Ian Pope 700717419
DSA 5620 ICP 2

Ian Pope 700717419 ICP 2

ICP 2 Question 1:

The difference between Counter.count ans self.count is that Counter.count belongs to the class, where self._count belongs to the instance. This means that Counter.count can be changed and accessed by all instances of Counter as opposed to just a single instance for self._count.

The output is:

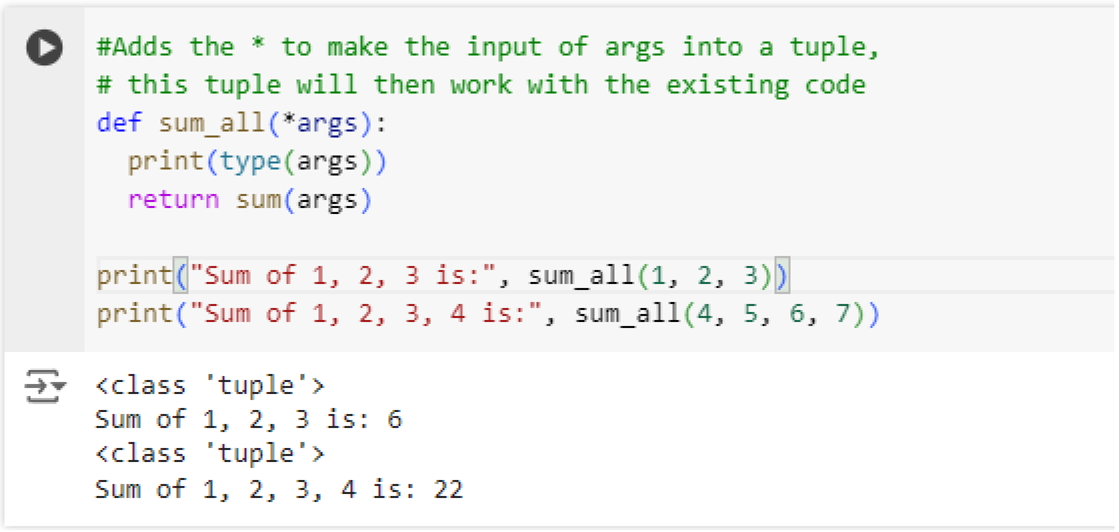
Instance count: 2, Class count: 3

Instance count: 1, Class count: 3

The increment method will raise both the class variable count and the instance variable _count by 1. The change for the class variable will be changed for all instances of Counter.

ICP 2 Question 2:

Fix bug in the program

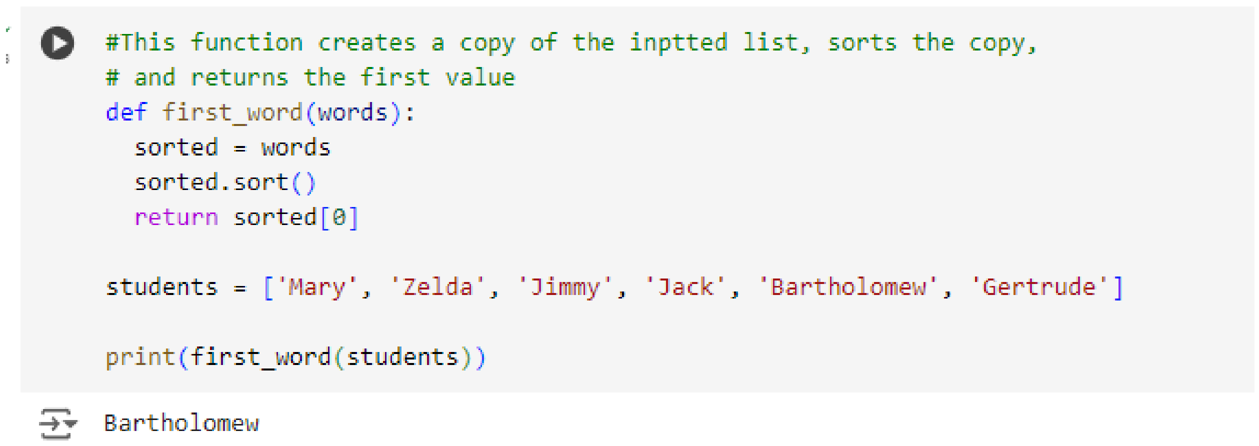


```
#Adds the * to make the input of args into a tuple,
# this tuple will then work with the existing code
def sum_all(*args):
  print(type(args))
  return sum(args)

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

```
<class 'tuple'>
Sum of 1, 2, 3 is: 6
<class 'tuple'>
Sum of 1, 2, 3, 4 is: 22
```

ICP 2 Question 3:

Creates a function that sorts a list of strings and returns the first word by alphabetical order.

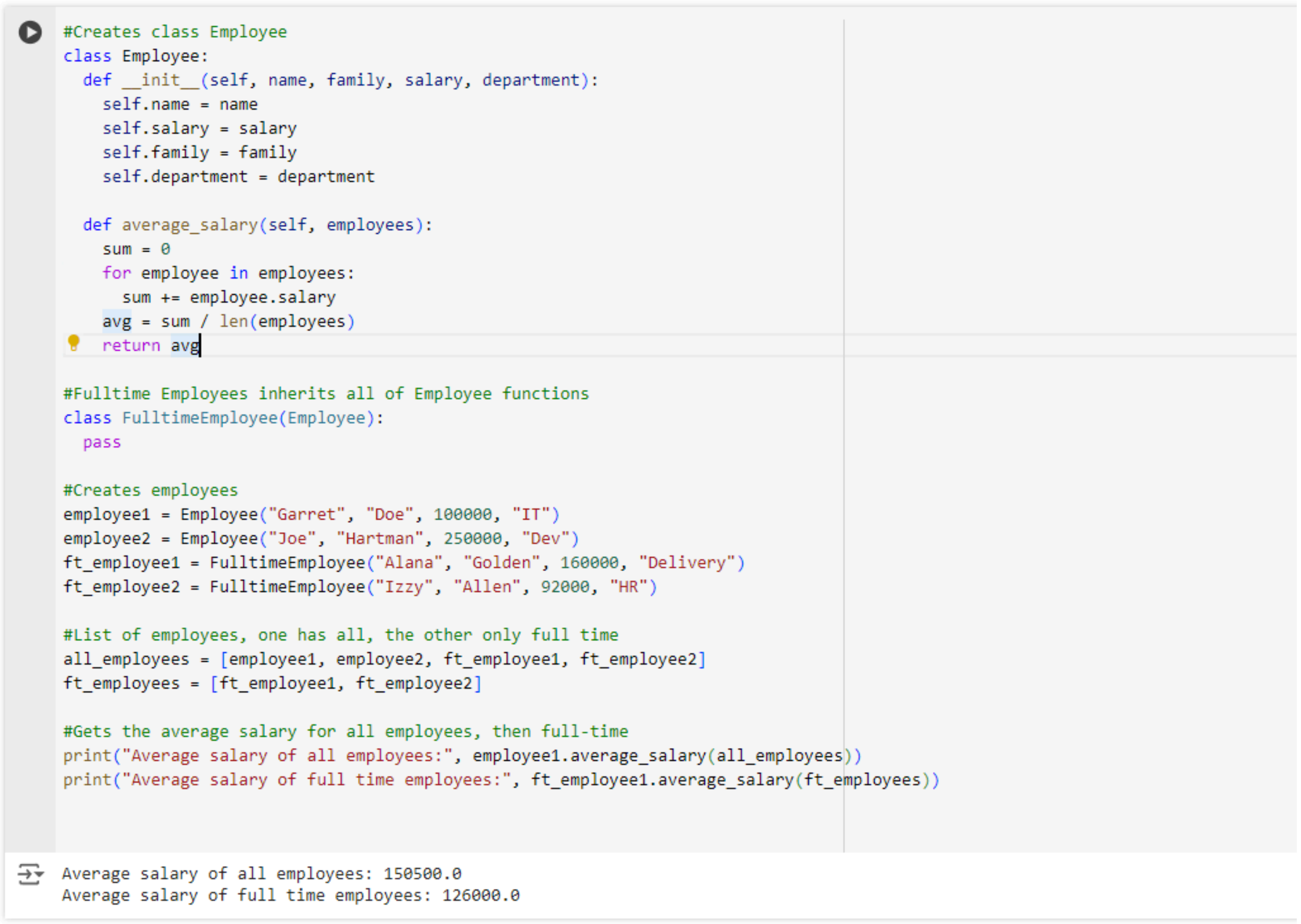


```python
#This function creates a copy of the inptted list, sorts the copy,
# and returns the first value
def first_word(words):
  sorted = words
  sorted.sort()
  return sorted[0]

students = ['Mary', 'Zelda', 'Jimmy', 'Jack', 'Bartholomew', 'Gertrude']

print(first_word(students))
```

```
Bartholomew
```

ICP 2 Question 4:

Create Employee class and functions, create a subclass FulltimeEmployee, and use the functions of the class on instances of the classes.



```python
#Creates class Employee
class Employee:
  def __init__(self, name, family, salary, department):
    self.name = name
    self.salary = salary
    self.family = family
    self.department = department

  def average_salary(self, employees):
    sum = 0
    for employee in employees:
      sum += employee.salary
    avg = sum / len(employees)
    return avg

#Fulltime Employees inherits all of Employee functions
class FulltimeEmployee(Employee):
  pass

#Creates employees
employee1 = Employee("Garret", "Doe", 100000, "IT")
employee2 = Employee("Joe", "Hartman", 250000, "Dev")
ft_employee1 = FulltimeEmployee("Alana", "Golden", 160000, "Delivery")
ft_employee2 = FulltimeEmployee("Izzy", "Allen", 92000, "HR")

#List of employees, one has all, the other only full time
all_employees = [employee1, employee2, ft_employee1, ft_employee2]
ft_employees = [ft_employee1, ft_employee2]

#Gets the average salary for all employees, then full-time
print("Average salary of all employees:", employee1.average_salary(all_employees))
print("Average salary of full time employees:", ft_employee1.average_salary(ft_employees))
```

```
Average salary of all employees: 150500.0
Average salary of full time employees: 126000.0
```