List in python Used to store multiple items of different types i.e. hetrogeneous in nature.It can be implemented by the help of square brackets.Each item stored has its own index starting from 0. List is mutable while programming.Can be nested,sliced,combinable,copiable.These are always in an order.

```
list=["Mehak","Jacky","Jacky","Oggy"]
print(list)
```

```
['Mehak', 'Jacky', 'Jacky', 'Oggy']
```

It can also be created using "list" constructor

```
# thislist = list(("Mehak","Jacky","Jacky","Oggy"))
# print(thislist)
```

Methods in list

```
#append
#list.append("item")
list1=["Mehak","Jacky","Jacky","Oggy"]
list1.append("Bob")
print(list1)
list2=["Cockroach","Dogy"]
list1.append(list2)
print(list1)

#clear
list3=["Mehak","Jacky"]
list3.clear()
print(list3)

#copy
list4=list1.copy()
print(list4)

#count
print(list2.count("Dogy"))

#extend

list3.extend(list4)
print(list3)

#index
print(list1.index("Mehak"))

#insert
```

```
list1.insert(1,"Olivia")
print(list)

#pop
list5=["mehak","Shinchan","Doramon","Nobita"]
list5.pop(2)
print(list5)

#remove
list5.remove("mehak")
print(list5)

#reverse
list5.reverse()
print(list5)


#sort
list5.sort()
print(list5)
```

```
['Mehak', 'Jacky', 'Jacky', 'Oggy', 'Bob']
['Mehak', 'Jacky', 'Jacky', 'Oggy', 'Bob', ['Cockroach', 'Dogy']]
[]
['Mehak', 'Jacky', 'Jacky', 'Oggy', 'Bob', ['Cockroach', 'Dogy']]
1
['Mehak', 'Jacky', 'Jacky', 'Oggy', 'Bob', ['Cockroach', 'Dogy']]
0
['Mehak', 'Jacky', 'Jacky', 'Oggy']
['mehak', 'Shinchan', 'Nobita']
['Shinchan', 'Nobita']
['Nobita', 'Shinchan']
['Nobita', 'Shinchan']
```

Q.1 colors = ['red', 'blue', 'green', 'yellow']

Using the colors list defined above, print the:

First element, Second element, Last element, Second-to-last element, Second and third elements, Element at index 4.

```
colors = ['red', 'blue', 'green', 'yellow']
print(colors[0], colors[1], colors[len(colors)-1], colors[len(colors)-
2], colors[2], colors[3])
```

```
red blue yellow green green yellow
```

Q.2 Below is a list with seven integer values representing the daily water level (in cm) in an imaginary lake. However, there is a mistake in the data. The third day's water level should be 693. Correct the mistake and print the changed list.

water_level = [730, 709, 682, 712, 733, 751, 740]

```
water_level = [730, 709, 682, 712, 733, 751, 740]
water_level.pop(2)
water_level.insert(2,693)
print(water_level)
```

```
[730, 709, 693, 712, 733, 751, 740]
```

Q.3 Add the data for the eighth day to the list from above. The water level was 772 cm on that day. Print the list contents afterwards.

```
water_level = [730, 709, 682, 712, 733, 751, 740]
water_level.append(772)
print(water_level)
```

```
[730, 709, 682, 712, 733, 751, 740, 772]
```

Q.4 Still using the same list, add three consecutive days using a single instruction. The water levels on the 9th through 11th days were 772 cm, 770 cm, and 745 cm. Add these values and then print the whole list.

```
water_level = [730, 709, 682, 712, 733, 751, 740]
water_level.extend([772,770,745])
print(water_level)
```

```
[730, 709, 682, 712, 733, 751, 740, 772, 770, 745]
```

Q.5 There are two ways to delete data from a list: by using the index or by using the value. Start with the original water_level list we defined in the second exercise and delete the first element using its index. Then define the list again and delete the first element using its value.

```
water_level = [730, 709, 682, 712, 733, 751, 740]
water_level.remove(730)
print(water_level)
water_level = [730, 709, 682, 712, 733, 751, 740]
water_level.pop(0)
print(water_level)
```

```
[709, 682, 712, 733, 751, 740]
[709, 682, 712, 733, 751, 740]
```

Tuples in python It is an ordered and immutable data-type in python,once it created it can't be modified like list. while making tuple of single element comma must be there also tup=("Mehak",). Tuple constructor mytuple=tuple(("itmen")) to make it mutable first make it list then agin convert into tuple using list() and tuple().

```
#Methods

#count()
tup=(("Mehak","Jacky","Jacky","Oggy"))
print(tup.count("Mehak"))

#Index(element,start,end)
print(tup.index("Jacky",0,4))
```

```
1
1
```

1. Tuple Creation and Access: Create a tuple named colors with the elements 'red', 'green', and 'blue'. Access the second element of the tuple and print it

```
tup=( 'red', 'green','blue')
print(tup[1])
```

```
green
```

1. Immutable Nature: Explain in your own words why tuples are considered immutable. Attempt to modify an element in an existing tuple and observe the resulting error.

```
# Creating a tuple
my_tuple = ('red', 'green', 'blue')

# Attempting to modify the first element
# my_tuple[0] = 'yellow' THIS LINE WILL SHOW ERROR
```

1. Tuple Slicing: Given the tuple numbers = (1, 2, 3, 4, 5), use slicing to extract the elements from index 1 to 3 (inclusive). What would be the output of numbers[::-1]?

```
numbers = (1, 2, 3, 4, 5)
print(numbers[::-1])

print(numbers[1:3])
```

```
(5, 4, 3, 2, 1)
(2, 3)
```

. Tuple Concatenation and Repetition: Create two tuples, fruits with elements 'apple', 'banana', and berries with elements 'strawberry', 'blueberry'. Concatenate the two tuples and store the result in a new tuple named combined *fruits. Repeat the combined* fruits tuple three times and print the result.

```
fruit=('apple', 'banana', 'berries' )
berry=('strawberry', 'blueberry')
fruit=fruit+berry
```

```
fruit=fruit*3
print(fruit)
```

```
('apple', 'banana', 'berries', 'strawberry', 'blueberry', 'apple',
'banana', 'berries', 'strawberry', 'blueberry', 'apple', 'banana',
'berries', 'strawberry', 'blueberry')
```

1. Built-in Tuple Methods: Create a tuple named grades with the elements 90, 85, 92, 88, 95. Use the count() method to find how many times the grade 88 appears in the tuple. Use the index() method to find the index of the grade 92.

```
tup=(90, 85, 92, 88, 95)
print(tup.count(88))
print(tup.index(92))
```

```
1
2
```

Multiple Data Types in a Tuple: Create a tuple named mixed _ types with elements 'apple', 42, and 3.14. Access and print the second element of the tuple.

```
mixed_types=('apple',42,3.14)
print(mixed_types[1])
```

```
42
```

1. Conversion: Convert the list ['cat' , 'dog' , 'rabbit'] into a tuple named animals. Print the tuple to verify the conversion.

```
tup=['cat' , 'dog' , 'rabbit']
animals=tuple(tup)
print(animals)
```

```
('cat', 'dog', 'rabbit')
```

Create a tuple outer _ tuple with two elements: 'apple' and another tuple ('red', 'green', 'yellow'). Access the second element of the inner tuple and print it.

```
outer_tuple=('red', 'green', 'yellow')
print(outer_tuple[1])
```

```
green
```

1.
   - You are managing the inventory for a small bookstore. Create a list of book titles available in the store. Add new titles to the list as they arrive. If a book is sold out, remove it from the list. Write a function to check if a specific book is in stock.

```
# Initial list of book titles available in the store
book = ["yoo", "joo", "goo", "soo", "boo"]
```

```
new_book = "hello"
book.append(new_book)
print("Current inventory:", book)
book_return = input("Enter the book you want: ")
if book_return in book:
    book.remove(book_return)
    print("Updated inventory:", book)
else:
    print("The book is not in the inventory.")

book_check=input("Enter the book to be checked: ")
if book_check in book:
  print("The book is in the inventory.")
else:
  print("The book is not in the inventory.")
```

```
Current inventory: ['yoo', 'joo', 'goo', 'soo', 'boo', 'hello']
Enter the book you want: goo
Updated inventory: ['yoo', 'joo', 'soo', 'boo', 'hello']
Enter the book to be checked: joo
The book is in the inventory.
```

```
# prompt: check version of colab
```

1.
   - Implement a simple to-do list application. Create a list to store tasks. Write functions to add a task, remove a task by its name, and display all tasks. Ensure that the tasks are displayed in the order they were added.

```
list=["Mehak","Jacky","Bob","Good","Jerry","Sponge"]
new_item=input("Enter the new item: ")
list.append(new_item)
print(list)
delete_item=input("Enter the item to be deleted: ")
list.remove(delete_item)
print(list)
```

```
Enter the new item: 12
['Mehak', 'Jacky', 'Bob', 'Good', 'Jerry', 'Sponge', '12']
Enter the item to be deleted: Bob
['Mehak', 'Jacky', 'Good', 'Jerry', 'Sponge', '12']
```

1.
   - Create a list to store items you need to buy from the grocery store. Write functions to add items, remove items, and check if a specific item is already on the list. Ensure that duplicate items are not added.

```
list=["Tomato","Mango","Watermelon","Papaya"]
new_item=input("Enter the new item: ")
if new_item not in list:
  list.append(new_item)
  print(list)
else:
  print("Item already exists")

Enter the new item: papaita
['Tomato', 'Mango', 'Watermelon', 'Papaya', 'papaita']
```

- You are tracking daily temperatures for a month. Create a list to store these temperatures. Write functions to find the highest and lowest temperatures, and to calculate the average temperature for the month.

```
list=[34.5,55.6,77,8,90.9]
print("Highest temperature:",max(list))
print("Lowest temperature:",min(list))
print("Average temperature:",sum(list)/len(list))

Highest temperature: 90.9
Lowest temperature: 8
Average temperature: 53.2
```

1.
   - You are responsible for assigning seats to students in a classroom. Create a list to represent the seating arrangement. Write functions to assign a seat to a new student, find a student's seat by name, and swap seats between two students.

```
# Initial list of seat assignments as lists [student name, seat
number]
seat1 = ["Alice", 1]
seat2 = ["Bob", 2]
seat3 = ["Charlie", 3]
seating_arrangement = [seat1, seat2, seat3]

# Print initial seating arrangement
print("Initial seating arrangement:", seating_arrangement)

# Assign a seat to a new student
new_student = ["David", 4]
seating_arrangement.append(new_student)
print("Seating arrangement after assigning a seat to a new student:",
seating_arrangement)

# Find a student's seat by name
student_to_find = "Bob"
found_seat = None
for seat in seating_arrangement:
    if seat[0] == student_to_find:
```

```
        found_seat = seat
        break

if found_seat:
    print(f"{student_to_find} is seated at seat number:
{found_seat[1]}")
else:
    print(f"{student_to_find} is not found in the seating
arrangement.")

# Swap seats between two students
student1 = "Alice"
student2 = "Charlie"
index1, index2 = None, None

for i in range(len(seating_arrangement)):
    if seating_arrangement[i][0] == student1:
        index1 = i
    if seating_arrangement[i][0] == student2:
        index2 = i

if index1 is not None and index2 is not None:
    # Swap seats
    seating_arrangement[index1], seating_arrangement[index2] =
seating_arrangement[index2], seating_arrangement[index1]

print("Seating arrangement after swapping seats between Alice and
Charlie:", seating_arrangement)
```

```
Initial seating arrangement: [['Alice', 1], ['Bob', 2], ['Charlie',
3]]
Seating arrangement after assigning a seat to a new student:
[['Alice', 1], ['Bob', 2], ['Charlie', 3], ['David', 4]]
Bob is seated at seat number: 2
Seating arrangement after swapping seats between Alice and Charlie:
[['Charlie', 3], ['Bob', 2], ['Alice', 1], ['David', 4]]
```

- Create a playlist for a party. Use a list to store song titles. Write functions to add a song, remove a song by title, and shuffle the playlist. Ensure no duplicate songs are added to the playlist.

```
import random
party=["honey singh","gippy","diljit","jackie"]
new_item=input("Enter the new item: ")
if new_item not in party:
  party.append(new_item)
  print(party)
else:
  print("Item already exists")
```

```
random.shuffle(party)
print(party)
```

```
Enter the new item: gogo
['honey singh', 'gippy', 'diljit', 'jackie', 'gogo']
['jackie', 'honey singh', 'gippy', 'diljit', 'gogo']
```