1. **What is a List .**

Ans: A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements.

2. **What is a Tuple.**

Ans: A tuple is a collection of elements that are ordered and finite, and can contain different types of data.

3. **What is Difference between List And Tuple.**

Ans: Tuple:

1. it is immutable.
2. The implication of iteration is comparatively Faster.
3. A datatype is appropriate for accessing the elements.
4. Tuple consumes Less memory as compared to the list.
5. It does not have many built-in methods.

List:

1. It is mutable.
2. The implication of iteration is time consuming.
3. List is better performing operation .
4. List consume more memory.
5. List have several built -in methods.

4. **What's The Difference Between The Python append()and extend() Methods?**

Ans: Append() and Extend() are used to add elements to a list, but they behave differently.

Append( ) Method:

- **Purpose:** Adds a single element to the end of the list.
- **How it works:** When you use append(), it adds the element as a whole item at the end of the list, meaning that if you append a list to another list, the entire list will be added as a single element (nested list).

Extend( ) Method:

- **Purpose:** Adds each element of an iterable to the end of the list. It extends the list by adding the elements from the iterable.
- **How it works:** When you use extend(), it iterates through the provided iterable and adds each item individually to the list. It does not add the iterable as a single item.
-

5. **How to sort a Tuple.**

Ans: Tuples in Python are immutable, meaning you cannot directly change the elements of a tuple.

Steps to Sort a Tuple:

1. Convert the tuple to a list using list().
2. Sort the list using the sort() method or the sorted() function.
3. Convert the list back to a tuple using tuple().

6. Difference between del and clear ?

Ans: **del Keyword :**

- Purpose: The del keyword is used to delete a variable or an element from a list at a specific index or to delete the entire list.
- How it works:
    - You can use del to delete an element at a specific index in the list.
    - It can also delete the entire list or other variables.
    - del is a statement, not a method, so it works in a broader scope than just lists.

clear( ) Method:

- Purpose**:** The clear() method is used to remove **all elements** from a list, leaving the list empty.
- How it works: clear() modifies the list in-place by removing all its elements, but it does not delete the list itself.

7. Difference between remove and pop ?

Ans: remove() Method :

- **Purpose:** The remove() method is used to remove the first occurrence of a specific element in the list.
- **How it works:**
    - You provide the **value** to be removed, not the index.
    - If the value is found, it is removed from the list.
    - If the value is not found, it raises a Value Error.

pop( ) Method:

- **Purpose:** The pop() method removes an element at a **specific index** and returns the removed element.
- **How it works :**
    - If no index is provided, it removes and returns the **last element** of the list.
    - If an index is provided, it removes and returns the element at that index.
    - If the list is empty and pop() is called, it raises an IndexError.

8. Difference between indexing and Slicing ?

Ans: Indexing:

- **Purpose:** Indexing is used to access a **single element** at a specific position in a sequence.
- **How it works:** You specify the index (position) of the element you want to retrieve.
    - Indexing starts at 0 for the first element, 1 for the second element, and so on.
    - You can also use negative indices to access elements from the end of the sequence, where -1 refers to the last element, -2 to the second last, and so on.

Slicing:

- **Purpose:** Slicing is used to access a **range of elements** from a sequence.
- **How it works:** You specify a **start index**, **end index**, and an optional **step** value.

9. Difference between sort and sorted ?

Ans: sort( ) Method:

- Purpose: The sort() method is used to sort a list in place, meaning it modifies the original list.
- How it works:
    - sort() is a method that can only be used with lists.
    - It does not return a new list ; instead, it modifies the list it is called on and returns None.
    - It can sort in ascending or descending order, based on the reverse parameter.

sorted( ) Function:

- Purpose: The sorted() function is used to return a new sorted list from any iterable .
- How it works:
    - sorted() is a function that can work with any iterable and returns a new sorted list.
    - It does not modify the original iterable.
    - It also allows sorting in ascending or descending order with the reverse parameter, and you can use a key function for custom sorting.

10. Difference between reverse and reversed ?

Ans: Reverse() Method:

- **Purpose:** The reverse() method is used to **reverse a list in place**, meaning it modifies the original list.
- **How it works:**
    - It only works on **mutable sequences** like lists.
    - It reverses the elements of the list and **modifies the list directly** .
    - It **does not return anything** .

Reversed( ) Function:

- **Purpose:** The reversed() function is used to return a **new iterator** that accesses the elements of the original sequence in reverse order.
- **How it works:**
    - It works with **any iterable** (lists, tuples, strings,), not just lists.
    - It does **not modify the original sequence**.
    - It returns a **reversed iterator**, so it doesn't return a list or sequence, but an iterator that you can convert to a list or use in a loop.

11. Difference between copy and deep copy ?

Ans: copy() :

- Creates a **new object** but does not recursively copy the nested objects inside it.
- The new object contains references to the original nested objects, meaning changes to mutable objects inside the copy will affect the original object.
- This is called a **shallow copy** because it only copies the outer object and the references to the inner objects.

Deep Copy( ):

- **new object** and **recursively copies all nested objects** inside it.
- The original and the new object are completely independent of each other.
- Changes made to any nested objects in the copy will not affect the original object.

```
# Q4
a=[12,23,4,5,87,98,22]
max(a)
```

```
98
```

```
my_list=[12,23,4,5,87,98,22]
my_list[0],my_list[-1]=my_list[-1],my_list[0]
# swapp_list=swap_first_last(my_list)
print(my_list)
```

```
[22, 23, 4, 5, 87, 98, 12]
```

```
x=[2,3,5,76,98,7,62]
temp=x[2]
x[2]=x[5]
x[5]=temp
print(x)
```

```
[2, 3, 7, 76, 98, 5, 62]
```

```
z=[2,3,5,76,98,7,62]
z.reverse()
print(z)
```

```
[62, 7, 98, 76, 5, 3, 2]
```

```
z=[2,7,5,76,98,7,62,8,31,7]
z.count(7)
```

```
3
```

```
z=[2,7,5,76,98,7,62,8,31,7]
y=sum(a)
print(y)
```

```
251
```

```
z=[2,7,11,98,5,62,8,31]
for i in z:
    print(i**2)
```

```
4
49
5776
9604
25
3844
64
961
```

```
z=[2,7,11,98,5,62,8,31]
y= len(z)
print(y)
```

```
8
```

```
my_list = [101,978,454, 201, 301,323,687, 401, 501]

smallest = my_list[0]
largest = my_list[0]

for n in my_list:
    if n < smallest:
        smallest = n
    if n > largest:
        largest = n

print("Smallest number:", smallest)
print("Largest number:", largest)
```

```
Smallest number: 101
Largest number: 978
```

```
radius= int(input("Enter a radius of circle:"))
pi=3.14
area=pi*radius*radius
print(f"The area of the circle is: {area}")
```

```
Enter a radius of circle: 5

The area of the circle is: 78.5
```

```
even_no=[]
odd_no=[]
for i in range(0,151):
    if i % 2 == 1:
        odd_no .append(i)
    else:
        even_no.append(i)

print("Odd number Between:",odd_no)
print("Even number Between:",even_no)
```

```
Odd number Between: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25,
27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59,
61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93,
95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121,
123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149]
Even number Between: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24,
26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58,
60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92,
```

```
94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120,
122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148,
150]

my_list=[1,23,32,64,69,78,98,90,80,94,101,86,54,34,23,13,19,30]
even_count=0
odd_count=0
for i in my_list:
    if i % 2 == 1:
        odd_count+=1
    else:
        even_count+=1
print("odd No.count:",odd_count)
print("even No count:",even_count)
```

```
odd No.count: 7
even No count: 11
```

```
div_by_4=[]
div_by_6=[]
div_by_8=[]
div_by_10=[]
div_by_3=[]
div_by_5=[]
div_by_7=[]
div_by_9=[]
for i in range(0,151+1):
    if i % 4==0:
        div_by_4.append(i)
    if i % 6 ==0:
        div_by_6.append(i)
    if i % 8==0:
        div_by_8.append(i)
    if i % 10 ==0:
        div_by_10.append(i)
    if i % 3==0:
        div_by_3.append(i)
    if i % 5==0:
        div_by_5.append(i)
    if i % 7==0:
        div_by_7.append(i)
    if i % 9==0:
        div_by_9.append(i)

print("Number of divisible by 4:",div_by_4)
print()
print("Number of divisible by 6:",div_by_6)
print()
print("Number of divisible by 8:",div_by_8)
print()
```

```
print("Number of divisible by 10:",div_by_10)
print()
print("Number of divisible by 3:",div_by_3)
print()
print("Number of divisible by 5:",div_by_5)
print()
print("Number of divisible by 7:",div_by_7)
print()
print("Number of divisible by 9:",div_by_9)
```

```
Number of divisible by 4: [0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40,
44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100, 104, 108,
112, 116, 120, 124, 128, 132, 136, 140, 144, 148]

Number of divisible by 6: [0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60,
66, 72, 78, 84, 90, 96, 102, 108, 114, 120, 126, 132, 138, 144, 150]

Number of divisible by 8: [0, 8, 16, 24, 32, 40, 48, 56, 64, 72, 80,
88, 96, 104, 112, 120, 128, 136, 144]

Number of divisible by 10: [0, 10, 20, 30, 40, 50, 60, 70, 80, 90,
100, 110, 120, 130, 140, 150]

Number of divisible by 3: [0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33,
36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84,
87, 90, 93, 96, 99, 102, 105, 108, 111, 114, 117, 120, 123, 126, 129,
132, 135, 138, 141, 144, 147, 150]

Number of divisible by 5: [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50,
55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130,
135, 140, 145, 150]

Number of divisible by 7: [0, 7, 14, 21, 28, 35, 42, 49, 56, 63, 70,
77, 84, 91, 98, 105, 112, 119, 126, 133, 140, 147]

Number of divisible by 9: [0, 9, 18, 27, 36, 45, 54, 63, 72, 81, 90,
99, 108, 117, 126, 135, 144]
```

```
My_list=[11,"Tableau",23,89.3,"Power bi ",53.62,"Excel",44,69.17]

integers =[]
strings=[]
floats=[]

for i in My_list:
    if isinstance(i,int):
        integers.append(i)
    elif isinstance(i,str):
        strings.append(i)
    else:
        floats.append(i)
```

```
print("Integers:",integers)
print("String:",strings)
print("Floats:", floats)
```

```
Integers: [11, 23, 44]
String: ['Tableau', 'Power bi ', 'Excel']
Floats: [89.3, 53.62, 69.17]
```

```
list1=[2,65,77,80]
list2=[65,8,70,98]
list1.extend(list2)
print("List 1:",list1)
```

```
List 1: [2, 65, 77, 80, 65, 8, 70, 98]
```

```
list1=[2,65,43,57,98]

list1.sort(reverse=True)

list2=list1[2]

print(list2)
```

```
57
```

```
'''lst=[2,2,2,6,6,6,6,5,5,5,1,3,3,3,3]
element=set(lst)
frequency=[]
for i in element:
    #frequency(i)==frequency
print()'''
```

```
'lst=[2,2,2,6,6,6,6,5,5,5,1,3,3,3,3]\nelement=set(lst)\nfrequency=[]\
nfor i in element:\n    #frequency(i)==frequency\nprint()'
```

```
a=[23,4,54,56,7,90,44,58,77,98]
length=len(a)
print(length)
```

```
10
```

```
my_list=[]
if len(my_list)==0:
    print("list is Empty")
else:
    print("list is not Empty")
```

```
list is Empty
```

```
lst1=[23,34,65,87]
lst2=[21,46,97,87]
```

```
a=lst1+lst2
print(a)
```

```
[23, 34, 65, 87, 21, 46, 97, 87]
```

```
a=[1,2,2,3,3,4,1,2,1,5]
occurrences = a.count(2)
print(occurrences)
```

```
3
```

```
my_list = [[1, 2], [3, 4], [5, 6]]
flattened_list = []      # convert list of list into single list.
for lst in my_list :
    for item in lst:
        flattened_list.append(item)
print(flattened_list)
```

```
[1, 2, 3, 4, 5, 6]
```

```
lst=[11,33,55]
result = int("".join(map(str, lst)))
print(result)
```

```
113355
```