ICP2 report

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

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

[1] ->difference between Counter.count and self._count:- Counter.count is a class variable. The variable is shared by all copies of the Counter class. When it's changed, the update affects every copy of the class. self._count is an instance variable.Each copy of the Counter class has its own version of this variable. Changing it only affects that one copy, not the others.

->Output of a.get_counts() and b.get_counts():- After running the code: a.increment() # a._count becomes 1, Counter.count becomes 1 a.increment() # a._count becomes 2, Counter.count becomes 2 b.increment() # b._count becomes 1, Counter.count becomes 3

Output of a.get_counts():
Instance count: 2, Class count: 3

- self._count for a is 2.
- Counter.count is 3 (shared across instances).

Output of b.get_counts():
Instance count: 1, Class count: 3

- self._count for b is 1.
- Counter.count is still 3 (shared across instances).

->increment method affects both the class and instance variables in the following way:- When a.increment() is called: self._count for the specific instance is incremented up by 1. The shared Counter.count for all instances of Counter also goes up by 1. The same happens for b.increment(), but it only increments self._count for b while still incrementing Counter.count.

```
[3]  #Bug- The function sum_all is written to accept only one argument(args),but when calling sum_all,multiple arguments are getting accessed.
     #Fix- Using *args syntax accepts any number of arguments.
     def sum_all(*args):
         return sum(args)

     print("Sum of 1, 2, 3 is:", sum_all(1, 2, 3))
     print("Sum of 4, 5, 6, 7 is:", sum_all(4, 5, 6, 7))
```

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

```
[9]  def first_word(words):
         # Sorting the list alphabetically to accomplish the result
         sorted_words = sorted(words)
         # Returns the first word from the sorted list
         return sorted_words[0]

     # Example
     students = ['Mary', 'Zelda', 'Jimmy', 'Jack', 'Bartholomew', 'Gertrude']
     print(first_word(students))
```

```
Bartholomew
```

```python
class Employee:
    # Class variable to count the no. of employees
    emp_count = 0
    total_salary = 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
        Employee.emp_count += 1
        Employee.total_salary += salary

    @staticmethod
    def average_salary():
        # Calculate the average salary
        if Employee.emp_count > 0:
            return Employee.total_salary / Employee.emp_count
        else:
            return 0

# Subclass FulltimeEmployee inherits from Employee
class FtEmployee(Employee):
    def __init__(self, name, family, salary, department):
        # Call the constructor of the parent class
        super().__init__(name, family, salary, department)

# Creating instances of Employee and FulltimeEmployee
emp1 = Employee("John", "Smith", 50000, "Engineering")
emp2 = Employee("Jane", "Doe", 60000, "Marketing")
fulltime_emp1 = FtEmployee("Alice", "Johnson", 70000, "Finance")

# Calling the member functions
print(f"Total Employees: {Employee.emp_count}")
print(f"Average Salary: {Employee.average_salary()}")
```

```
Total Employees: 3
Average Salary: 60000.0
```

Github Repository link:-https://github.com/niharika0912/BDA.git