Day 22

Python Lists

- Lists are ordered collection of data items.
- They store multiple items in a single variable.
- List items are separated by commas and enclosed within square brackets [].
- Lists are changeable meaning we can alter them after creation.

Example 1:

```
lst1 = [1,2,2,3,5,4,6]
lst2 = ["Red", "Green", "Blue"]
print(lst1)
print(lst2)
```

Output:

```
[1, 2, 2, 3, 5, 4, 6]
['Red', 'Green', 'Blue']
```

Example 2:

```
details = ["Abhijeet", 18, "FYBScIT", 9.8]
print(details)
```

Output:

```
['Abhijeet', 18, 'FYBScIT', 9.8]
```

As we can see, a single list can contain items of different data types.

List Index

Each item/element in a list has its own unique index. This index can be used to access any particular item from the list. The first item has index [0], second item has index [1], third item has index [2] and so on.

Example:

```
colors = ["Red", "Green", "Blue", "Yellow", "Green"]
#          [0]      [1]      [2]      [3]       [4]
```

Accessing list items

We can access list items by using its index with the square bracket syntax []. For example colors[0] will give "Red", colors[1] will give "Green" and so on...

Positive Indexing:

As we have seen that list items have index, as such we can access items using these indexes.

Example:

```
colors = ["Red", "Green", "Blue", "Yellow", "Green"]
#          [0]      [1]      [2]      [3]       [4]
print(colors[2])
print(colors[4])
print(colors[0])
```

Output:

```
Blue
Green
Red
```

Negative Indexing:

Similar to positive indexing, negative indexing is also used to access items, but from the end of the list. The last item has index [-1], second last item has index [-2], third last item has index [-3] and so on.

Example:

```
colors = ["Red", "Green", "Blue", "Yellow", "Green"]
#          [-5]     [-4]     [-3]     [-2]      [-1]
print(colors[-1])
print(colors[-3])
print(colors[-5])
```

Output:

```
Green
Blue
Red
```

Check whether an item in present in the list?

We can check if a given item is present in the list. This is done using the in keyword.

Example:

```
colors = ["Red", "Green", "Blue", "Yellow", "Green"]
if "Yellow" in colors:
   print("Yellow is present.")
else:
   print("Yellow is absent.")
```

Output:

```
Yellow is present.
```

Example:

```
colors = ["Red", "Green", "Blue", "Yellow", "Green"]
if "Orange" in colors:
   print("Orange is present.")
else:
   print("Orange is absent.")
```

Output:

```
Orange is absent.
```

Range of Index:

You can print a range of list items by specifying where you want to start, where do you want to end and if you want to skip elements in between the range.

Syntax:

```
listName[start : end : jumpIndex]
```

Note: jump Index is optional. We will see this in later examples.

Example: printing elements within a particular range:

```
animals = ["cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow"]
print(animals[3:7])        #using positive indexes
print(animals[-7:-2])      #using negative indexes'
```

Output:

```
['mouse', 'pig', 'horse', 'donkey']
['bat', 'mouse', 'pig', 'horse', 'donkey']
```

Here, we provide index of the element from where we want to start and the index of the element till which we want to print the values.

Note: The element of the end index provided will not be included.

Example: printing all element from a given index till the end

```
animals = ["cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow"]
print(animals[4:])          #using positive indexes
print(animals[-4:])         #using negative indexes
```

Output:

```
['pig', 'horse', 'donkey', 'goat', 'cow']
['horse', 'donkey', 'goat', 'cow']
```

When no end index is provided, the interpreter prints all the values till the end.

Example: printing all elements from start to a given index

```
animals = ["cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow"]
print(animals[:6])          #using positive indexes
print(animals[:-3])         #using negative indexes
```

Output:

```
['cat', 'dog', 'bat', 'mouse', 'pig', 'horse']
['cat', 'dog', 'bat', 'mouse', 'pig', 'horse']
```

When no start index is provided, the interpreter prints all the values from start up to the end index provided.

Example: Printing alternate values

```
animals = ["cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow"]
print(animals[::2])                 #using positive indexes
print(animals[-8:-1:2])   #using negative indexes
```

Output:

```
['cat', 'bat', 'pig', 'donkey', 'cow']
['dog', 'mouse', 'horse', 'goat']
```

Here, we have not provided start and index, which means all the values will be considered. But as we have provided a jump index of 2 only alternate values will be printed.

Example: printing every 3rd consecutive value withing a given range

```
animals = ["cat", "dog", "bat", "mouse", "pig", "horse", "donkey", "goat", "cow"]
print(animals[1:8:3])
```

Output:

```
['dog', 'pig', 'goat
```

Here, jump index is 3. Hence it prints every 3rd element within given index.

List Comprehension

List comprehensions are used for creating new lists from other iterables like lists, tuples, dictionaries, sets, and even in arrays and strings.

Syntax:

List = [Expression(item) for item in iterable if Condition]

Expression: It is the item which is being iterated.

Iterable: It can be list, tuples, dictionaries, sets, and even in arrays and strings.

Condition: Condition checks if the item should be added to the new list or not.

Example 1: Accepts items with the small letter “o” in the new list

```
names = ["Milo", "Sarah", "Bruno", "Anastasia", "Rosa"]
namesWith_O = [item for item in names if "o" in item]
print(namesWith_O)
```

Output:

```
['Milo', 'Bruno', 'Rosa']
```

Example 2: Accepts items which have more than 4 letters

```
names = ["Milo", "Sarah", "Bruno", "Anastasia", "Rosa"]
namesWith_O = [item for item in names if (len(item) > 4)]
print(namesWith_O)
```

Output:

```
['Sarah', 'Bruno', 'Anastasia']
```