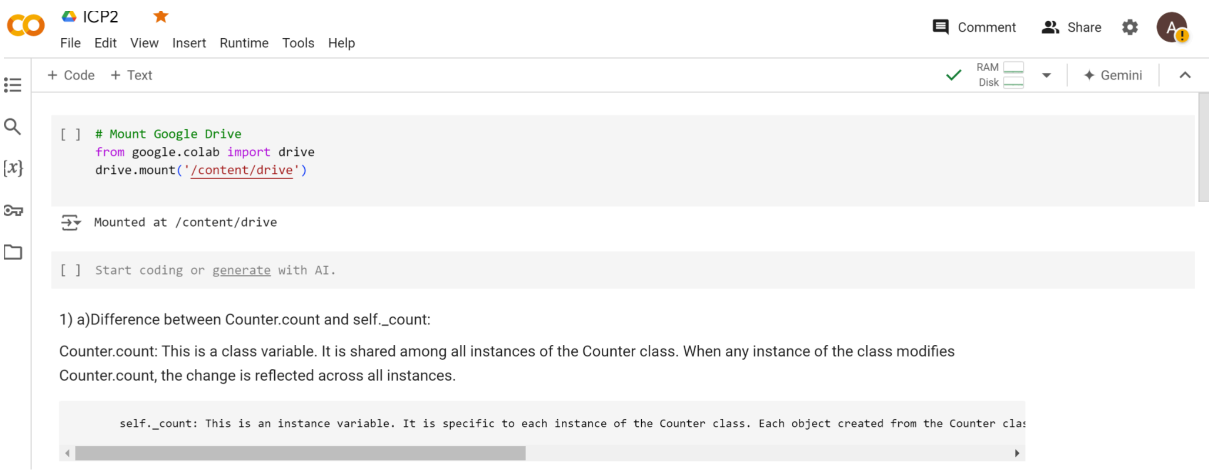
ICP 2



```
[ ] # Mount Google Drive
    from google.colab import drive
    drive.mount('/content/drive')
```

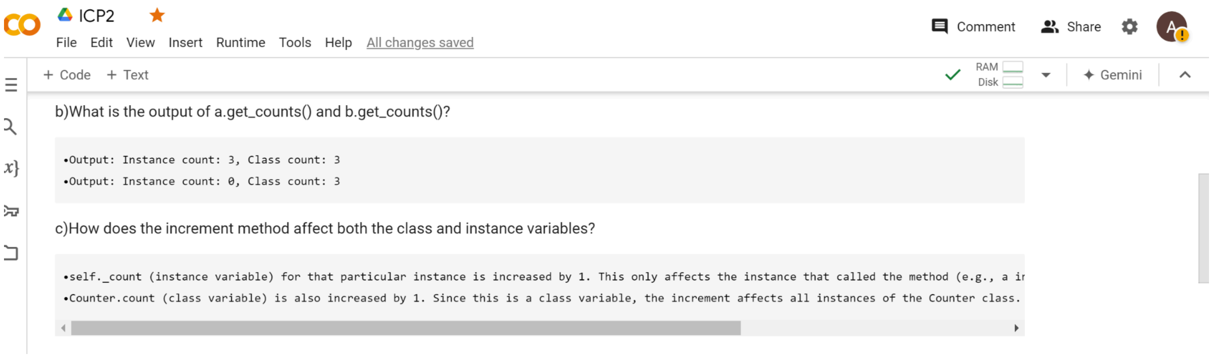
```
Mounted at /content/drive
```

```
[ ] Start coding or generate with AI.
```

1) a)Difference between Counter.count and self._count:

Counter.count: This is a class variable. It is shared among all instances of the Counter class. When any instance of the class modifies Counter.count, the change is reflected across all instances.

```
    self._count: This is an instance variable. It is specific to each instance of the Counter class. Each object created from the Counter clas
```

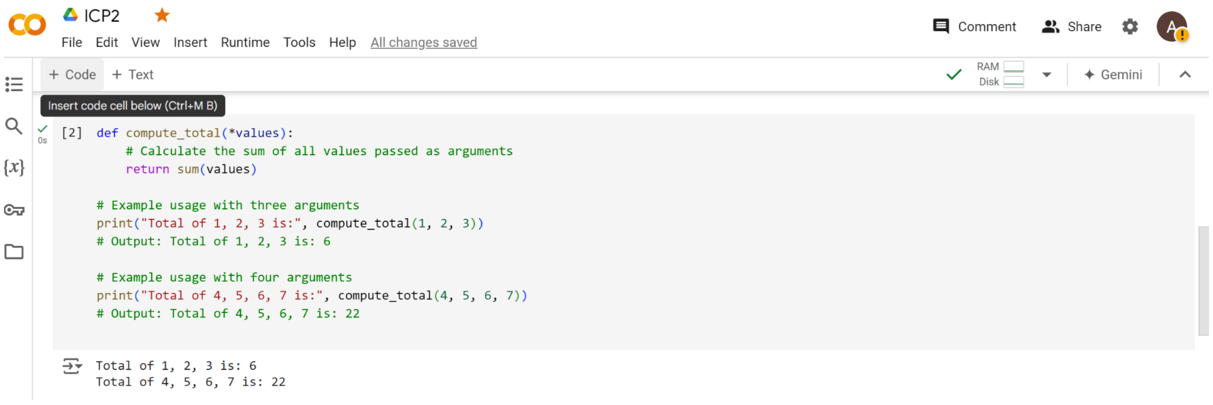


b)What is the output of a.get_counts() and b.get_counts()?

```
•Output: Instance count: 3, Class count: 3
•Output: Instance count: 0, Class count: 3
```

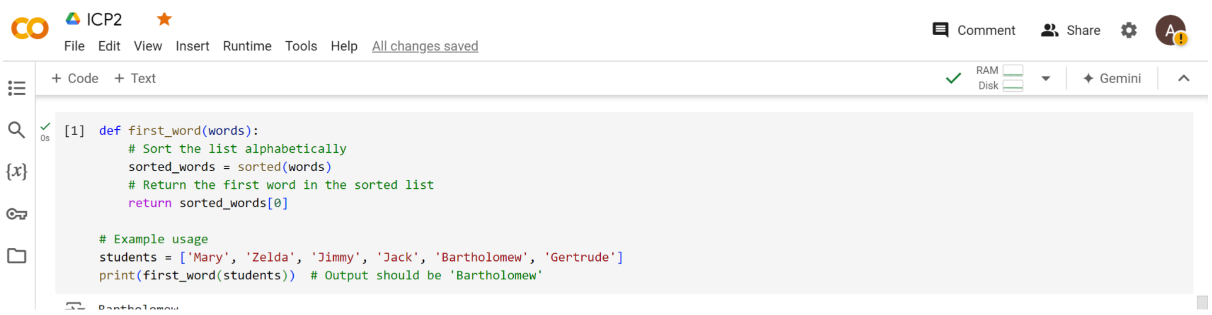
c)How does the increment method affect both the class and instance variables?

```
•self._count (instance variable) for that particular instance is increased by 1. This only affects the instance that called the method (e.g., a ir
•Counter.count (class variable) is also increased by 1. Since this is a class variable, the increment affects all instances of the Counter class.
```

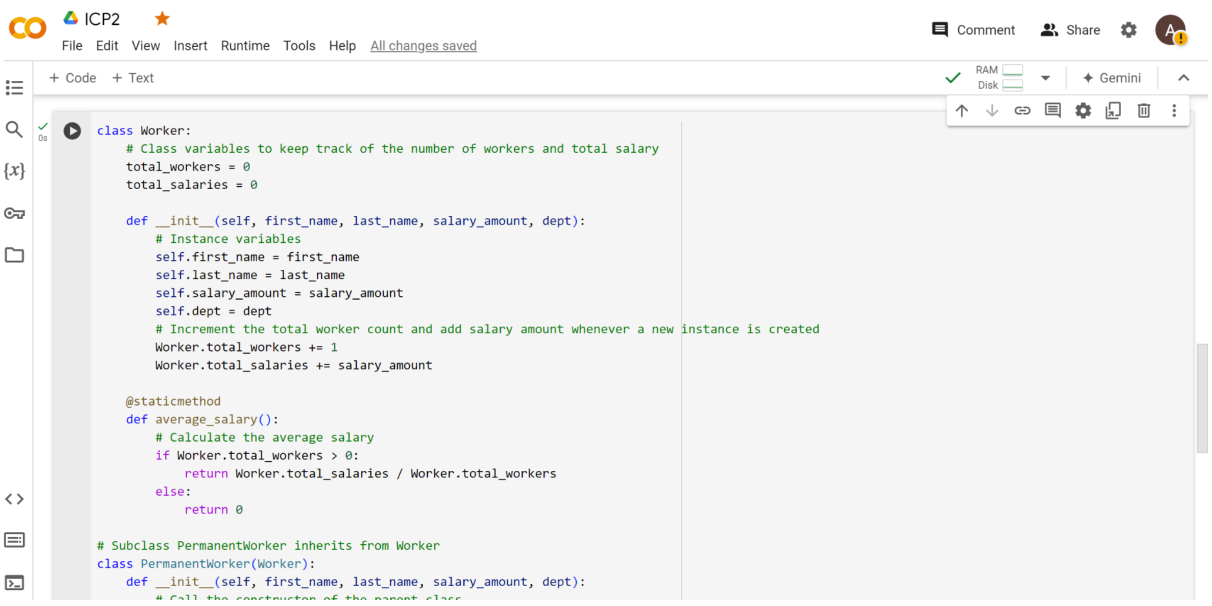

ICP2
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Insert code cell below (Ctrl+M B)
RAM Disk
Gemini

```
[2] def compute_total(*values):
        # Calculate the sum of all values passed as arguments
        return sum(values)

    # Example usage with three arguments
    print("Total of 1, 2, 3 is:", compute_total(1, 2, 3))
    # Output: Total of 1, 2, 3 is: 6

    # Example usage with four arguments
    print("Total of 4, 5, 6, 7 is:", compute_total(4, 5, 6, 7))
    # Output: Total of 4, 5, 6, 7 is: 22
```

```
Total of 1, 2, 3 is: 6
Total of 4, 5, 6, 7 is: 22
```



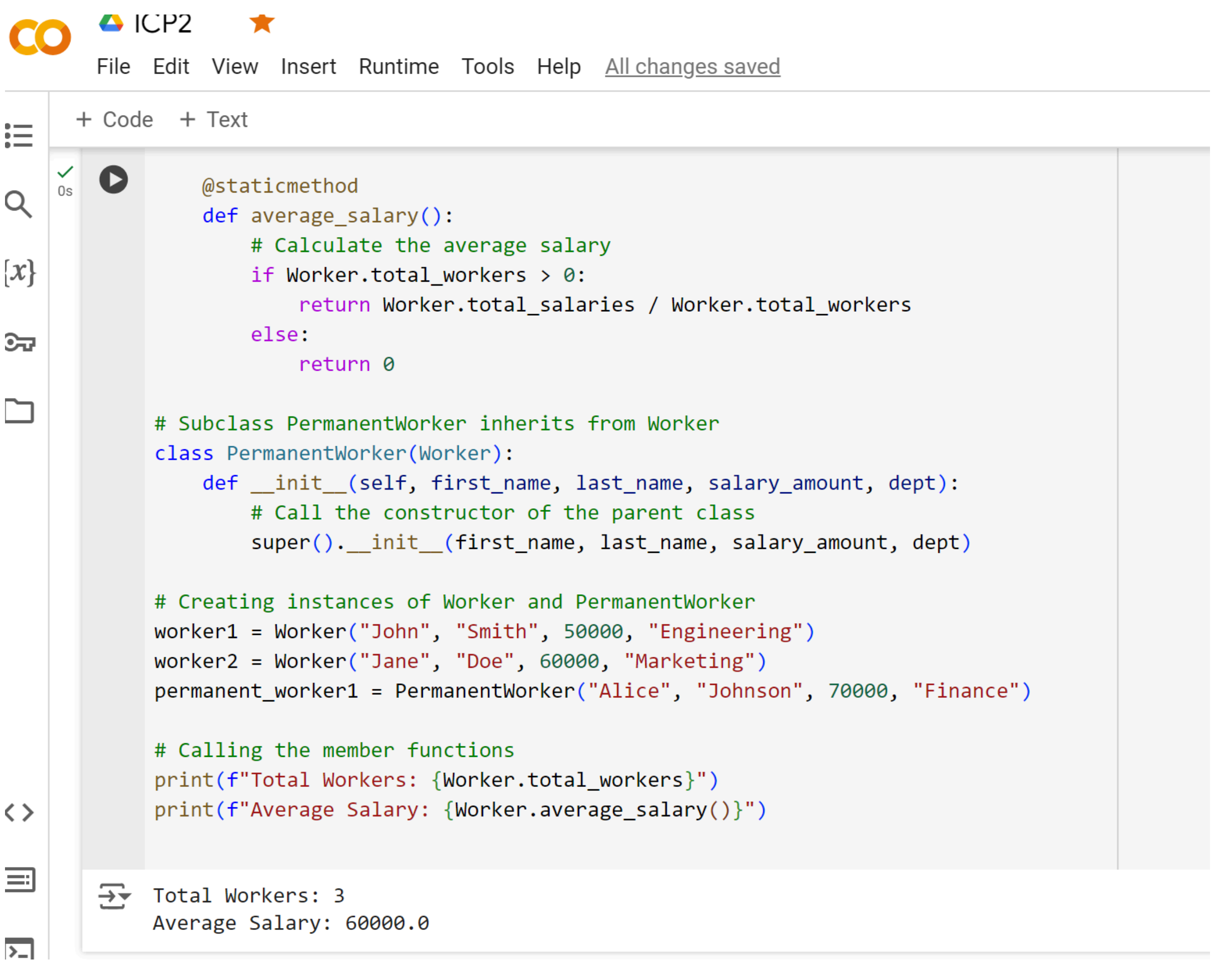
```
[1] def first_word(words):
        # Sort the list alphabetically
        sorted_words = sorted(words)
        # Return the first word in the sorted list
        return sorted_words[0]

    # Example usage
    students = ['Mary', 'Zelda', 'Jimmy', 'Jack', 'Bartholomew', 'Gertrude']
    print(first_word(students))  # Output should be 'Bartholomew'

    Bartholomew
```



```
class Worker:
    # Class variables to keep track of the number of workers and total salary
    total_workers = 0
    total_salaries = 0

    def __init__(self, first_name, last_name, salary_amount, dept):
        # Instance variables
        self.first_name = first_name
        self.last_name = last_name
        self.salary_amount = salary_amount
        self.dept = dept
        # Increment the total worker count and add salary amount whenever a new instance is created
        Worker.total_workers += 1
        Worker.total_salaries += salary_amount

    @staticmethod
    def average_salary():
        # Calculate the average salary
        if Worker.total_workers > 0:
            return Worker.total_salaries / Worker.total_workers
        else:
            return 0

# Subclass PermanentWorker inherits from Worker
class PermanentWorker(Worker):
    def __init__(self, first_name, last_name, salary_amount, dept):
        # Call the constructor of the parent class
```

My github link: https://github.com/w8162583/bda.git

My youtube video link: https://youtu.be/4iluHanYx5A?si=EYL33Hz9rlSW621F



```
    @staticmethod
    def average_salary():
        # Calculate the average salary
        if Worker.total_workers > 0:
            return Worker.total_salaries / Worker.total_workers
        else:
            return 0

# Subclass PermanentWorker inherits from Worker
class PermanentWorker(Worker):
    def __init__(self, first_name, last_name, salary_amount, dept):
        # Call the constructor of the parent class
        super().__init__(first_name, last_name, salary_amount, dept)

# Creating instances of Worker and PermanentWorker
worker1 = Worker("John", "Smith", 50000, "Engineering")
worker2 = Worker("Jane", "Doe", 60000, "Marketing")
permanent_worker1 = PermanentWorker("Alice", "Johnson", 70000, "Finance")

# Calling the member functions
print(f"Total Workers: {Worker.total_workers}")
print(f"Average Salary: {Worker.average_salary()}")
```

```
Total Workers: 3
Average Salary: 60000.0
```

My video link: https://youtu.be/4iluHanYx5A

My github link: