Q1

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

# Sort the list
ages.sort()
print("Sorted Ages:", ages)

# Min and Max age
min_age = ages[0]
max_age = ages[-1]
print("Min Age:", min_age)
print("Max Age:", max_age)

# Add the min age and the max age again to the list
ages.append(min_age)
ages.append(max_age)
print("List after adding Min and Max age:", ages)

# Find the median age
length = len(ages)
if length % 2 == 0:  # Even number of items
    median_age = (ages[length // 2 - 1] + ages[length // 2]) / 2
else:  # Odd number of items
    median_age = ages[length // 2]
print("Median Age:", median_age)

# Find the average age
average_age = sum(ages) / length
print("Average Age:", average_age)

# Find the range of the ages
range_of_ages = max_age - min_age
print("Range of Ages:", range_of_ages)
```

```
Sorted Ages: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
Min Age: 19
Max Age: 26
List after adding Min and Max age: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
Median Age: 24.0
Average Age: 22.75
Range of Ages: 7
```

Q2

```python
# Create an empty dictionary called dog
dog = {}

# Add name, color, breed, legs, age to the dog dictionary
dog['name'] = 'Leo'
dog['color'] = 'White'
dog['breed'] = 'Labrador'
dog['legs'] = 4
dog['age'] = 7
print("Dog Dictionary:", dog)

# Create a student dictionary and add first_name, last_name, gender, age, marital status,
# skills, country, city and address as keys for the dictionary
student = {
    'first_name': 'Soham',
    'last_name': 'Patil',
    'gender': 'Male',
    'age': 22,
    'marital_status': 'Single',
    'skills': ['C#', 'Python'],
    'country': 'United States Of America',
    'city': 'Boston',
    'address': '1 Greenwood Circle'
}
print("\nStudent Dictionary:", student)

# Get the length of the student dictionary
print("\nLength of Student Dictionary:", len(student))

# Get the value of skills and check the data type
skills = student['skills']
print("\nSkills:", skills)
print("Type of Skills:", type(skills))

# Modify the skills values
skills.append('C#')
skills.append('Python')
print("\nSkills:", skills)

# Get the dictionary keys as a list
keys_list = list(student.keys())
print("\nKeys List:", keys_list)

# Get the dictionary values as a list
values_list = list(student.values())
print("\nValues List:", values_list)
```

```
Dog Dictionary: {'name': 'Leo', 'color': 'White', 'breed': 'Labrador', 'legs': 4, 'age': 7}

Student Dictionary: {'first_name': 'Soham', 'last_name': 'Patil', 'gender': 'Male', 'age': 22, 'marital_status': 'Single', 'skills': ['C#', 'Python'], 'country': 'United States Of America', 'city': 'Boston

Length of Student Dictionary: 9

Skills: ['C#', 'Python']
Type of Skills: <class 'list'>

Skills: ['C#', 'Python', 'C#', 'Python']

Keys List: ['first_name', 'last_name', 'gender', 'age', 'marital_status', 'skills', 'country', 'city', 'address']

Values List: ['Soham', 'Patil', 'Male', 22, 'Single', ['C#', 'Python', 'C#', 'Python'], 'United States Of America', 'Boston', '1 Greenwood Circle']
```

Q3

Q3

```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]

# Find the length of the set it_companies
print(len(it_companies))

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

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

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

# Difference between remove and discard:
# `remove(x)` raises a KeyError if x is not present, while `discard(x)` does nothing.

# Join A and B (Union)
print(A.union(B))

# Find A intersection B
print(A.intersection(B))

# Is A subset of B
print(A.issubset(B))

# Are A and B disjoint sets
print(A.isdisjoint(B))

# Join A with B and B with A (Both will give the same result as union is commutative)
print(A.union(B))
print(B.union(A))

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

# Convert the ages to a set
ages_set = set(age)
print("Length of age list:", len(age))
print("Length of age set:", len(ages_set))

# Delete the set completely
del it_companies
print(it_companies)
```

```
7
{'Google', 'Apple', 'Twitter', 'IBM', 'Oracle', 'Amazon', 'Microsoft', 'Facebook'}
{'Facebook', 'TikTok', 'Amazon', 'Google', 'Snapchat', 'Spotify', 'IBM', 'Twitter', 'Oracle', 'Microsoft', 'Apple'}
{'Facebook', 'TikTok', 'Amazon', 'Google', 'Spotify', 'IBM', 'Twitter', 'Oracle', 'Microsoft', 'Apple'}
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26}
True
False
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26, 27, 28}
{27, 28}
Length of age list: 8
Length of age set: 5
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-910ac7485a56> in <cell line: 51>()
     49 # Delete the set completely
     50 del it_companies
---> 51 print(it_companies)

NameError: name 'it_companies' is not defined

SEARCH STACK OVERFLOW
```

Q4

Q4

```
[16] class Employee:
         # Data member to count the number of Employees
         employee_count = 0
         total_salary = 0

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

             # Increase employee count and add to total salary for average calculation
             Employee.employee_count += 1
             Employee.total_salary += salary

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

     class FulltimeEmployee(Employee):
         # Inherits properties from Employee class
         pass

     # Create instances of Fulltime Employee class and Employee class
     employee1 = Employee("Rahul", "Singh", 50000, "IT")
     employee2 = FulltimeEmployee("Shruthika", "Tiwari", 60000, "HR")

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

     print(f"Employee Count: {Employee.employee_count}")
```

```
Average Salary: 55000.0
Employee Count: 2
```

Github Repo: https://github.com/SXP36810/BigData

Youtube link: https://youtu.be/-rL50857pHA