Lists

Data structures are containers that organize and group data types together in different ways. A list in python is one of the most common and basic data structures that are written in square brackets "[]" with elements stored inside separated by commas. A list is also ordered and mutable. For example:

Code	Output

`list = ["Hi!", "Welcome", "to", "CSE", "110"]`
 	
`print(list)`
	`['Hi!', 'Welcome', 'to', 'CSE', '110']`

Mutability

The word "mutability" in python means the ability for certain types of data to be changed without recreating it entirely. It makes the program run more efficiently and quickly. For example:

Code	Output

`beatles = ["John", "Paul", "Alonzo", "Ringo"]`
 	
`#Before Modification`
`print(beatles)`
	`['John', 'Paul', 'Alonzo', 'Ringo']`

`beatles[2] = "George"`
 	
`#After Modification`
`print(beatles)`
	`['John', 'Paul', 'George', 'Ringo']`

Necessity of Lists

Lists don't need to be always homogeneous. They maintain the order of sequences whose values are not fixed. We can easily access and modify values inside a list. For example, adding value or removing is possible in a list. On the other hand, tuples are immutable and cannot be changed (you will get an explanation in the next section named "tuple"). Again, in python lists and strings are quite similar. We can use the "for" loop to iterate over lists, "+" plus operator in order to concatenate and use in a keyword to check if the sequence contains a value.

To access the Items of Lists:

1. Print the items in a list, we use index values:

Code	Output
`example_list = ["Kitkat", "Oreo", "Hersheys"]` `print(example_list[2])`	`Hersheys`

2. In a list, negative indexing means beginning from the end. Example: -1 refers to the last item, -2 refers to the second-last item, etc..

Code	Output
example_list = ["Kitkat", "Oreo","Hersheys"] print(example_list[-1]) print(example_list[-2])	Hersheys Oreo

3. While specifying a range in a list, the return value will give a new list with the specified items:

Code	Output
example_list = ["Hersheys","Kitkat", "Oreo", "MIMI", "Cadbury", "Monchuri-Milk Candy"] print(example_list[2:5])	['Oreo', 'MIMI', 'Cadbury']

4. While specifying a range of negative indexes in a list, we can start the search from the end of a list. Given example returns the items from index -5 (included) to index -1 (excluded).

Code	Output
example_list = ["Hersheys","Kitkat", "Oreo", "MIMI", "Cadbury", "Monchuri-Milk Candy"] print(example_list[-5:-1])	['Kitkat', 'Oreo', 'MIMI', 'Cadbury']

5. By leaving out the start value, the range will start from the first item of the list. Again, by leaving out the end value, the range will go on to the end of the list.

Code	Output
#Example-1)leaving out the start value: example_list = ["Hersheys","Kitkat", "Oreo", "MIMI", "Cadbury", "Monchuri-Milk Candy"] print(example_list[:3])	['Hersheys', 'Kitkat', 'Oreo']
#Example-2)leaving out the end value: example_list = ["Hersheys","Kitkat", "Oreo", "MIMI", "Cadbury", "Monchuri-Milk Candy"] print(example_list[4:])	['Cadbury', 'Monchuri-Milk Candy']

6. Since lists are mutable, we can change the items inside it.

Code	Output
example_list = ["Hersheys","Kitkat", "Oreo", "MIMI", "Cadbury", "Monchuri-Milk Candy"] print(example_list[2:5])	['Oreo', 'MIMI', 'Cadbury']

Basic Operations of List:

Example:	Output
#1) To make a copy of items of the list into a new list, using [:] operator: example_list = ["Hersheys","Kitkat", "Oreo", "MIMI", "Cadbury", "Monchuri-Milk Candy"] duplicate_list = example_list [:] print(duplicate_list)	['Hersheys', 'Kitkat', 'Oreo', 'MIMI', 'Cadbury', 'Monchuri-Milk Candy']
#2) The copied list remains unchanged when the item of original list gets modified: original_list = ["Hersheys","Kitkat", "Oreo", "MIMI", "Cadbury", "Monchuri-Milk Candy"] duplicate_list = original_list [:] original_list[-1] = "Pran Mango Bar" print(original_list) print (duplicate_list)	['Hersheys', 'Kitkat', 'Oreo', 'MIMI', 'Cadbury', **'Pran Mango Bar'**] ['Hersheys', 'Kitkat', 'Oreo', 'MIMI', 'Cadbury', **'Monchuri-Milk Candy'**]
#3) Looping through list using for loop: example_list = ["Hersheys","Kitkat", "Oreo", "MIMI"] for i in example_list: print(i)	Hersheys Kitkat Oreo MIMI
#4) To check if an item exists in the tuple:	Yes,MIMI is in the list

`example_list = ["Hersheys","Kitkat", "Oreo", "MIMI"]`
 	
`if "MIMI" in example_list:`
`  print("Yes,MIMI is in the list")`	

#5) To know the number of items in a list:
 	
`example_list = ["Hersheys","Kitkat", "Oreo", "MIMI"]`
 	
`print(len(example_list))`
	`4`

#6) List membership Test:
 	
`example_list = ["Hersheys","Kitkat", "Oreo", "MIMI"]`
 	
`print('Hersheys' in example_list)`
 	
`print('Cadbury' in example_list)`
	`True`
	
	`False`

Methods Associated with Lists:

Methods Associated with Lists	
L.append(e)	adds the object e to the end of L.
L.count(e)	returns the number of times that e occurs in L.
L.insert(i,e)	inserts the object e into L at index i.
L.extend(L1)	adds the items in list L1 to the end of L.
L.remove(e)	deletes the first occurrence of e from L.
L.index(e)	returns the index of the first occurrence of e in L.
L.pop(i)	removes and returns the item at index i in L. If i is omitted, it defaults to -1, to remove and return the last element of L.
L.sort()	sorts the elements of L in ascending order.
L.reverse()	reverses the order of the elements in L.

Built-in Methods of Lists with Examples:

Method	Code	Output
append()	subjects = ['CSE', 'EEE', 'Civil'] subjects.append('Mechanical') print(subjects)	['CSE', 'EEE', 'Civil', 'Mechanical']
count()	subjects = ['CSE', 'EEE', 'Civil', 'CSE'] x = subjects.count('CSE') print(x)	2
insert()	subjects = ['CSE', 'EEE', 'Civil'] subjects.insert(1, 'Mechanical') print(subjects)	['CSE', 'Mechanical', 'EEE', 'Civil']
extend()	subjects = ['CSE', 'EEE', 'Civil'] cars = ['Ford', 'BMW', 'Volvo'] subjects.extend(cars) print(subjects)	['CSE', 'EEE', 'Civil', 'Ford', 'BMW', 'Volvo']
remove()	subjects = ['CSE', 'EEE', 'Civil'] subjects.remove('Civil') print(subjects)	['CSE', 'EEE']
index()	subjects = ['CSE', 'EEE', 'Civil'] x = subjects.index('Civil') print(x)	2
pop()	subjects = ['CSE', 'EEE', 'Civil'] subjects.pop(1) print(subjects)	['CSE', 'Civil']
sort()	cars = ['Ford', 'BMW', 'Volvo'] cars.sort() print(cars)	['BMW', 'Ford', 'Volvo']
reverse()	subjects = ['CSE', 'EEE', 'Civil'] subjects.reverse() print(subjects)	['Civil', 'EEE', 'CSE']

Slicing

We can understand slicing by visualizing the index to be in between the elements as shown below. How the slicing works have been explained in detail in the String chapter. The slicing mechanism works identically in all the data structures.

String	F	A	N	T	A	S	T	I	C
Positive indexing	0	1	2	3	4	5	6	7	8
Negative indexing	-9	-8	-7	-6	-5	-4	-3	-2	-1

Slicing elements of a list in python

If we want to access a range, we need two indices that will slice that portion from the list. For example:

Example:	**Output**

`#Declare a List named 'my_list'`
 	
`my_list = ['F','A','N','T','A','S','T','I','C']`	

`#1) elements 3rd to 5th`
 	
`print(my_list[2:5])`
	`['N', 'T', 'A']`

`#2) elements beginning to 4th`
 	
`print(my_list[:-5])`
	`['F', 'A', 'N', 'T']`

`#3) elements 6th to end`
 	
`print(my_list[5:])`
	`['S', 'T', 'I', 'C']`

`#4) elements beginning to end`
 	
`print(my_list[:])`
	`['F', 'A', 'N', 'T', 'A', 'S', 'T', 'I', 'C']`