Day 4 Tutorial : List.

Task 1: Knowledge and understanding (list)

1. **What is a list?**

   A list is a group of things written down one below the other. It can be anything you want, like a shopping list, a to -do list, or a list of your favourite animals, it stored in the square brackets[].

2 . Explain the key characteristics of lists.

- Ordered: Lists are ordered collections, which means the items within a list have a specific sequence or position. You can access elements in a list by their index, and the order of elements remains consistent unless explicitly modified.

- Mutable: Lists are mutable, meaning that you can add, remove, or modify elements within a list after it's been created. This allows for dynamic data manipulation and updates.

- Heterogeneous Elements: Lists can store a mix of different data types within the same list. For example, a single list can contain integers, strings, floats, or even other lists as its elements.

- Dynamic Sizing: Lists can change in size during runtime. You can add elements to a list using methods like append(), insert(), or by assigning values to specific indices. Similarly, you can remove elements using methods like remove() or pop().

- Access by Index: Elements in a list are accessed using zero-based indexing. This means that the first element is at index 0, the second element is at index 1, and so on. You can use square brackets to access elements, e.g., my_list[0] to access the first element.

- Iterability: Lists are iterable, meaning you can easily loop through their elements using constructs like for loops. This makes it convenient to perform operations on each element within the list.

- Common Operations: Lists support various common operations such as concatenation, slicing, sorting, and searching for elements. For example, you can use slicing to extract a portion of the list, or you can use the sort() method to arrange the elements in ascending or descending order.

- Nesting: Lists can contain other lists as elements, creating a nested or multi-dimensional structure. This is useful for representing complex data, such as tables or matrices.

- Duplicates: Lists can contain duplicate elements. There is no restriction on having the same value multiple times within a list.

3.How do you access individual elements in a list?

Can access individual elements in a list by using indexing. In most programming languages, including Python, list indices start at 0 for the first element, and you can use square brackets to access elements.

Here are the basics of how to access elements in a list**:**

- Accessing by index
- Negative index
- Slicing

4. What happens if you try to access an index that is out of the list's range?

If you try to access an index that is out of the list's range, you will get an IndexError. This is a Python exception that is raised when you try to access an element in a list using an index that is less than zero or greater than the length of the list**.**

For example:

```
my_list = [1, 2, 3]

# Try to access the fourth element in the list.
print(my_list[3])
```

5. Can a list contain elements of different data types? Give an example.

Yes, a list can contain elements of different data types. This is one of the key features of lists, and it makes them a very versatile data structure**.**

Example:

```
my_list = [1, "hello", True, 3.14, None]
```

6. How do you add a new element to the end of a list, and which method can be used?

To add a new element to the end of a list, you can use the append() method. The append() method takes one argument, which is the element to be added to the list.

Example:

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

7. Explain the difference between mutable and immutable data structures, and provide an example of a mutable data structure.

Mutable data structures are data structures that can be changed after they have been created.

Immutable data structures are data structures that cannot be changed after they have been created.

Examples of mutable data structures include:

- Lists
- Dictionaries
- Sets

Examples of immutable data structures include:

- Strings
- Tuples
- Numbers

Task 2: Practical (List)

1) Write a Python program to handle a list of 10 numbers. The program will prompt the user to handle any of the following operations.

a) Add a new number to the end of the list.

b) Remove the first occurrence of a specific number from the list.

c) Calculate the sum of all the numbers in the list.

d) Find the min and max value of the list

e) Sort the list in descending order

f) Create a sub list based on user input (index or item)

```
# Initialize an empty list to store 10 numbers
numbers = []


# Function to display the current list
def display_list():
   print("Current list:", numbers)


# Function to add a number to the end of the list
def add_number():
   num = float(input("Enter a number to add to the end of the list: "))
   numbers.append(num)
   display_list()


# Function to remove the first occurrence of a specific number
def remove_number():
   num = float(input("Enter a number to remove from the list: "))
   if num in numbers:
    numbers.remove(num)
   print(f"{num} has been removed from the list.")
   else:
   print(f"{num} is not in the list.")
   display_list()

   # Function to calculate the sum of all numbers in the list


def calculate_sum():
   total = sum(numbers)
```

```
    print("Sum of all numbers:", total)


# Function to find the min and max value in the list
def find_min_max():
    if len(numbers) == 0:
        print("The list is empty.")
    else:
        min_value = min(numbers)
        max_value = max(numbers)
        print("Minimum value:", min_value)
        print("Maximum value:", max_value)


# Function to sort the list in descending order
def sort_descending():
    numbers.sort(reverse=True)
    display_list()


# Function to create a sublist based on user input
def create_sublist():
    user_input = input(
        "Enter 'index' to create a sublist based on an index or 'item' to create a sublist based on an
item: ")
    if user_input == 'index':
        index = int(input("Enter the index to start the sublist: "))
        sub_list = numbers[index:]
        print("Sublist:", sub_list)
    elif user_input == 'item':
        item = float(input("Enter an item to create a sublist: "))
        sub_list = [num for num in numbers if num == item]
        print("Sublist:", sub_list)
    else:
        print("Invalid input.")


# Main program loop
while True:
    print("\nOptions:")
    print("a) Add a new number to the end of the list")
    print("b) Remove the first occurrence of a specific number from the list")
    print("c) Calculate the sum of all the numbers in the list")
    print("d) Find the min and max value of the list")
    print("e) Sort the list in descending order")
    print("f) Create a sub list based on user input")
    print("q) Quit")
```

```
    choice = input("Enter your choice: ").lower()

    if choice == 'a':
        add_number()
    elif choice == 'b':
        remove_number()
    elif choice == 'c':
        calculate_sum()
    elif choice == 'd':
        find_min_max()
    elif choice == 'e':
        sort_descending()
    elif choice == 'f':
        create_sublist()
    elif choice == 'q':
        break
    else:
        print("Invalid choice. Please select a valid option.")

# End of the program
print("Goodbye!")
```

2) Write a Python program to handle a predefined list with mixture of data (numbers, characters). The program will prompt the user to handle any of the following operations.

a) Add a new string to the a specific location of the list.

b) Extend the list with new number

c) Find the number of occurrence of a specific item in the list

d) Sort the list in descending order

e) Replace the item with another new item

```
# Predefined list with a mixture of data
predefined_list = [42, 'apple', 7.5, 'banana', 10, 'cherry']


# Function to display the current list
def display_list():
    print("Current list:", predefined_list)
```

```
# Function to add a new string to a specific location in the list
def add_string():
    location = int(input("Enter the index to add a new string: "))
    new_string = input("Enter the new string: ")
    predefined_list.insert(location, new_string)
    display_list()


# Function to extend the list with a new number
def extend_list():
    new_number = float(input("Enter the new number to add to the list: "))
    predefined_list.append(new_number)
    display_list()


# Function to find the number of occurrences of a specific item in the list
def count_occurrences():
    item = input("Enter the item to count in the list: ")
    count = predefined_list.count(item)
    print(f"Number of occurrences of {item} in the list: {count}")


# Function to sort the list in descending order
def sort_descending():
    predefined_list.sort(reverse=True)
    display_list()


# Function to replace an item with another new item
def replace_item():
    item_to_replace = input("Enter the item to replace: ")
    if item_to_replace in predefined_list:
        new_item = input("Enter the new item: ")
        index = predefined_list.index(item_to_replace)
        predefined_list[index] = new_item
        print(f"{item_to_replace} has been replaced with {new_item}.")
    else:
        print(f"{item_to_replace} is not in the list.")
    display_list()


# Main program loop
while True:
    print("\nOptions:")
    print("a) Add a new string to a specific location of the list")
    print("b) Extend the list with a new number")
    print("c) Find the number of occurrences of a specific item in the list")
    print("d) Sort the list in descending order")
```

```
    print("e) Replace an item with another new item")
    print("q) Quit")

    choice = input("Enter your choice: ").lower()

    if choice == 'a':
        add_string()
    elif choice == 'b':
        extend_list()
    elif choice == 'c':
        count_occurrences()
    elif choice == 'd':
        sort_descending()
    elif choice == 'e':
        replace_item()
    elif choice == 'q':
        break
    else:
        print("Invalid choice. Please select a valid option.")

# End of the program
print("Goodbye!")
```