Question 1.Write the output of following python code :

```
S = "computer"
print(S[::2])
print(S[::-1])
print(S[:])
```

(June 2023)

Final Output:

```
cmue
retupmoc
computer
```

Question 2.Assume that the variable data refers to the string "Python rules!". Use a string
method to perform the following tasks:
a. Obtain a list of the words in the string.
b. Convert the string to uppercase.
c. Locate the position of the string "rules" .
d. Replace the exclamation point with a question mark.

```
a). data = "Python rules!"
      a)words = data.split()
print(words)
     b)upper_data = data.upper()
print(upper_data)
   c)position = data.find("rules")
print(position)
   d)new_data = data.replace("!", "?")
print(new_data)
```

Question 3. Write a code segment that opens a file for input and prints the number of fourletter words in the file
(June 2023)

```
 with open("input.txt", "r") as file:
    words = file.read().split()
    count = sum(1 for word in words if len(word) == 4)
print("Number of four-letter words:", count)
```

4.What are mutable and immutable properties in the case of Python data structures? Give one example each for mutable and immutable data structures in Python

a. Mutable Objects: Can be changed after creation.
Example:

```
my_list = [1, 2, 3]
my_list.append(4)  # Modifies the list by adding an element
```

```
my_list[1] = 5  # Changes an existing element
print(my_list)
```

b. Immutable Objects: Cannot be changed after creation.
Example:

```
my_tuple = (1, 2, 3)
my_tuple[1] = 5
```

Question 5.Differentiate between lists and tuples with the help of examples(jan 2024)

Difference Between Lists and Tuples in Python

a. Mutability:

Lists are mutable, meaning their elements can be changed, added, or removed.

Tuples are immutable, meaning their elements cannot be changed once created.

b. Syntax:

A list is created using square brackets [], e.g., my_list = [1, 2, 3].

A tuple is created using parentheses (), e.g., my_tuple = (1, 2, 3).

c. Performance:

Tuples are faster than lists because they are immutable and require less memory.

Lists are slightly slower due to their dynamic nature.

d. Usage:

Use a list when data needs to be modified frequently.

Use a tuple when data should remain constant.

Example:

```
 List example (mutable)
my_list = [1, 2, 3]
my_list.append(4)  #Allowed
print(my_list)  Output: [1, 2, 3, 4]
```

Tuple example (immutable)
my_tuple = (1, 2, 3)
my_tuple.append(4)
print(my_tuple) # Output: (1, 2, 3)

Question 6.Illustrate the use of any four dictionary methods.(Jan 2024)

Creating a dictionary
my_dict = {"name": "Alice", "age": 25, "city": "New York"}

1. get() - Retrieves a value for a given key
   print(my_dict.get("age")) # Output: 25

2. keys() - Returns all keys in the dictionary
   print(my_dict.keys()) # Output: dict_keys(['name', 'age', 'city'])

3. values() - Returns all values in the dictionary
print(my_dict.values()) # Output: dict_values(['Alice', 25, 'New York'])

4. pop() - Removes a key-value pair and returns the value
   removed_value = my_dict.pop("city")
   print(removed_value) # Output: New York
   print(my_dict) # Output: {'name': 'Alice', 'age': 25}

Question 7.Describe the concept of recursive function in Python with suitable example(may 2024)
Concept of Recursive Function in Python
A recursive function is a function that calls itself to solve a problem by breaking it down into smaller subproblems. It must have a base case to stop the recursion.

Example: Factorial using Recursion

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))
```

Recursion is useful for problems like factorial, Fibonacci sequence, and tree traversals.

Question 8.Explain how to read numeric values from a file, perform some operations, and
then write the results back to the file?
(May 2024)

Reading Numeric Values from a File, Performing Operations, and Writing Back

1. Read numeric values from a file.

2. Perform operations (e.g., sum, square, etc.).

3. Write results back to the file.

Example: Squaring Numbers and Writing Back

```
Step 1: Read numbers from file
        with open("numbers.txt", "r") as file:
            numbers = list(map(int, file.read().split()))
Step 2: Perform an operation (square each number)
        squared_numbers = [num ** 2 for num in numbers]
Step 3: Write results back to the file
        with open("results.txt", "w") as file:
            for num in squared_numbers:
        file.write(str(num) + "\n")
        print("Squared numbers written to results.txt")
```

Question 9.Compare and contrast the fundamental characteristics and use cases of lists,
tuples, and sets in Python(may 2024)

Comparison of Lists, Tuples, and Sets in Python

1. Lists (list)

Mutable (can be changed).

Allows duplicate elements.

Ordered (elements have a fixed order).

Used for storing and modifying collections of items.

Example:

```
my_list = [1, 2, 3, 2]
my_list.append(4)
print(my_list)  # [1, 2, 3, 2, 4]
```

2. Tuples (tuple)

Immutable (cannot be changed).

Allows duplicate elements.

Ordered like lists.

Used for fixed data that should not be modified.

Example:

```
my_tuple = (1, 2, 3, 2)
# my_tuple.append(4)
print(my_tuple)  # (1, 2, 3, 2)
```

3. Sets (set)

Mutable, but unordered (no fixed order).

No duplicate elements allowed.

Used for unique collections and mathematical set operations.

Example:

```
my_set = {1, 2, 3, 2}
print(my_set)  # {1, 2, 3}
```