**Python Lists**

are just like dynamically sized arrays, declared in other languages (vector in C++ and ArrayList in Java). In simple language, a list is a collection of things, enclosed in [ ] and separated by commas.

*The list is a sequence data type which is used to store the collection of data.*[*Tuples*](https://www.geeksforgeeks.org/python-tuples/)*and*[*String*](https://www.geeksforgeeks.org/python-strings/)*are other types of sequence data types.*

**Example of list in Python**

Here we are creating Python **List** using [].

* Python3

|  |
| --- |
| Var = ["Geeks", "for", "Geeks"]  print(Var) |

**Output:**

["Geeks", "for", "Geeks"]

Lists are the simplest containers that are an integral part of the Python language. Lists need not be homogeneous always which makes it the most powerful tool in [Python](https://www.geeksforgeeks.org/python-programming-language/). A single list may contain DataTypes like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation.

**Creating a List in Python**

Lists in Python can be created by just placing the sequence inside the square brackets[]. Unlike [Sets](https://www.geeksforgeeks.org/python-sets/), a list doesn’t need a built-in function for its creation of a list.

***Note:****Unlike Sets, the list may contain mutable elements.*

**Example 1: Creating a list in Python**

* Python3

|  |
| --- |
| # Python program to demonstrate  # Creation of List    # Creating a List  List = []  print("Blank List: ")  print(List)    # Creating a List of numbers  List = [10, 20, 14]  print("\nList of numbers: ")  print(List)    # Creating a List of strings and accessing  # using index  List = ["Geeks", "For", "Geeks"]  print("\nList Items: ")  print(List[0])  print(List[2]) |

**Output**

Blank List:

[]

List of numbers:

[10, 20, 14]

List Items:

Geeks

Geeks

**Complexities for Creating Lists**

**Time Complexity:**O(1)

**Space Complexity:**O(n)

**Example 2:  Creating a list with multiple distinct or duplicate elements**

A list may contain duplicate values with their distinct positions and hence, multiple distinct or duplicate values can be passed as a sequence at the time of list creation.

* Python3

|  |
| --- |
| # Creating a List with  # the use of Numbers  # (Having duplicate values)  List = [1, 2, 4, 4, 3, 3, 3, 6, 5]  print("\nList with the use of Numbers: ")  print(List)    # Creating a List with  # mixed type of values  # (Having numbers and strings)  List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']  print("\nList with the use of Mixed Values: ")  print(List) |

**Output**

List with the use of Numbers:

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

List with the use of Mixed Values:

[1, 2, 'Geeks', 4, 'For', 6, 'Geeks']

**Accessing elements from the List**

In order to access the list items refer to the index number. Use the index operator [ ] to access an item in a list. The index must be an integer. Nested lists are accessed using nested indexing.

**Example 1: Accessing elements from list**

* Python3

|  |
| --- |
| # Python program to demonstrate  # accessing of element from list    # Creating a List with  # the use of multiple values  List = ["Geeks", "For", "Geeks"]    # accessing a element from the  # list using index number  print("Accessing a element from the list")  print(List[0])  print(List[2]) |

**Output**

Accessing a element from the list

Geeks

Geeks

**Example 2: Accessing elements from a multi-dimensional list**

* Python3

|  |
| --- |
| # Creating a Multi-Dimensional List  # (By Nesting a list inside a List)  List = [['Geeks', 'For'], ['Geeks']]    # accessing an element from the  # Multi-Dimensional List using  # index number  print("Accessing a element from a Multi-Dimensional list")  print(List[0][1])  print(List[1][0]) |

**Output**

Accessing a element from a Multi-Dimensional list

For

Geeks

**Negative indexing**

In Python, negative sequence indexes represent positions from the end of the array. Instead of having to compute the offset as in List[len(List)-3], it is enough to just write List[-3]. Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second-last item, etc.

* Python3

|  |
| --- |
| List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']    # accessing an element using  # negative indexing  print("Accessing element using negative indexing")    # print the last element of list  print(List[-1])    # print the third last element of list  print(List[-3]) |

**Output**

Accessing element using negative indexing

Geeks

For

**Complexities for Accessing elements in a Lists:**

**Time Complexity:** O(1)

**Space Complexity:**O(1)

**Getting the size of Python list**

Python[len()](https://www.geeksforgeeks.org/python-ways-to-find-length-of-list/) is used to get the length of the list.

* Python3

|  |
| --- |
| # Creating a List  List1 = []  print(len(List1))    # Creating a List of numbers  List2 = [10, 20, 14]  print(len(List2)) |

**Output**

0

3

**Taking Input of a Python List**

We can take the input of a list of elements as string, integer, float, etc. But the default one is a string.

**Example 1:**

* Python3

|  |
| --- |
| # Python program to take space  # separated input as a string  # split and store it to a list  # and print the string list    # input the list as string  string = input("Enter elements (Space-Separated): ")    # split the strings and store it to a list  lst = string.split()  print('The list is:', lst)   # printing the list |

**Output:**

Enter elements: GEEKS FOR GEEKS

The list is: ['GEEKS', 'FOR', 'GEEKS']

**Example 2:**

* Python

|  |
| --- |
| # input size of the list  n = int(input("Enter the size of list : "))  # store integrs in a list using map,  # split and strip functions  lst = list(map(int, input("Enter the integer\  elements:").strip().split()))[:n]    # printing the list  print('The list is:', lst) |

**Output:**

Enter the size of list : 4

Enter the integer elements: 6 3 9 10

The list is: [6, 3, 9, 10]

To know more see [this](https://www.geeksforgeeks.org/python-get-a-list-as-input-from-user/).

**Adding Elements to a Python List**

**Method 1: Using append() method**

Elements can be added to the List by using the built-in [**append()**](https://www.geeksforgeeks.org/list-methods-python/) function. Only one element at a time can be added to the list by using the append() method, for the addition of multiple elements with the append() method, loops are used. Tuples can also be added to the list with the use of the append method because tuples are immutable. Unlike Sets, Lists can also be added to the existing list with the use of the append() method.

* Python3

|  |
| --- |
| # Python program to demonstrate  # Addition of elements in a List    # Creating a List  List = []  print("Initial blank List: ")  print(List)    # Addition of Elements  # in the List  List.append(1)  List.append(2)  List.append(4)  print("\nList after Addition of Three elements: ")  print(List)    # Adding elements to the List  # using Iterator  for i in range(1, 4):      List.append(i)  print("\nList after Addition of elements from 1-3: ")  print(List)    # Adding Tuples to the List  List.append((5, 6))  print("\nList after Addition of a Tuple: ")  print(List)    # Addition of List to a List  List2 = ['For', 'Geeks']  List.append(List2)  print("\nList after Addition of a List: ")  print(List) |

**Output**

Initial blank List:

[]

List after Addition of Three elements:

[1, 2, 4]

List after Addition of elements from 1-3:

[1, 2, 4, 1, 2, 3]

List after Addition of a Tuple:

[1, 2, 4, 1, 2, 3, (5, 6)]

List after Addition of a List:

[1, 2, 4, 1, 2, 3, (5, 6), ['For', 'Geeks']]

**Complexities for Adding elements in a Lists(append() method):**

**Time Complexity:** O(1)

S**pace Complexity:** O(1)

**Method 2: Using insert() method**

append() method only works for the addition of elements at the end of the List, for the addition of elements at the desired position, [insert()](https://www.geeksforgeeks.org/python-list-insert/)method is used. Unlike append() which takes only one argument, the insert() method requires two arguments(position, value).

* Python3

|  |
| --- |
| # Python program to demonstrate  # Addition of elements in a List    # Creating a List  List = [1,2,3,4]  print("Initial List: ")  print(List)    # Addition of Element at  # specific Position  # (using Insert Method)  List.insert(3, 12)  List.insert(0, 'Geeks')  print("\nList after performing Insert Operation: ")  print(List) |

**Output**

Initial List:

[1, 2, 3, 4]

List after performing Insert Operation:

['Geeks', 1, 2, 3, 12, 4]

**Complexities for Adding elements in a Lists(insert() method):**

**Time Complexity:** O(n)

**Space Complexity:** O(1)

**Method 3: Using extend() method**

Other than append() and insert() methods, there’s one more method for the Addition of elements, [**extend()**](https://www.geeksforgeeks.org/append-extend-python/), this method is used to add multiple elements at the same time at the end of the list.

***Note:***[*append() and extend()*](https://www.geeksforgeeks.org/append-extend-python/)*methods can only add elements at the end.*

* Python3

|  |
| --- |
| # Python program to demonstrate  # Addition of elements in a List    # Creating a List  List = [1, 2, 3, 4]  print("Initial List: ")  print(List)    # Addition of multiple elements  # to the List at the end  # (using Extend Method)  List.extend([8, 'Geeks', 'Always'])  print("\nList after performing Extend Operation: ")  print(List) |

**Output**

Initial List:

[1, 2, 3, 4]

List after performing Extend Operation:

[1, 2, 3, 4, 8, 'Geeks', 'Always']

**Complexities for Adding elements in a Lists(extend() method):**

**Time Complexity:** O(n)

**Space Complexity:** O(1)

**Reversing a List**

A list can be reversed by using the [reverse() method in Python](https://www.geeksforgeeks.org/python-list-reverse/).

* Python3

|  |
| --- |
| # Reversing a list  mylist = [1, 2, 3, 4, 5, 'Geek', 'Python']  mylist.reverse()  print(mylist) |

**Output**

['Python', 'Geek', 5, 4, 3, 2, 1]

**Removing Elements from the List**

**Method 1: Using remove() method**

Elements can be removed from the List by using the built-in [**remove()**](https://www.geeksforgeeks.org/python-list-remove/) function but an Error arises if the element doesn’t exist in the list. Remove() method only removes one element at a time, to remove a range of elements, the iterator is used. The remove() method removes the specified item.

***Note:****Remove method in List will only remove the first occurrence of the searched element.*

**Example 1:**

* Python3

|  |
| --- |
| # Python program to demonstrate  # Removal of elements in a List    # Creating a List  List = [1, 2, 3, 4, 5, 6,          7, 8, 9, 10, 11, 12]  print("Initial List: ")  print(List)    # Removing elements from List  # using Remove() method  List.remove(5)  List.remove(6)  print("\nList after Removal of two elements: ")  print(List) |

**Output**

Initial List:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

List after Removal of two elements:

[1, 2, 3, 4, 7, 8, 9, 10, 11, 12]

**Example 2:**

* Python3

|  |
| --- |
| # Creating a List  List = [1, 2, 3, 4, 5, 6,          7, 8, 9, 10, 11, 12]  # Removing elements from List  # using iterator method  for i in range(1, 5):      List.remove(i)  print("\nList after Removing a range of elements: ")  print(List) |

**Output**

List after Removing a range of elements:

[5, 6, 7, 8, 9, 10, 11, 12]

**Complexities for Deleting elements in a Lists(remove() method):**

**Time Complexity:**O(n)

**Space Complexity:** O(1)

**Method 2: Using pop() method**

[pop() function](https://www.geeksforgeeks.org/python-list-pop/) can also be used to remove and return an element from the list, but by default it removes only the last element of the list, to remove an element from a specific position of the List, the index of the element is passed as an argument to the pop() method.

* Python3

|  |
| --- |
| List = [1, 2, 3, 4, 5]    # Removing element from the  # Set using the pop() method  List.pop()  print("\nList after popping an element: ")  print(List)    # Removing element at a  # specific location from the  # Set using the pop() method  List.pop(2)  print("\nList after popping a specific element: ")  print(List) |

**Output**

List after popping an element:

[1, 2, 3, 4]

List after popping a specific element:

[1, 2, 4]

**Complexities for Deleting elements in a Lists(pop() method):**

**Time Complexity:** O(1)/O(n) (O(1) for removing the last element, O(n) for removing the first and middle elements)

**Space Complexity:** O(1)

**Slicing of a List**

We can get substrings and sublists using a slice. In Python List, there are multiple ways to print the whole list with all the elements, but to print a specific range of elements from the list, we use the [Slice operation](https://www.geeksforgeeks.org/python-list-comprehension-and-slicing/).

Slice operation is performed on Lists with the use of a colon(:).

**To print elements from beginning to a range use:**

*[: Index]*

To print elements from end-use:

*[:-Index]*

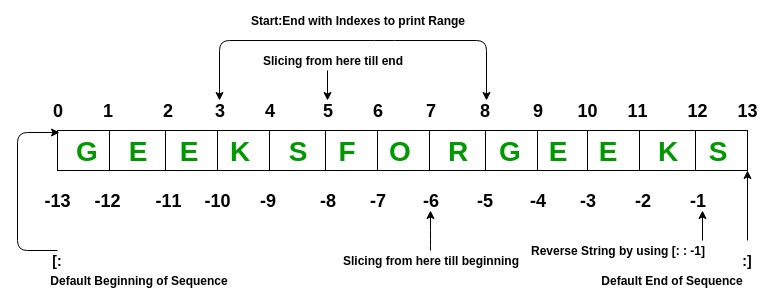
To print elements from a specific Index till the end use

*[Index:]*

To print the whole list in reverse order, use

*[::-1]*

**Note –**To print elements of List from rear-end, use Negative Indexes.



**UNDERSTANDING SLICING OF LISTS:**

* pr[0] accesses the first item, 2.
* pr[-4] accesses the fourth item from the end, 5.
* pr[2:] accesses [5, 7, 11, 13], a list of items from third to last.
* pr[:4] accesses [2, 3, 5, 7], a list of items from first to fourth.
* pr[2:4] accesses [5, 7], a list of items from third to fifth.
* pr[1::2] accesses [3, 7, 13], alternate items, starting from the second item.
* Python3

|  |
| --- |
| # Python program to demonstrate  # Removal of elements in a List    # Creating a List  List = ['G', 'E', 'E', 'K', 'S', 'F',          'O', 'R', 'G', 'E', 'E', 'K', 'S']  print("Initial List: ")  print(List)    # Print elements of a range  # using Slice operation  Sliced\_List = List[3:8]  print("\nSlicing elements in a range 3-8: ")  print(Sliced\_List)    # Print elements from a  # pre-defined point to end  Sliced\_List = List[5:]  print("\nElements sliced from 5th "        "element till the end: ")  print(Sliced\_List)    # Printing elements from  # beginning till end  Sliced\_List = List[:]  print("\nPrinting all elements using slice operation: ")  print(Sliced\_List) |

**Output**

Initial List:

['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']

Slicing elements in a range 3-8:

['K', 'S', 'F', 'O', 'R']

Elements sliced from 5th element till the end:

['F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']

Printing all elements using slice operation:

['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']

**Negative index List slicing**

* Python3

|  |
| --- |
| # Creating a List  List = ['G', 'E', 'E', 'K', 'S', 'F',          'O', 'R', 'G', 'E', 'E', 'K', 'S']  print("Initial List: ")  print(List)    # Print elements from beginning  # to a pre-defined point using Slice  Sliced\_List = List[:-6]  print("\nElements sliced till 6th element from last: ")  print(Sliced\_List)    # Print elements of a range  # using negative index List slicing  Sliced\_List = List[-6:-1]  print("\nElements sliced from index -6 to -1")  print(Sliced\_List)    # Printing elements in reverse  # using Slice operation  Sliced\_List = List[::-1]  print("\nPrinting List in reverse: ")  print(Sliced\_List) |

**Output**

Initial List:

['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']

Elements sliced till 6th element from last:

['G', 'E', 'E', 'K', 'S', 'F', 'O']

Elements sliced from index -6 to -1

['R', 'G', 'E', 'E', 'K']

Printing List in reverse:

['S', 'K', 'E', 'E', 'G', 'R', 'O', 'F', 'S', 'K', 'E', 'E', 'G']

**List Comprehension**

[**Python List comprehensions**](https://www.geeksforgeeks.org/python-list-comprehension/) are used for creating new lists from other iterables like tuples, strings, arrays, lists, etc. A list comprehension consists of brackets containing the expression, which is executed for each element along with the for loop to iterate over each element.

**Syntax:**

*newList = [ expression(element) for element in oldList if condition ]*

**Example:**

* Python3

|  |
| --- |
| # Python program to demonstrate list  # comprehension in Python    # below list contains square of all  # odd numbers from range 1 to 10  odd\_square = [x \*\* 2 for x in range(1, 11) if x % 2 == 1]  print(odd\_square) |

**Output**

[1, 9, 25, 49, 81]

For better understanding, the above code is similar to as follows:

* Python3

|  |
| --- |
| # for understanding, above generation is same as,  odd\_square = []    for x in range(1, 11):      if x % 2 == 1:          odd\_square.append(x\*\*2)    print(odd\_square) |

**Output**

[1, 9, 25, 49, 81]

Refer to the below articles to get detailed information about List Comprehension.

* [Python List Comprehension and Slicing](https://www.geeksforgeeks.org/python-list-comprehension-and-slicing/)
* [Nested List Comprehensions in Python](https://www.geeksforgeeks.org/nested-list-comprehensions-in-python/)
* [List comprehension and ord() in Python](https://www.geeksforgeeks.org/list-comprehension-and-ord-in-python-to-remove-all-characters-other-than-alphabets/)

**Basic Example on Python List**

* [Python program to interchange first and last elements in a list](https://www.geeksforgeeks.org/python-program-to-interchange-first-and-last-elements-in-a-list/)
* [Python program to swap two elements in a list](https://www.geeksforgeeks.org/python-program-to-swap-two-elements-in-a-list/)
* [Python – Swap elements in String list](https://www.geeksforgeeks.org/python-swap-elements-in-string-list/)
* [Python | Ways to find length of list](https://www.geeksforgeeks.org/python-ways-to-find-length-of-list/)
* [Maximum of two numbers in Python](https://www.geeksforgeeks.org/maximum-of-two-numbers-in-python/?ref=leftbar-rightbar)
* [Minimum of two numbers in Python](https://www.geeksforgeeks.org/minimum-of-two-numbers-in-python/)

To Practice the basic list operation, please read this article – [Python List of program](https://www.geeksforgeeks.org/python-list-exercise/)

**List Methods**

| Function | Description |
| --- | --- |
| [Append()](https://www.geeksforgeeks.org/append-extend-python/) | Add an element to the end of the list |
| [Extend()](https://www.geeksforgeeks.org/append-extend-python/) | Add all elements of a list to another list |
| [Insert()](https://www.geeksforgeeks.org/list-methods-in-python-set-2-del-remove-sort-insert-pop-extend/) | Insert an item at the defined index |
| [Remove()](https://www.geeksforgeeks.org/list-methods-in-python-set-2-del-remove-sort-insert-pop-extend/) | Removes an item from the list |
| [Clear()](https://www.geeksforgeeks.org/list-methods-in-python-set-2-del-remove-sort-insert-pop-extend/) | Removes all items from the list |
| [Index()](https://www.geeksforgeeks.org/python-list-index/) | Returns the index of the first matched item |
| [Count()](https://www.geeksforgeeks.org/python-list-function-count/) | Returns the count of the number of items passed as an argument |
| [Sort()](https://www.geeksforgeeks.org/sort-in-python/) | Sort items in a list in ascending order |
| [Reverse()](https://www.geeksforgeeks.org/list-methods-in-python-set-2-del-remove-sort-insert-pop-extend/) | Reverse the order of items in the list |
| [copy()](https://www.geeksforgeeks.org/python-list-copy-method/) | Returns a copy of the list |

To know more refer to this article – [Python List methods](https://www.geeksforgeeks.org/list-methods-python/)

The operations mentioned above modify the list Itself.

**Built-in functions with List**

| Function | Description |
| --- | --- |
| [reduce()](https://www.geeksforgeeks.org/reduce-in-python/) | apply a particular function passed in its argument to all of the list elements stores the intermediate result and only returns the final summation value |
| [sum()](https://www.geeksforgeeks.org/sum-function-python/) | Sums up the numbers in the list |
| [ord()](https://www.geeksforgeeks.org/ord-function-python/) | Returns an integer representing the Unicode code point of the given Unicode character |
| [cmp()](https://www.geeksforgeeks.org/python-2-number-cmplist-method/) | This function returns 1 if the first list is “greater” than the second list |
| [max()](https://www.geeksforgeeks.org/python-max-function/) | return maximum element of a given list |
| [min()](https://www.geeksforgeeks.org/python-min-function/) | return minimum element of a given list |
| [all()](https://www.geeksforgeeks.org/any-all-in-python/) | Returns true if all element is true or if the list is empty |
| [any()](https://www.geeksforgeeks.org/any-all-in-python/) | return true if any element of the list is true. if the list is empty, return false |
| len() | Returns length of the list or size of the list |
| [enumerate()](https://www.geeksforgeeks.org/enumerate-in-python/) | Returns enumerate object of the list |
| accumulate() | apply a particular function passed in its argument to all of the list elements returns a list containing the intermediate results |
| [filter()](https://www.geeksforgeeks.org/filter-in-python/) | tests if each element of a list is true or not |
| [map()](https://www.geeksforgeeks.org/python-map-function/) | returns a list of the results after applying the given function to each item of a given iterable |
| [lambda()](https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/) | This function can have any number of arguments but only one expression, which is evaluated and returned. |