List

List is versatile data structure that allow you to store and manipulate sequences of elements.

- When you only need to iterate over the elements of a list, a for loop is generally the most Pythonic and readable way to do it.
- If you need to modify elements or get both the index and value, consider using enumerate.

Lets discover how to iterate over the list.

In [ ]:
```python
# Iteration using for loop
my_list = [1, 2, 3, 4, 5]

for item in my_list:
    print(item)
```

```
1
2
3
4
5
```

In [ ]:
```python
#Iteration using list index
for i in range(len(my_list)):
    print(my_list[i])
```

```
1
2
3
4
5
```

In [ ]:
```python
# Itreation using list comprehesion
# List comprehension is a concise way to create lists and can be used for iteration
squared_values = [x**2 for x in my_list]
print(squared_values)
```

```
[1, 4, 9, 16, 25]
```

In [ ]:
```python
# Iteration using enumerate()
# The enumerate function can be used to iterate over both the elements and their in
for index, value in enumerate(my_list):
    print(f"Index: {index}, Value: {value}")
```

```
Index: 0, Value: 1
Index: 1, Value: 2
Index: 2, Value: 3
Index: 3, Value: 4
Index: 4, Value: 5
```

In [ ]:
```
# Iteration usinf while loop
# You can use a while loop with an index variable to iterate through the list.
index = 0
while index < len(my_list):
    print(my_list[index])
    index += 1
```

```
1
2
3
4
5
```

List operations

1. Creation of list
2. Accessing element from list
3. Access subset of element using list slicing
4. Modifying list element
5. Adding elements in the existing list
6. Deleting an element
7. Concatenate 2 lists
8. Finding list length
9. Checking Membership of list element
10. To count occurences of list element
11. Sort list elements in ascending order
12. Reverse the order of list elements

1. Creation of list

In [ ]:
```
my_list = [1, 2, 3, 4, 5]
my_list
```

Out[ ]:
```
[1, 2, 3, 4, 5]
```

2. Accessing element from list

In [ ]:
```
# Accessing elements by index
print(my_list[0])  # Output: 1

# Negative indexing (counting from the end)
print(my_list[-1])  # Output: 5
```

```
1
5
```

3. Access subset of element using list slicing

In [ ]:
```
# Slicing to get a subset of elements
subset = my_list[1:4]  # Elements at index 1, 2, 3
print(subset)  # Output: [2, 3, 4]
```

```
[2, 3, 4]
```

4. Modifying list element

In [ ]:
```
# Updating an element
my_list[2] = 10
print(my_list)  # Output: [1, 2, 10, 4, 5]
```

```
[1, 2, 10, 4, 5]
```

5. Adding elements in the existing list

In [ ]:
```
# Using append() method
my_list.append(6)  # Appends 6 to the end
print(my_list)  # Output: [1, 2, 10, 4, 5, 6]
```

```
[1, 2, 10, 4, 5, 6]
```

In [ ]:
```
# Using insert() method
my_list.insert(2, 8)  # Inserts 8 at index 2
print(my_list)  # Output: [1, 2, 8, 10, 4, 5, 6]
```

```
[1, 2, 8, 10, 4, 5, 6]
```

6. Deleting element

In [ ]:
```
# Delete by value
my_list.remove(8)  # Removes the first occurrence of 8
print(my_list)  # Output: [1, 2, 10, 4, 5, 6]
```

```
[1, 2, 10, 4, 5, 6]
```

In [ ]:
```
# Delete by Index
del my_list[1]  # Removes the element at index 1
print(my_list)  # Output: [1, 10, 4, 5, 6]
```

```
[1, 10, 4, 5, 6]
```

7. Concatenate 2 lists

In [ ]:
```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
concatenated_list = list1 + list2
print(concatenated_list)  # Output: [1, 2, 3, 4, 5, 6]
```

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

8. Finding list length

In [ ]:
```
length = len(my_list)
print(length)  # Output: 5
```

```
5
```

9. Checking Membership of list element

In [ ]:
```
print(3 in my_list)  # Output: True
```

```
False
```

10. To count occurences of list element

In [ ]:
```
count = my_list.count(10)
print(count)  # Output: 1
```

```
1
```

11. Sort list elements in ascending order

In [ ]:
```
my_list.sort()  # Sorts the list in ascending order
print(my_list)
```

```
[1, 4, 5, 6, 10]
```

12. Reverse the order of list elements

In [ ]:
```
my_list.reverse()  # Reverses the order of elements
print(my_list)
```

```
[10, 6, 5, 4, 1]
```

In [ ]: