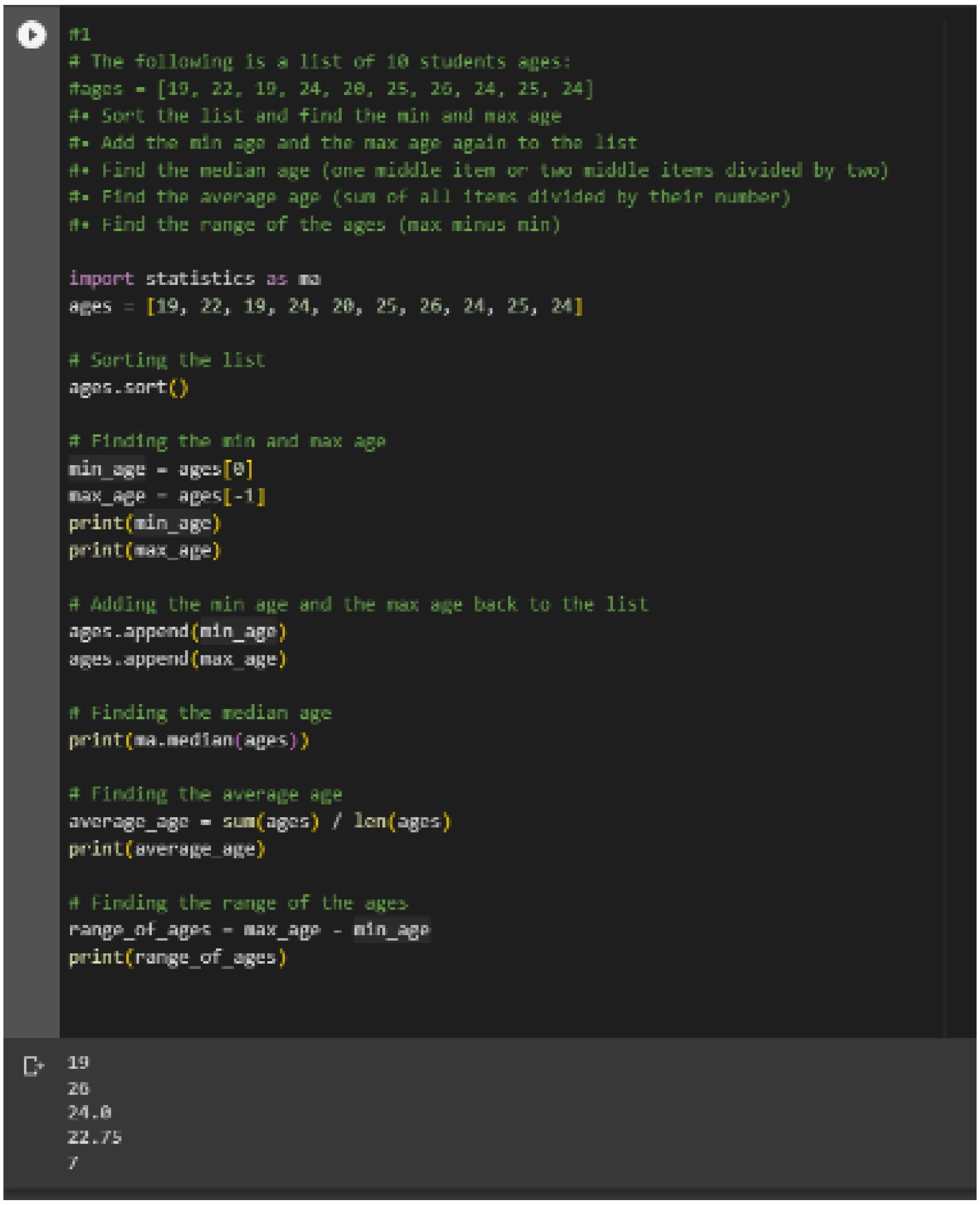
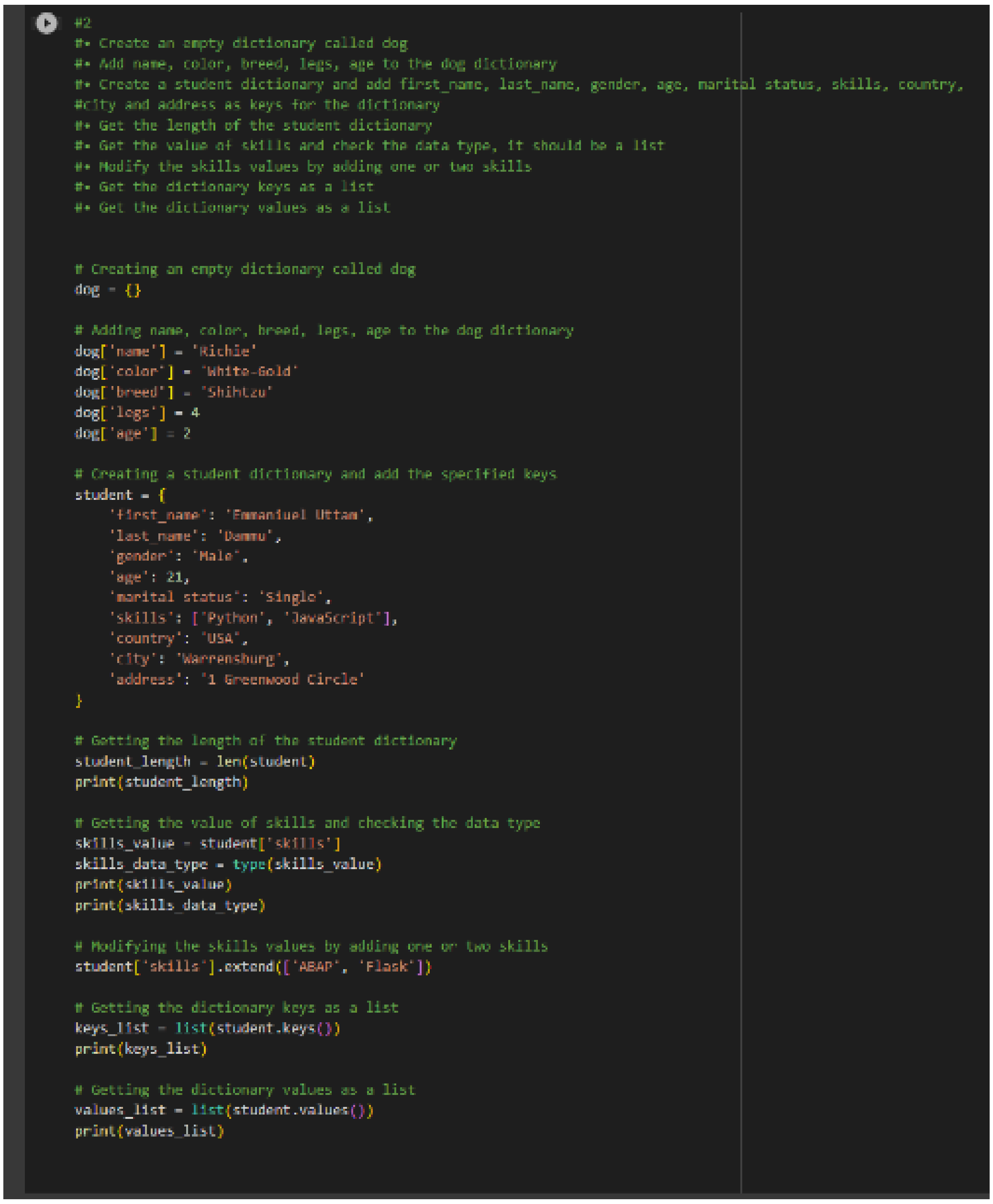
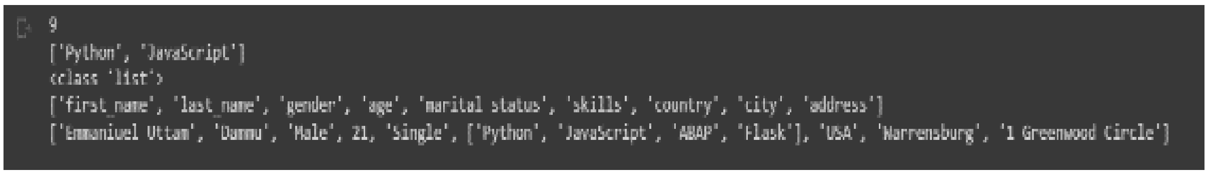


```python
#1
# The following is a list of 10 students ages:
#ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
#• Sort the list and find the min and max age
#• Add the min age and the max age again to the list
#• Find the median age (one middle item or two middle items divided by two)
#• Find the average age (sum of all items divided by their number)
#• Find the range of the ages (max minus min)

import statistics as ma
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]

# Sorting the list
ages.sort()

# Finding the min and max age
min_age = ages[0]
max_age = ages[-1]
print(min_age)
print(max_age)

# Adding the min age and the max age back to the list
ages.append(min_age)
ages.append(max_age)

# Finding the median age
print(ma.median(ages))

# Finding the average age
average_age = sum(ages) / len(ages)
print(average_age)

# Finding the range of the ages
range_of_ages = max_age - min_age
print(range_of_ages)
```

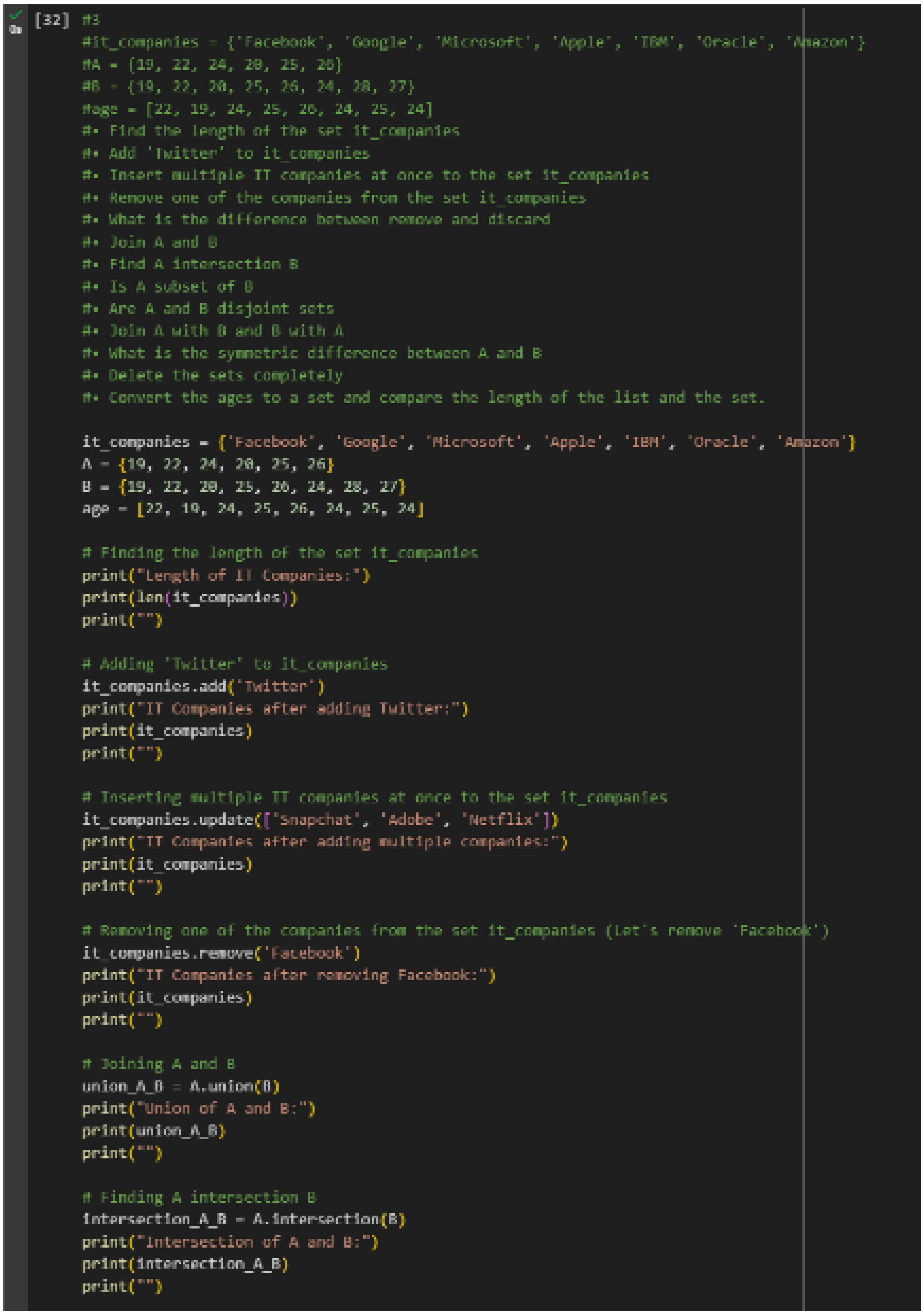
```
19
26
24.0
22.75
7
```



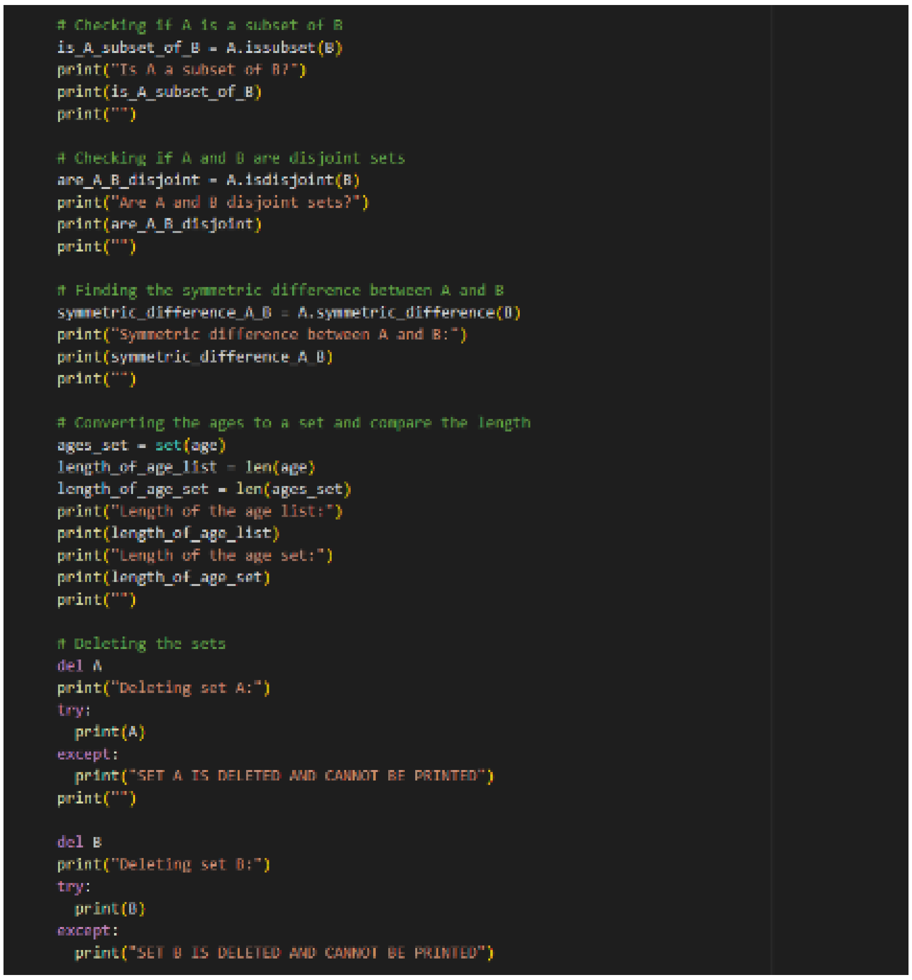
```python
#2
#• Create an empty dictionary called dog
#• Add name, color, breed, legs, age to the dog dictionary
#• Create a student dictionary and add first_name, last_name, gender, age, marital status, skills, country,
#city and address as keys for the dictionary
#• Get the length of the student dictionary
#• Get the value of skills and check the data type, it should be a list
#• Modify the skills values by adding one or two skills
#• Get the dictionary keys as a list
#• Get the dictionary values as a list


# Creating an empty dictionary called dog
dog = {}

# Adding name, color, breed, legs, age to the dog dictionary
dog['name'] = 'Richie'
dog['color'] = 'White-Gold'
dog['breed'] = 'ShihTzu'
dog['legs'] = 4
dog['age'] = 2

# Creating a student dictionary and add the specified keys
student = {
    'first_name': 'Emmaniuel Uttam',
    'last_name': 'Dammu',
    'gender': 'Male',
    'age': 21,
    'marital status': 'Single',
    'skills': ['Python', 'JavaScript'],
    'country': 'USA',
    'city': 'Warrensburg',
    'address': '1 Greenwood Circle'
}

# Getting the length of the student dictionary
student_length = len(student)
print(student_length)

# Getting the value of skills and checking the data type
skills_value = student['skills']
skills_data_type = type(skills_value)
print(skills_value)
print(skills_data_type)

# Modifying the skills values by adding one or two skills
student['skills'].extend(['ABAP', 'Flask'])

# Getting the dictionary keys as a list
keys_list = list(student.keys())
print(keys_list)

# Getting the dictionary values as a list
values_list = list(student.values())
print(values_list)
```



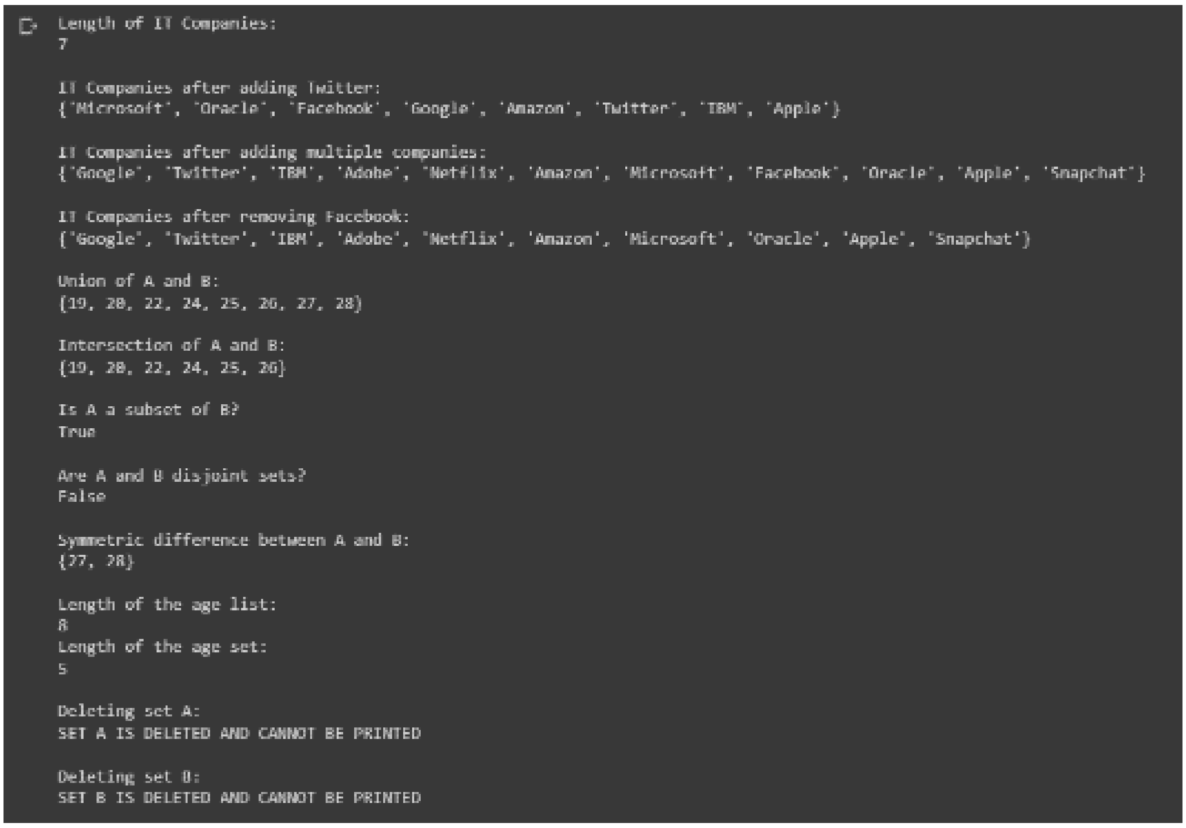
```
9
['Python', 'JavaScript']
<class 'list'>
['first_name', 'last_name', 'gender', 'age', 'marital status', 'skills', 'country', 'city', 'address']
['Emmaniuel Uttam', 'Dammu', 'Male', 21, 'Single', ['Python', 'JavaScript', 'ABAP', 'Flask'], 'USA', 'Warrensburg', '1 Greenwood Circle']
```



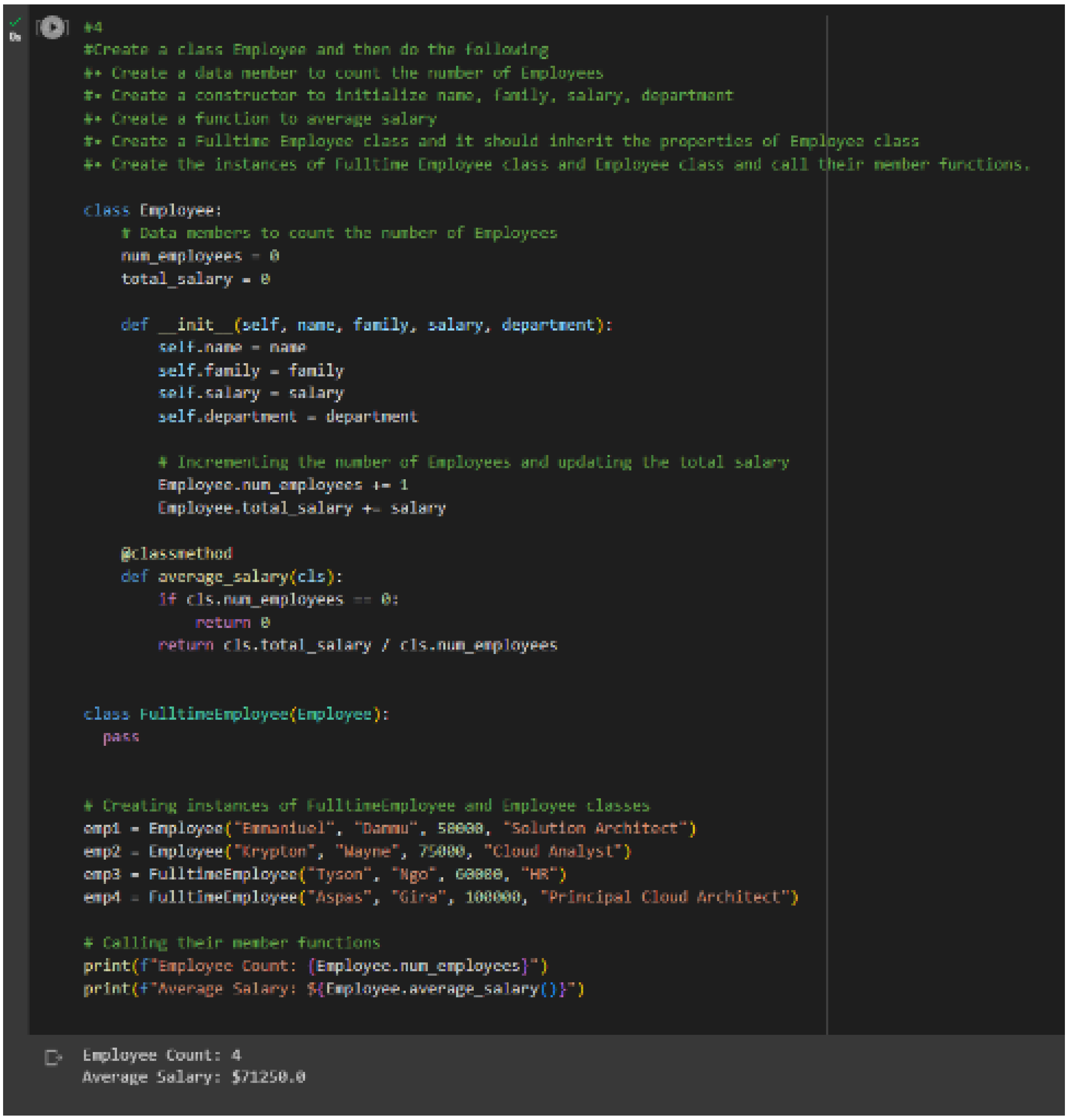
```
[32] #3
     #it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
     #A = {19, 22, 24, 20, 25, 26}
     #B = {19, 22, 20, 25, 26, 24, 28, 27}
     #age = [22, 19, 24, 25, 26, 24, 25, 24]
     #• Find the length of the set it_companies
     #• Add 'Twitter' to it_companies
     #• Insert multiple IT companies at once to the set it_companies
     #• Remove one of the companies from the set it_companies
     #• What is the difference between remove and discard
     #• Join A and B
     #• Find A intersection B
     #• Is A subset of B
     #• Are A and B disjoint sets
     #• Join A with B and B with A
     #• What is the symmetric difference between A and B
     #• Delete the sets completely
     #• Convert the ages to a set and compare the length of the list and the set.

     it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
     A = {19, 22, 24, 20, 25, 26}
     B = {19, 22, 20, 25, 26, 24, 28, 27}
     age = [22, 19, 24, 25, 26, 24, 25, 24]

     # Finding the length of the set it_companies
     print("Length of IT Companies:")
     print(len(it_companies))
     print("")

     # Adding 'Twitter' to it_companies
     it_companies.add('Twitter')
     print("IT Companies after adding Twitter:")
     print(it_companies)
     print("")

     # Inserting multiple IT companies at once to the set it_companies
     it_companies.update(['Snapchat', 'Adobe', 'Netflix'])
     print("IT Companies after adding multiple companies:")
     print(it_companies)
     print("")

     # Removing one of the companies from the set it_companies (Let's remove 'Facebook')
     it_companies.remove('Facebook')
     print("IT Companies after removing Facebook:")
     print(it_companies)
     print("")

     # Joining A and B
     union_A_B = A.union(B)
     print("Union of A and B:")
     print(union_A_B)
     print("")

     # Finding A intersection B
     intersection_A_B = A.intersection(B)
     print("Intersection of A and B:")
     print(intersection_A_B)
     print("")
```



```python
# Checking if A is a subset of B
is_A_subset_of_B = A.issubset(B)
print("Is A a subset of B?")
print(is_A_subset_of_B)
print("")

# Checking if A and B are disjoint sets
are_A_B_disjoint = A.isdisjoint(B)
print("Are A and B disjoint sets?")
print(are_A_B_disjoint)
print("")

# Finding the symmetric difference between A and B
symmetric_difference_A_B = A.symmetric_difference(B)
print("Symmetric difference between A and B:")
print(symmetric_difference_A_B)
print("")

# Converting the ages to a set and compare the length
ages_set = set(age)
length_of_age_list = len(age)
length_of_age_set = len(ages_set)
print("Length of the age list:")
print(length_of_age_list)
print("Length of the age set:")
print(length_of_age_set)
print("")

# Deleting the sets
del A
print("Deleting set A:")
try:
  print(A)
except:
  print("SET A IS DELETED AND CANNOT BE PRINTED")
print("")

del B
print("Deleting set B:")
try:
  print(B)
except:
  print("SET B IS DELETED AND CANNOT BE PRINTED")
```



```
Length of IT Companies:
7

IT Companies after adding Twitter:
{'Microsoft', 'Oracle', 'Facebook', 'Google', 'Amazon', 'Twitter', 'IBM', 'Apple'}

IT Companies after adding multiple companies:
{'Google', 'Twitter', 'IBM', 'Adobe', 'Netflix', 'Amazon', 'Microsoft', 'Facebook', 'Oracle', 'Apple', 'Snapchat'}

IT Companies after removing Facebook:
{'Google', 'Twitter', 'IBM', 'Adobe', 'Netflix', 'Amazon', 'Microsoft', 'Oracle', 'Apple', 'Snapchat'}

Union of A and B:
{19, 20, 22, 24, 25, 26, 27, 28}

Intersection of A and B:
{19, 20, 22, 24, 25, 26}

Is A a subset of B?
True

Are A and B disjoint sets?
False

Symmetric difference between A and B:
{27, 28}

Length of the age list:
8
Length of the age set:
5

Deleting set A:
SET A IS DELETED AND CANNOT BE PRINTED

Deleting set B:
SET B IS DELETED AND CANNOT BE PRINTED
```



```python
#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.

class Employee:
    # Data members to count the number of Employees
    num_employees = 0
    total_salary = 0

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

        # Incrementing the number of Employees and updating the total salary
        Employee.num_employees += 1
        Employee.total_salary += salary

    @classmethod
    def average_salary(cls):
        if cls.num_employees == 0:
            return 0
        return cls.total_salary / cls.num_employees


class FulltimeEmployee(Employee):
  pass


# Creating instances of FulltimeEmployee and Employee classes
emp1 = Employee("Emmaniuel", "Dammu", 58000, "Solution Architect")
emp2 = Employee("Krypton", "Wayne", 75000, "Cloud Analyst")
emp3 = FulltimeEmployee("Tyson", "Ngo", 60000, "HR")
emp4 = FulltimeEmployee("Aspas", "Gira", 100000, "Principal Cloud Architect")

# Calling their member functions
print(f"Employee Count: {Employee.num_employees}")
print(f"Average Salary: ${Employee.average_salary()}")
```

```
Employee Count: 4
Average Salary: $71250.0
```

Github Repo Link: https://github.com/Krypton0626/Bigdata/tree/main/ICP%202

YouTube video Link: https://youtu.be/7uryavt9ATU