Python!

By- Mugdha Panhale,

Finitely Infinite Systems Pvt. Ltd.

Data Structures in Python

- Lists
- Dictionaries
- Tuples
- Sets

List

What is a data structure?
 - a data structure is a way of organizing data in an easy to manage format that enables efficient access and modification of the data that resides in the structure itself.

And the first type of data structure we're going to be exploring is the list. It is built-in data structure in Python, that allows us to work with a collection of data in a sequenced order.

To create a List Data Structure:-
Syntax:

```
list_name = [ ]
```

type(list_name) → **<class 'list'>**

List

Examples:

- cars = [‘Audi Q7’, ‘Tesla Model S’, ‘BMW X5’, ‘Mercedes-Benz GLE’, ‘Lexus RX’]
- even_numbers = [2, 4, 6, 8, 10]
- mixed_list = [True, ‘hello’, 0.49, 100, [1, 2, 3, 4]]

Accessing List Elements:

numbers = [1, 2, 3, 4, 5] # index starts with 0 [zero]
numbers[0] → o/p : 1
numbers[1] → o/p : 2
numbers[-1] → o/p : 5
numbers[7] → o/p : IndexError: list index out of range
numbers[1.5] → o/p : TypeError: list indices must be integers or slices, not float

position = [1, 2, 3, [4, 5, 6]]
print(position[3][2]) → o/p : **5**
print(position[-1][-2] → **o/p : 5 # list supports negative indexing starting with -1.**

List Slicing

Slicing: Extracted portion of a list. We provide two integers in the square bracket syntax separated by a colon.
The first integer represents the index of the first item we want to go from and this value is included in your result.
And the second integer value is the value we want to go up to ie it is one more than the last index we want to include.
The third value (its optional) that you can pass into a slicing operation and this value pertains to the step.

Examples:

- numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- numbers[2: 6] → [3, 4, 5, 6]
- numbers[: 6] → [1, 2, 3, 4, 5, 6]
- numbers[6 :] → [7, 8, 9, 10]
- numbers[: : 2] → [1, 3, 5, 7, 9]
- numbers[1 : : 2] → [2, 4, 6, 8, 10]

List Methods

Strings are immutable objects, ie their structure cannot be changed.
Lists are mutable objects, so their structure can be changed.
Examples:

- **greetings = [‘hello’, ‘hi’, ‘hey’]**
- greetings.reverse() →[‘hey’, ‘hi’, ‘hello’]
- **shopping_list = [‘cheese’, ‘milk’, ‘apple’, ‘bread’]**
- shopping_list.sort() →[‘apple’, ‘bread’, ‘cheese’, ‘milk’] # default in Ascending order
- shopping_list.sort(**reverse = Ture**) →[‘milk’, ‘cheese’, ‘bread’, ‘apple’]
- sorted_list = shopping_list.sort()
- print(sorted_list) → None
- **Note: when using the built- in sort method, remember the original list is modified.**
- **If you want new sorted list rather than modified original list, we use sorted() function.**
- new_sorted_list = sorted(shopping_list)
- print(new_sorted_list) →[‘apple’, ‘bread’, ‘cheese’, ‘milk’]
- sorted(shopping_list, **reverse = True**) → [‘milk’, ‘cheese’, ‘bread’, ‘apple’]

List Methods

Examples:

- lst = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’]
- print(lst[0]) → ‘a’
- lst [1 : 4] = [‘p’, ‘q’, ’r’]
- print(lst) → [‘a’, ‘p’, ‘q’, ‘r’, ‘e’, ’f’]
- lst [0 : 3] = [‘m’, ‘n’]
- print(lst) → [‘m’, ‘n’, ‘d’, ‘e’, ‘f’] # It removes 3 items and refills two values.
- lst1 = [‘j’, ‘k’]
- lst += lst1
- print(lst) → [‘a’, ‘p’, ‘q’, ‘r’, ‘e’, ’f’, ‘l’, ‘m’] #list concatenation
- lst.append(‘k’) →[‘a’, ‘p’, ‘q’, ‘r’, ‘e’, ’f’, ‘k’]
- lst.append(lst1) →[‘a’, ‘p’, ‘q’, ‘r’, ‘e’, ’f’, [’j’, ‘k’]] # Nested List
- lst.extend(lst1) →[‘a’, ‘p’, ‘q’, ‘r’, ‘e’, ’f’, ’j’, ‘k’] # adds as a individual elements in a List
- lst.insert(1, ‘z’) → [‘a’, ‘z’, ‘b’, ‘c’, ‘d’]
- lst.insert(1, lst1) → ['a', [‘j', ‘k'], 'b', 'c', 'd', 'e', 'f']

List Methods

Examples:

- lst = [‘a’, ‘b’, ‘c’, ‘a’, ‘f’]
- lst.remove(‘a’) → ['b', 'c', 'a', 'f’]
- lst.pop() → [‘a’, ‘b’, ‘c’, ‘a’] #last value will be popped if no argument provided.
- lst.pop(-3) → [‘a’, ‘b’, ‘a’, ‘f’]
- removed_element = lst.pop(-3) Or (3) → [‘a’, ‘b’, ‘a’, ‘f’] # we can collect popped element.
- print(removed_element) → ’c’
- lst = [‘a’, ‘b’, ‘c’, ‘a’, ‘f’]
- del lst[1 : 4] → [‘a’, ‘a’, ‘f’]
- del lst
- print(lst) → #NameError: name 'lst' is not defined. # it deletes entire list
- lst.clear() → [ ] # it clears list elements and saves empty list.
- lst[1 : 4] = [ ] → it removes sliced elements from the list
- print(lst) → [‘a’, ‘f’]

List Methods

Examples:

- lst = [‘a’, ‘b’, ‘c’, ‘a’, ‘f’]
- print(len(lst)) → 5
- lst = [[‘a’,’b’,’c’] , [‘p’,’q’,’r]]
- print(len(lst)) → 2
- print(len(lst[0])) → 3
- range(10) → range (0, 10) #this function returns range object.
- list(range(5)) → [0,1,2,3,4]
- list(range(2, 5)) → [2,3,4]
- list(range(0, 101, 10)) → [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
- names = [‘Peter’, ‘Tom’, ‘Jack’]
- ages = [55, 20, 34]
- names_ages = zip(names, ages) → <zip object at 0x0000021BB765FE40> # zip object
- list(names_ages) → [('peter’, 55), ('tom', 20), ('jack’, 34)] # internally as tuple

List Methods , Membership keywords

Examples:

- lst = [‘a’, ‘b’, ‘c’, ‘a’, ‘f’]
- lst.index(‘a’) → 0 # stipulates the first occurrence of value.
- lst.count(‘a’) → 2

- list(‘hello’) → [‘h’, ‘e’, ‘l’, ‘l’, ‘o’]
- list(‘hello’, ‘bye’) → [‘h’, ‘e’, ‘l’, ‘l’, ‘o’, ‘,’, ‘b’, ‘y’,’e’]
- colours = ‘red, blue, green’
- lst_colours = colours.split(‘,’) → [‘red’, ‘blue’, ‘green’] # return list with Delimiter ’,’
- r, b, g = lst_colours → print(r) : red, print(b) : blue and print(g) : green

- sentence = ‘I love learning Python’
- lst_sentence = sentence.split(‘ ‘) # splitting based on space ‘ ‘ , returning us list of ele.
- print(lst_sentence) → [‘I', 'love', 'learning', ‘Python’]
- ‘ ‘.join(lst_sentence) → ‘I love learning Python’ # it recreates sentence based on ‘ ‘ space
- colours_list = [‘red’ , ‘blue’, ‘green’]
- ‘ and ‘.join(colours_list) → ‘red and blue and green’

List Membership

Membership:

- courses = [‘c++’, ‘c#’, ‘mvc’ ,’mvc core’,’ javascript’, ‘react’]
- print(‘python’ in courses) → False
- print(‘python’ not in courses) → True

List Summary


```
my_list = ['one', 'two', 'three']
```

.append()	.remove()	.index()	del
.extend()	.pop()	.count()	in / not in
.insert()	.clear()	.reverse()	list()
		.sort()	len()
			zip()

Dictionary

Syntax:

My_dict = { ‘Key’ , ‘Value’ }

my_dict = {‘custoered’ , }