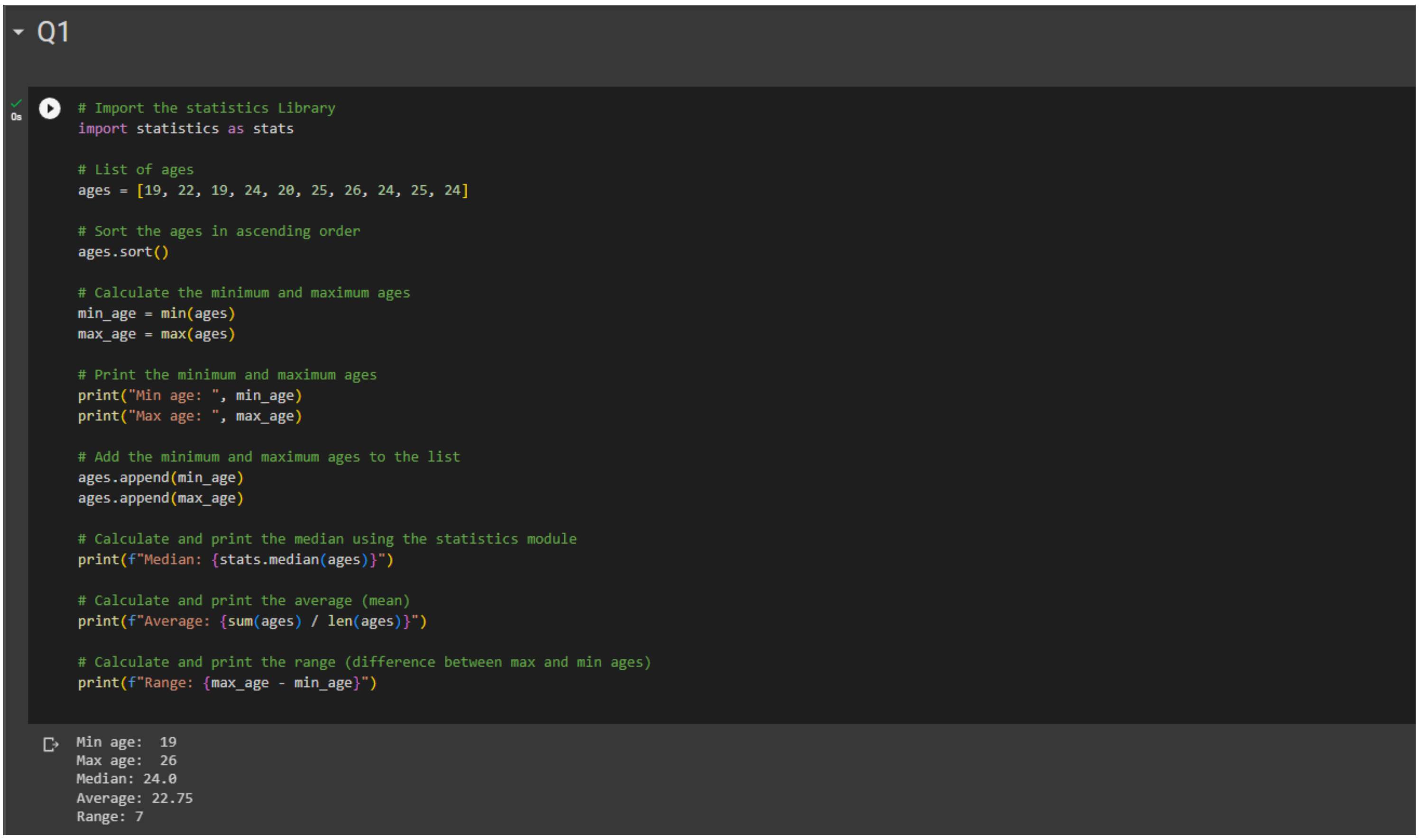
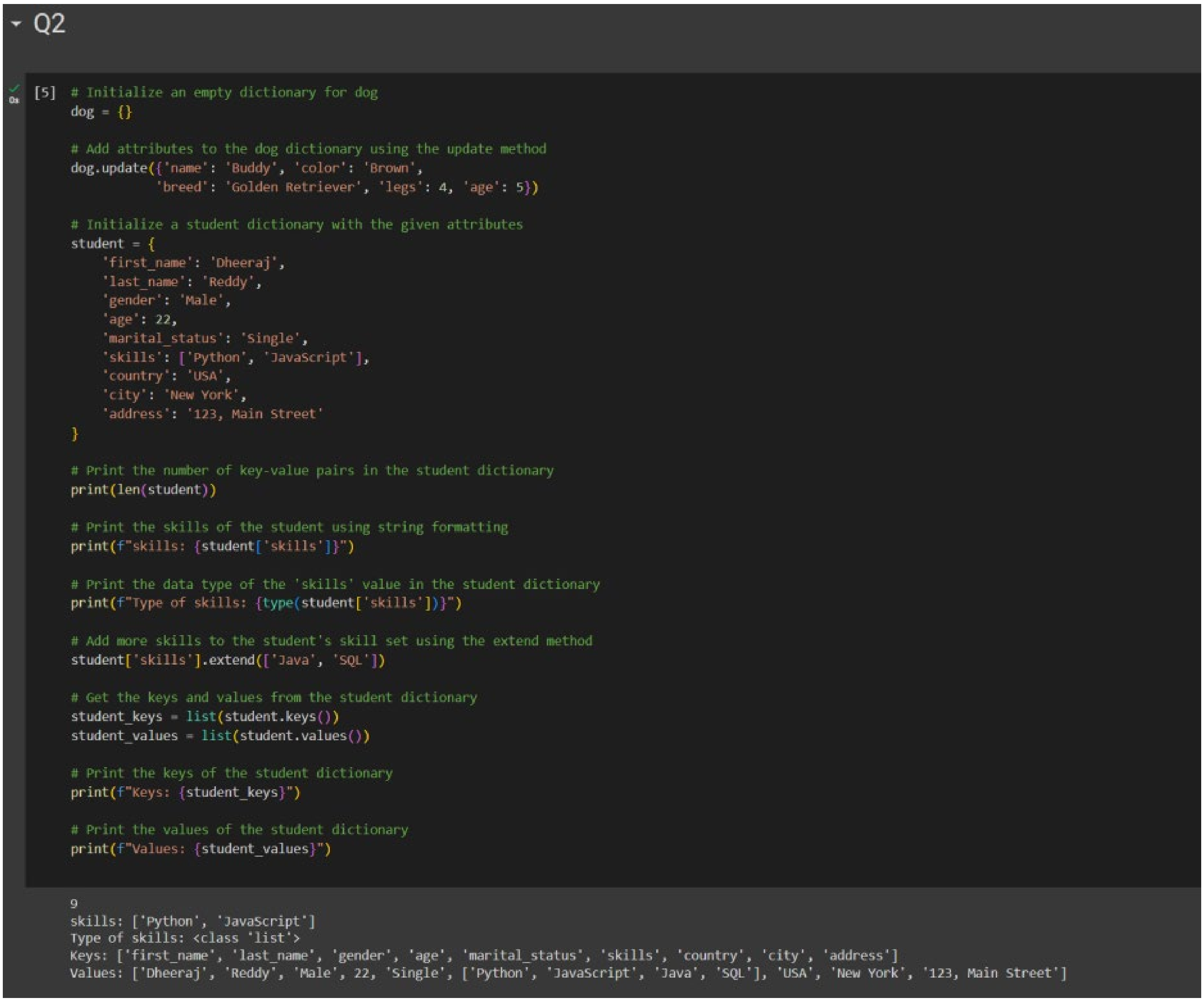


Q1

```python
# Import the statistics Library
import statistics as stats

# List of ages
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]

# Sort the ages in ascending order
ages.sort()

# Calculate the minimum and maximum ages
min_age = min(ages)
max_age = max(ages)

# Print the minimum and maximum ages
print("Min age: ", min_age)
print("Max age: ", max_age)

# Add the minimum and maximum ages to the list
ages.append(min_age)
ages.append(max_age)

# Calculate and print the median using the statistics module
print(f"Median: {stats.median(ages)}")

# Calculate and print the average (mean)
print(f"Average: {sum(ages) / len(ages)}")

# Calculate and print the range (difference between max and min ages)
print(f"Range: {max_age - min_age}")
```

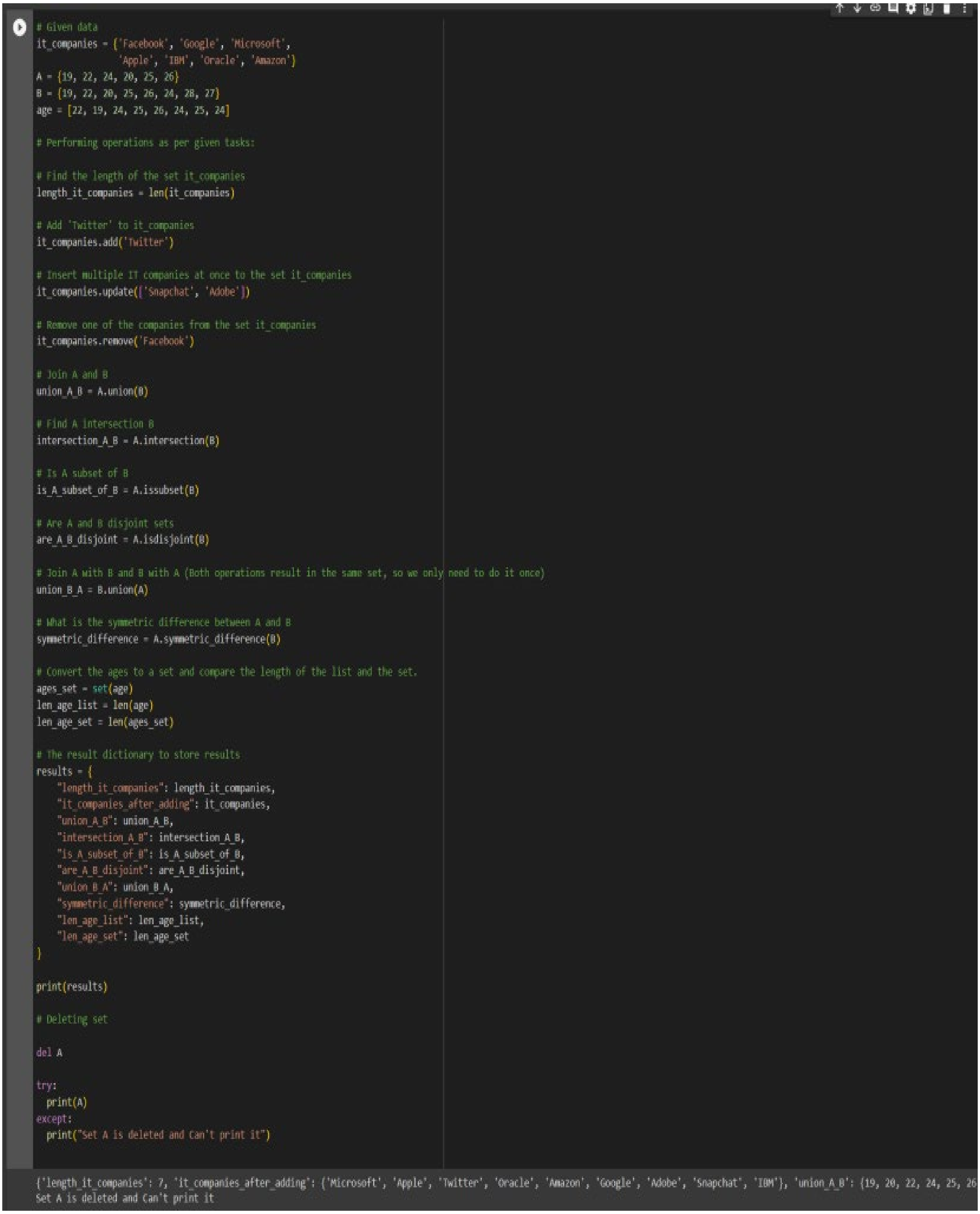
```
Min age:  19
Max age:  26
Median: 24.0
Average: 22.75
Range: 7
```



Q2

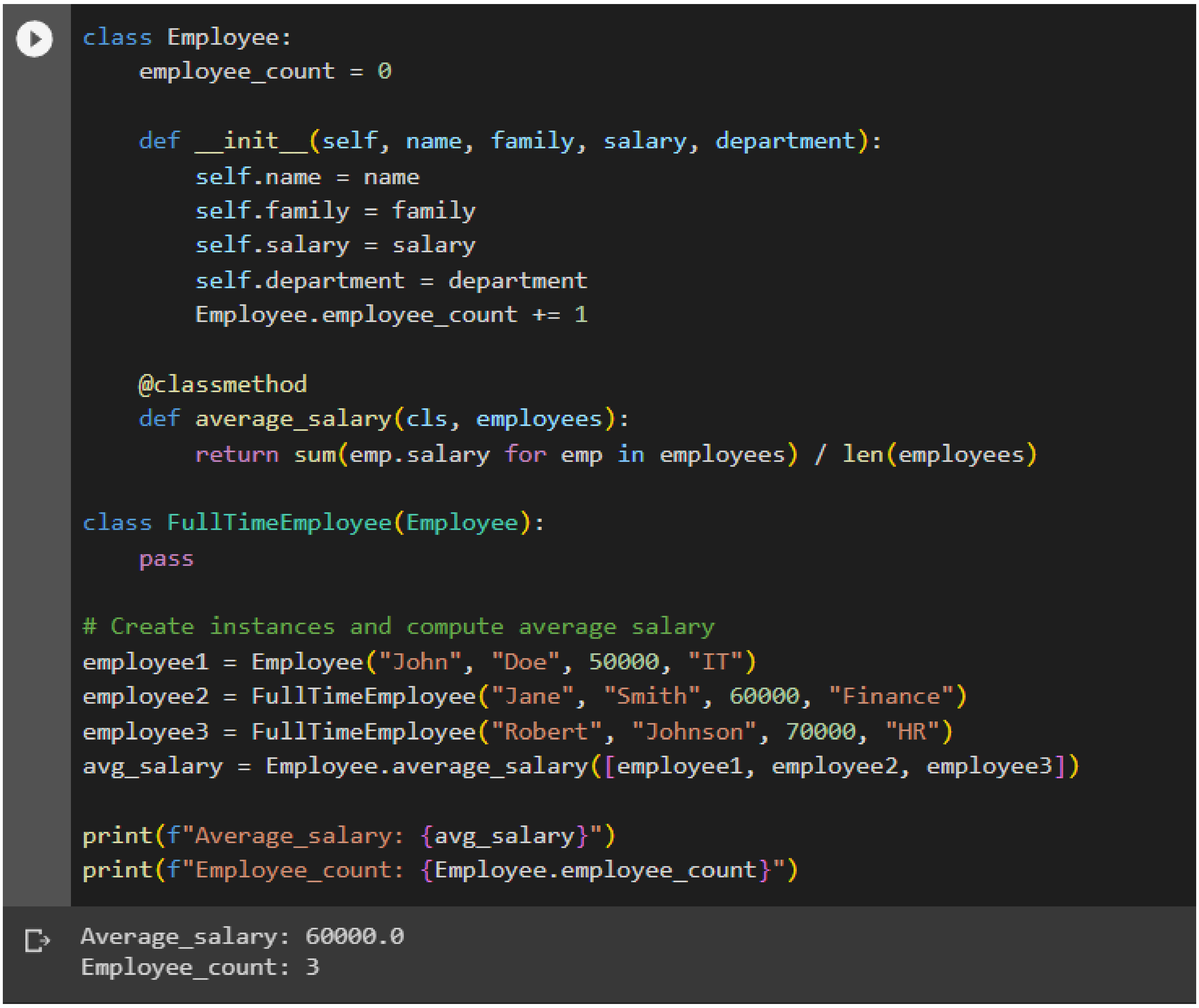
```python
# Initialize an empty dictionary for dog
dog = {}

# Add attributes to the dog dictionary using the update method
dog.update({'name': 'Buddy', 'color': 'Brown',
            'breed': 'Golden Retriever', 'legs': 4, 'age': 5})

# Initialize a student dictionary with the given attributes
student = {
    'first_name': 'Dheeraj',
    'last_name': 'Reddy',
    'gender': 'Male',
    'age': 22,
    'marital_status': 'Single',
    'skills': ['Python', 'JavaScript'],
    'country': 'USA',
    'city': 'New York',
    'address': '123, Main Street'
}

# Print the number of key-value pairs in the student dictionary
print(len(student))

# Print the skills of the student using string formatting
print(f"skills: {student['skills']}")

# Print the data type of the 'skills' value in the student dictionary
print(f"Type of skills: {type(student['skills'])}")

# Add more skills to the student's skill set using the extend method
student['skills'].extend(['Java', 'SQL'])

# Get the keys and values from the student dictionary
student_keys = list(student.keys())
student_values = list(student.values())

# Print the keys of the student dictionary
print(f"Keys: {student_keys}")

# Print the values of the student dictionary
print(f"Values: {student_values}")
```

```
9
skills: ['Python', 'JavaScript']
Type of skills: <class 'list'>
Keys: ['first_name', 'last_name', 'gender', 'age', 'marital_status', 'skills', 'country', 'city', 'address']
Values: ['Dheeraj', 'Reddy', 'Male', 22, 'Single', ['Python', 'JavaScript', 'Java', 'SQL'], 'USA', 'New York', '123, Main Street']
```



```python
# Given data
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]

# Performing operations as per given tasks:

# Find the length of the set it_companies
length_it_companies = len(it_companies)

# Add 'Twitter' to it_companies
it_companies.add('Twitter')

# Insert multiple IT companies at once to the set it_companies
it_companies.update(['Snapchat', 'Adobe'])

# Remove one of the companies from the set it_companies
it_companies.remove('Facebook')

# Join A and B
union_A_B = A.union(B)

# Find A intersection B
intersection_A_B = A.intersection(B)

# Is A subset of B
is_A_subset_of_B = A.issubset(B)

# Are A and B disjoint sets
are_A_B_disjoint = A.isdisjoint(B)

# Join A with B and B with A (Both operations result in the same set, so we only need to do it once)
union_B_A = B.union(A)

# What is the symmetric difference between A and B
symmetric_difference = A.symmetric_difference(B)

# Convert the ages to a set and compare the length of the list and the set.
ages_set = set(age)
len_age_list = len(age)
len_age_set = len(ages_set)

# The result dictionary to store results
results = {
    "length_it_companies": length_it_companies,
    "it_companies_after_adding": it_companies,
    "union_A_B": union_A_B,
    "intersection_A_B": intersection_A_B,
    "is_A_subset_of_B": is_A_subset_of_B,
    "are_A_B_disjoint": are_A_B_disjoint,
    "union_B_A": union_B_A,
    "symmetric_difference": symmetric_difference,
    "len_age_list": len_age_list,
    "len_age_set": len_age_set
}

print(results)

# Deleting set

del A

try:
  print(A)
except:
  print("Set A is deleted and Can't print it")
```

```
{'length_it_companies': 7, 'it_companies_after_adding': {'Microsoft', 'Apple', 'Twitter', 'Oracle', 'Amazon', 'Google', 'Adobe', 'Snapchat', 'IBM'}, 'union_A_B': {19, 20, 22, 24, 25, 26
Set A is deleted and Can't print it
```



```python
class Employee:
    employee_count = 0

    def __init__(self, name, family, salary, department):
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.employee_count += 1

    @classmethod
    def average_salary(cls, employees):
        return sum(emp.salary for emp in employees) / len(employees)

class FullTimeEmployee(Employee):
    pass

# Create instances and compute average salary
employee1 = Employee("John", "Doe", 50000, "IT")
employee2 = FullTimeEmployee("Jane", "Smith", 60000, "Finance")
employee3 = FullTimeEmployee("Robert", "Johnson", 70000, "HR")
avg_salary = Employee.average_salary([employee1, employee2, employee3])

print(f"Average_salary: {avg_salary}")
print(f"Employee_count: {Employee.employee_count}")
```

```
Average_salary: 60000.0
Employee_count: 3
```

Github Repo: https://github.com/dheerukarra/BigDataAnalytics/tree/main/ICP%202

Video Link: https://youtu.be/jfGT4SKkSLU