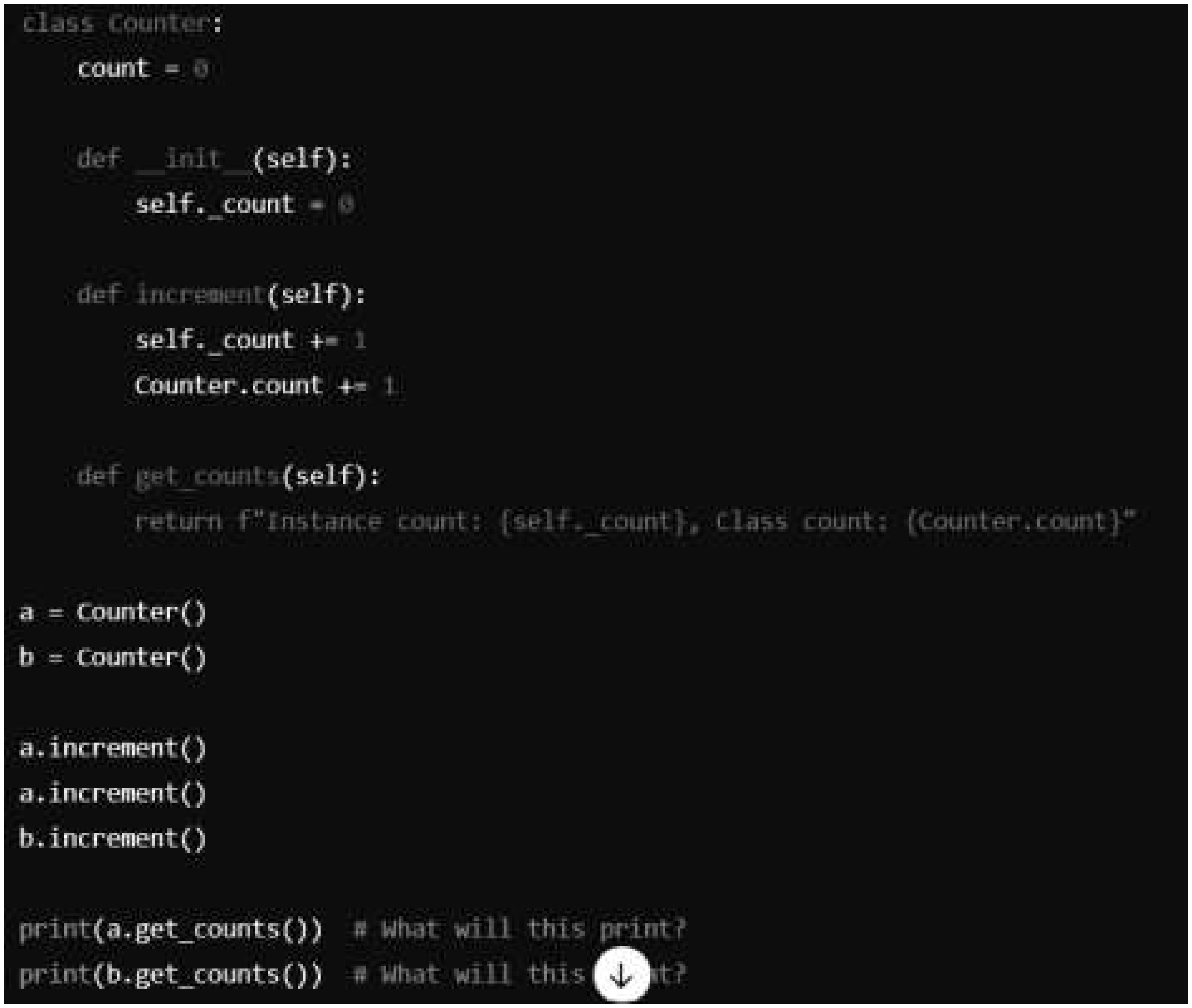
Big data Analytics: ICP2

In class programming:

1. **Question: Consider the following Python code:**



```python
class Counter:
    count = 0

    def __init__(self):
        self._count = 0

    def increment(self):
        self._count += 1
        Counter.count += 1

    def get_counts(self):
        return f"Instance count: {self._count}, Class count: {Counter.count}"

a = Counter()
b = Counter()

a.increment()
a.increment()
b.increment()

print(a.get_counts())  # What will this print?
print(b.get_counts())  # What will this   t?
```

Tasks:

• Explain the difference between Counter.count and self._count.

The main difference between Counter.count and self.count is that Counter.count will count the total number of times anything is incremented, as it is declared in the class and not in the function. But self._count only counts each specific instance, like a or b.

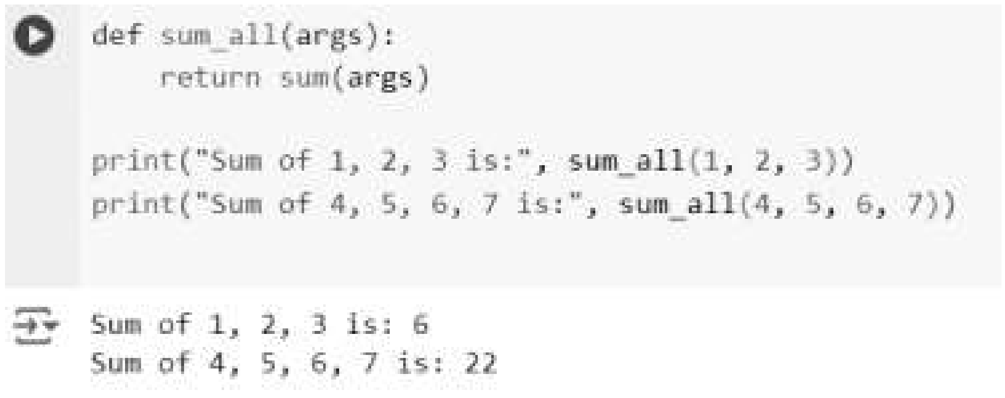
• What is the output of a.get_counts() and b.get_counts()?
a.get_counts() = instance count: 2, Class count: 3

b.get_counts() = instance count: 1, Class count: 3

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

It affects both because Counter.count is declared in the class, so it updates that value, whereas self._count is declared in init, so it only increments self

2. Find and remove the bug from the code to obtain the given output.



```
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
```

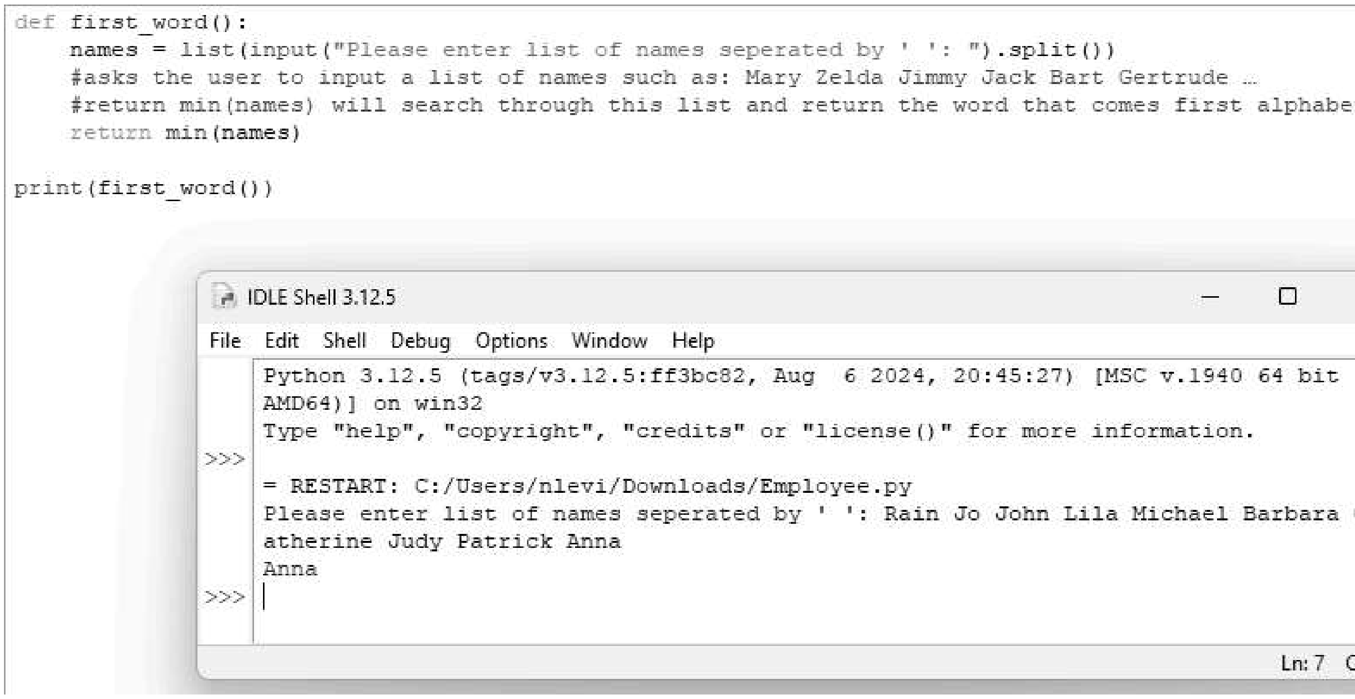
The bug with this code is that at first sum_all is only expecting one argument, if you change args to *args it will fix the problem and will work as intended.

3. Write a function called first_word that takes a list of character strings as input and returns the first element of the list in alphabetical order. For example, your function should work like this:

students = ['Mary', 'Zelda', 'Jimmy', 'Jack', 'Bartholomew', 'Gertrude'] (***Input)***

first_word(students) (***Function)***

'Bartholomew' (***Output)***



```
def first_word():
    names = list(input("Please enter list of names seperated by ' ': ").split())
    #asks the user to input a list of names such as: Mary Zelda Jimmy Jack Bart Gertrude ...
    #return min(names) will search through this list and return the word that comes first alphabe
    return min(names)

print(first_word())
```

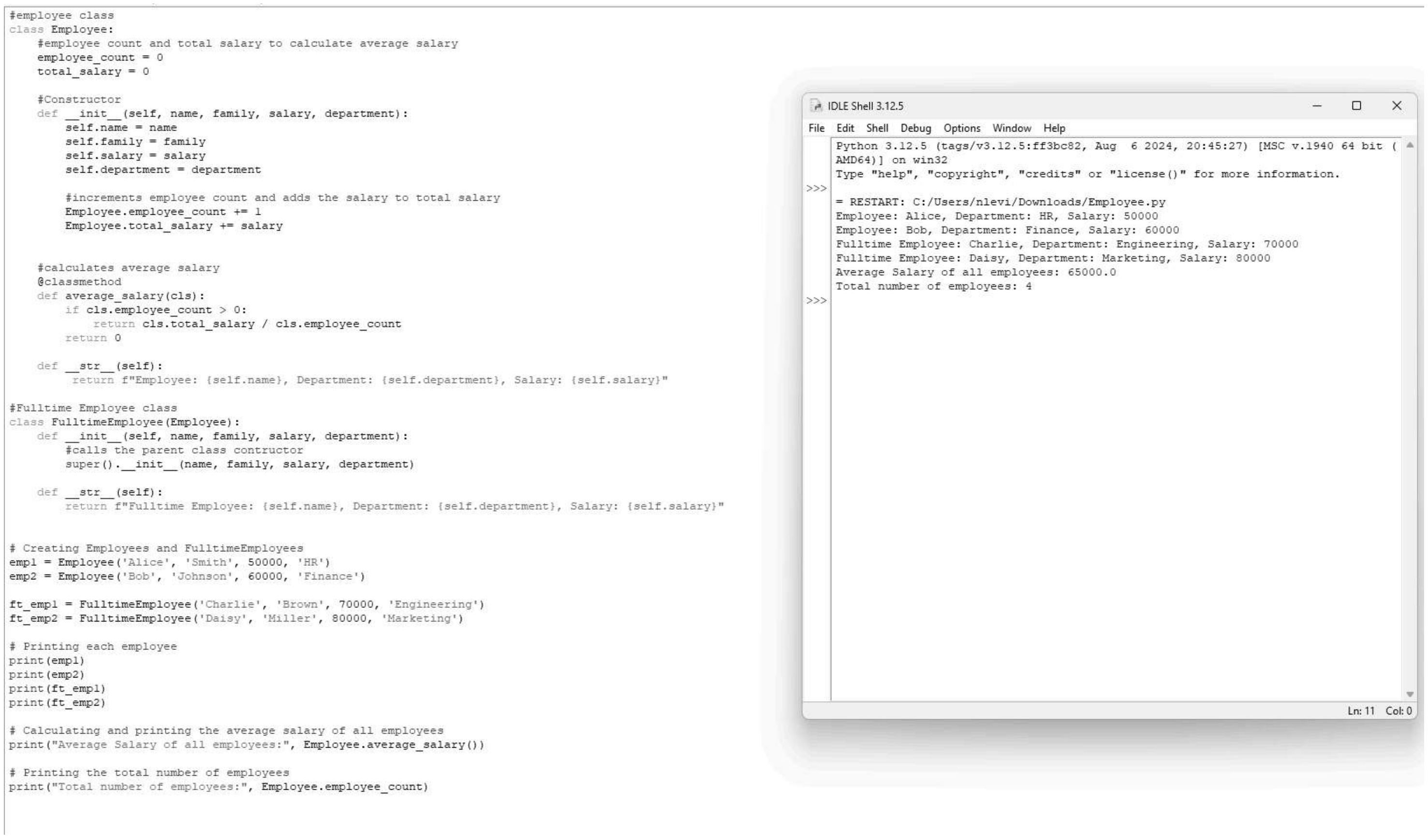
```
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug  6 2024, 20:45:27) [MSC v.1940 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/nlevi/Downloads/Employee.py
Please enter list of names seperated by ' ': Rain Jo John Lila Michael Barbara
atherine Judy Patrick Anna
Anna
>>> 
```

This function ^^^ will output the element that comes first alphabetically.

Hint: **You'll need to first sort your list in the function to accomplish this, then identify the first element. Within a function, it is a good idea to use multiple lines of code to separate out the different steps. Just make sure all the code that belongs to the function is indented!**

4. 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.



```
#employee class
class Employee:
    #employee count and total salary to calculate average salary
    employee_count = 0
    total_salary = 0

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

        #increments employee count and adds the salary to total salary
        Employee.employee_count += 1
        Employee.total_salary += salary


    #calculates average salary
    @classmethod
    def average_salary(cls):
        if cls.employee_count > 0:
            return cls.total_salary / cls.employee_count
        return 0

    def __str__(self):
         return f"Employee: {self.name}, Department: {self.department}, Salary: {self.salary}"

#Fulltime Employee class
class FulltimeEmployee(Employee):
    def __init__(self, name, family, salary, department):
        #calls the parent class contructor
        super().__init__(name, family, salary, department)

    def __str__(self):
        return f"Fulltime Employee: {self.name}, Department: {self.department}, Salary: {self.salary}"


# Creating Employees and FulltimeEmployees
emp1 = Employee('Alice', 'Smith', 50000, 'HR')
emp2 = Employee('Bob', 'Johnson', 60000, 'Finance')

ft_emp1 = FulltimeEmployee('Charlie', 'Brown', 70000, 'Engineering')
ft_emp2 = FulltimeEmployee('Daisy', 'Miller', 80000, 'Marketing')

# Printing each employee
print(emp1)
print(emp2)
print(ft_emp1)
print(ft_emp2)

# Calculating and printing the average salary of all employees
print("Average Salary of all employees:", Employee.average_salary())

# Printing the total number of employees
print("Total number of employees:", Employee.employee_count)
```

```
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug  6 2024, 20:45:27) [MSC v.1940 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/nlevi/Downloads/Employee.py
Employee: Alice, Department: HR, Salary: 50000
Employee: Bob, Department: Finance, Salary: 60000
Fulltime Employee: Charlie, Department: Engineering, Salary: 70000
Fulltime Employee: Daisy, Department: Marketing, Salary: 80000
Average Salary of all employees: 65000.0
Total number of employees: 4
>>>
```

Follow the submission guidelines used for previous ICP.

Evaluation Criteria:
1. Completeness of Features
2. Code Quality (https://en.wikipedia.org/wiki/Best_coding_practices)
3. Time

Note: *Cheating, plagiarism, disruptive behavior and other forms of unacceptable conduct are subject to strong sanctions in accordance with university policy.*

Video Link: https://youtu.be/mvFvZ-FxrGQ