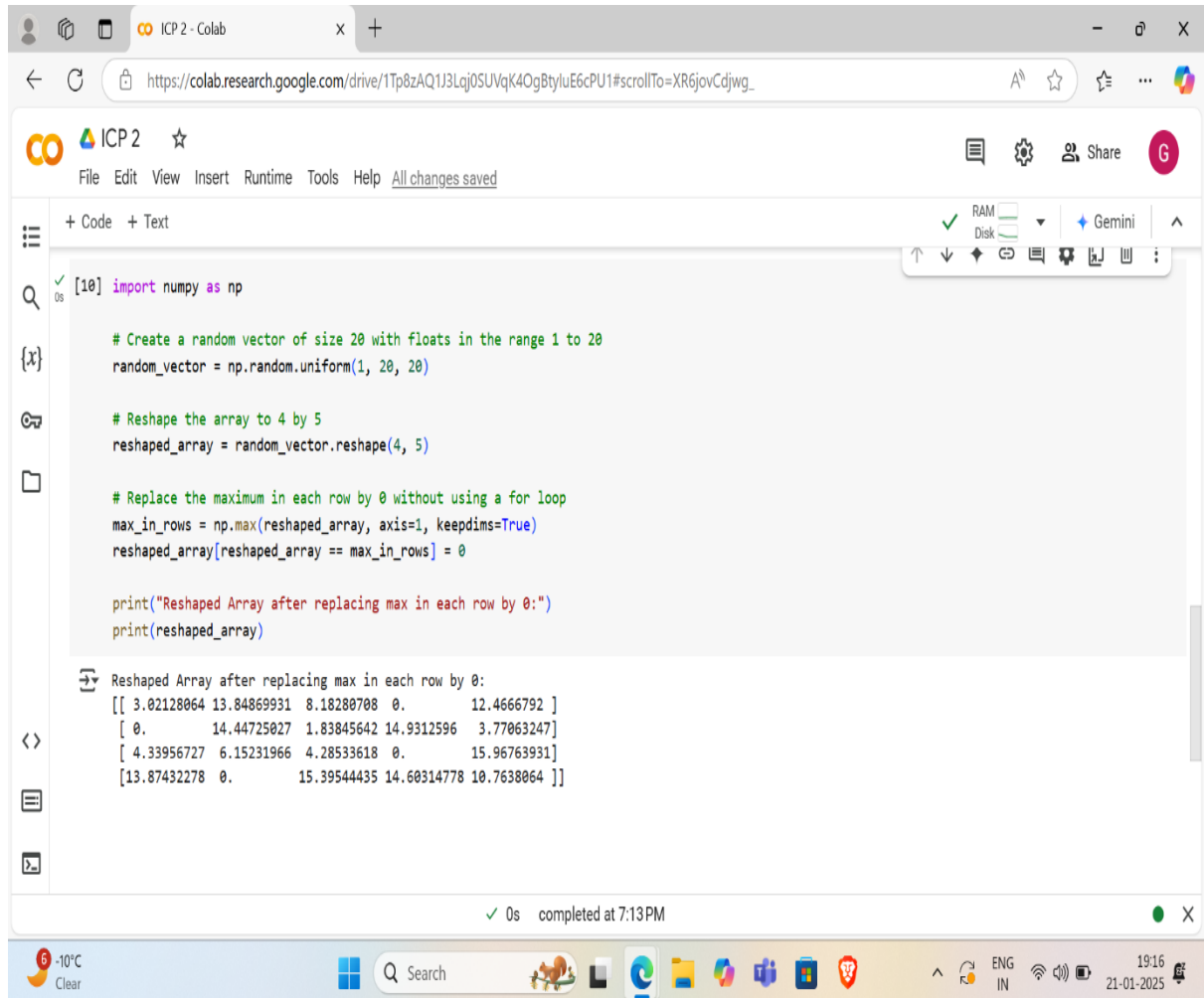
-10°C Clear

Search

ENG IN

19:16 21-01-2025

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)



The screenshot shows a Google Colab notebook interface. The code cell contains the following Python code:

```
[10] import numpy as np

# Create a random vector of size 20 with floats in the range 1 to 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 maximum in each row by 0 without using a for loop
max_in_rows = np.max(reshaped_array, axis=1, keepdims=True)
reshaped_array[reshaped_array == max_in_rows] = 0

print("Reshaped Array after replacing max in each row by 0:")
print(reshaped_array)
```

The output of the code is displayed below the code cell:

```
Reshaped Array after replacing max in each row by 0:
[[ 3.02128064 13.84869931  8.18280708  0.         12.4666792 ]
 [ 0.         14.44725027  1.83845642 14.9312596   3.77063247]
 [ 4.33956727  6.15231966  4.28533618  0.         15.96763931]
 [13.87432278  0.         15.39544435 14.60314778 10.7638064 ]]
```

The status bar at the bottom of the Colab interface indicates that the code was completed at 7:13 PM.