**PYTHON WEEK 3 DAY 2 HOW TO USE LISTS**

**How to create a list?**

In Python programming, a list is created by placing all the items (elements) inside square brackets [], separated by commas.

It can have any number of items and they may be of different types (integer, float, string etc.).

# empty list

my\_list = []

# list of integers

my\_list = [1, 2, 3]

# list with mixed data types

my\_list = [1, "Hello", 3.4]

A list can also have another list as an item. This is called a nested list.

# nested list

my\_list = ["mouse", [8, 4, 6], ['a']]

# Access List Elements

There are various ways in which we can access the elements of a list.

### List Index

We can use the index operator [] to access an item in a list. In Python, indices start at 0. So, a list having 5 elements will have an index from 0 to 4.

Trying to access indexes other than these will raise an IndexError. The index must be an integer. We can't use float or other types, this will result in TypeError.

Nested lists are accessed using nested indexing.

# List indexing

my\_list = ['p', 'r', 'o', 'b', 'e']

# Output: p

print(my\_list[0])

# Output: o

print(my\_list[2])

# Output: e

print(my\_list[4])

# Nested List

n\_list = ["Happy", [2, 0, 1, 5]]

# Nested indexing

print(n\_list[0][1])

print(n\_list[1][3])

# Error! Only integer can be used for indexing

print(my\_list[4.0])

**Output**

p

o

e

a

5

Traceback (most recent call last):

  File "<string>", line 21, in <module>

TypeError: list indices must be integers or slices, not float

**Negative indexing**

Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

# Negative indexing in lists

my\_list = ['p','r','o','b','e']

print(my\_list[-1])

print(my\_list[-5])

When we run the above program, we will get the following output:

e

p

# ow to slice lists in Python?

We can access a range of items in a list by using the slicing operator :(colon).

# List slicing in Python

my\_list = ['p','r','o','g','r','a','m','i','z']

# elements 3rd to 5th

print(my\_list[2:5])

# elements beginning to 4th

print(my\_list[:-5])

# elements 6th to end

print(my\_list[5:])

# elements beginning to end

print(my\_list[:])

**Output**

['o', 'g', 'r']

['p', 'r', 'o', 'g']

['a', 'm', 'i', 'z']

['p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z']

Slicing can be best visualized by considering the index to be between the elements as shown below. So if we want to access a range, we need two indices that will slice that portion from the list.

Element Slicing from a list in Python

# Add/Change List Elements

Lists are mutable, meaning their elements can be changed unlike [string](https://www.programiz.com/python-programming/string) or [tuple](https://www.programiz.com/python-programming/tuple).

We can use the assignment operator = to change an item or a range of items.

# Correcting mistake values in a list

odd = [2, 4, 6, 8]

# change the 1st item

odd[0] = 1

print(odd)

# change 2nd to 4th items

odd[1:4] = [3, 5, 7]

print(odd)

**Output**

[1, 4, 6, 8]

[1, 3, 5, 7]

We can add one item to a list using the append() method or add several items using extend() method.

# Appending and Extending lists in Python

odd = [1, 3, 5]

odd.append(7)

print(odd)

odd.extend([9, 11, 13])

print(odd)

**Output**

[1, 3, 5, 7]

[1, 3, 5, 7, 9, 11, 13]

We can also use + operator to combine two lists. This is also called concatenation.

The \* operator repeats a list for the given number of times.

# Concatenating and repeating lists

odd = [1, 3, 5]

print(odd + [9, 7, 5])

print(["re"] \* 3)

**Output**

[1, 3, 5, 9, 7, 5]

['re', 're', 're']

Furthermore, we can insert one item at a desired location by using the method insert() or insert multiple items by squeezing it into an empty slice of a list.

# Demonstration of list insert() method

odd = [1, 9]

odd.insert(1,3)

print(odd)

odd[2:2] = [5, 7]

print(odd)

**Output**

[1, 3, 9]

[1, 3, 5, 7, 9]

# Delete/Remove List Elements

We can delete one or more items from a list using the keyword del. It can even delete the list entirely.

# Deleting list items

my\_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']

# delete one item

del my\_list[2]

print(my\_list)

# delete multiple items

del my\_list[1:5]

print(my\_list)

# delete entire list

del my\_list

# Error: List not defined

print(my\_list)

**Output**

['p', 'r', 'b', 'l', 'e', 'm']

['p', 'm']

Traceback (most recent call last):

  File "<string>", line 18, in <module>

NameError: name 'my\_list' is not defined

We can use remove() method to remove the given item or pop() method to remove an item at the given index.

The pop() method removes and returns the last item if the index is not provided. This helps us implement lists as stacks (first in, last out data structure).

We can also use the clear() method to empty a list.

my\_list = ['p','r','o','b','l','e','m']

my\_list.remove('p')

# Output: ['r', 'o', 'b', 'l', 'e', 'm']

print(my\_list)

# Output: 'o'

print(my\_list.pop(1))

# Output: ['r', 'b', 'l', 'e', 'm']

print(my\_list)

# Output: 'm'

print(my\_list.pop())

# Output: ['r', 'b', 'l', 'e']

print(my\_list)

my\_list.clear()

# Output: []

print(my\_list)

**Output**

['r', 'o', 'b', 'l', 'e', 'm']

o

['r', 'b', 'l', 'e', 'm']

m

['r', 'b', 'l', 'e']

[]

Finally, we can also delete items in a list by assigning an empty list to a slice of elements.

>>> my\_list = ['p','r','o','b','l','e','m']

>>> my\_list[2:3] = []

>>> my\_list

['p', 'r', 'b', 'l', 'e', 'm']

>>> my\_list[2:5] = []

>>> my\_list

['p', 'r', 'm']

# Python List Methods

Methods that are available with list objects in Python programming are tabulated below.

They are accessed as list.method(). Some of the methods have already been used above.

|  |
| --- |
| [Python List Methods](https://www.programiz.com/python-programming/methods/list) |
| [**append() -**](https://www.programiz.com/python-programming/methods/list/append) [Add an element to the end of the list](https://www.programiz.com/python-programming/methods/list/append) |
| [**extend()**](https://www.programiz.com/python-programming/methods/list/extend) [-](https://www.programiz.com/python-programming/methods/list/extend) [Add all elements of a list to the another list](https://www.programiz.com/python-programming/methods/list/extend) |
| [**insert()**](https://www.programiz.com/python-programming/methods/list/insert) [-](https://www.programiz.com/python-programming/methods/list/insert) [Insert an item at the defined index](https://www.programiz.com/python-programming/methods/list/insert) |
| [**remove()**](https://www.programiz.com/python-programming/methods/list/remove) [-](https://www.programiz.com/python-programming/methods/list/remove) [Removes an item from the list](https://www.programiz.com/python-programming/methods/list/remove) |
| [**pop()**](https://www.programiz.com/python-programming/methods/list/pop) [-](https://www.programiz.com/python-programming/methods/list/pop) [Removes and returns an element at the given index](https://www.programiz.com/python-programming/methods/list/pop) |
| [**clear()**](https://www.programiz.com/python-programming/methods/list/clear) [- Removes all items from the list](https://www.programiz.com/python-programming/methods/list/clear) |
| [**index()**](https://www.programiz.com/python-programming/methods/list/index) [- Returns the index of the first matched item](https://www.programiz.com/python-programming/methods/list/index) |
| [**count()**](https://www.programiz.com/python-programming/methods/list/count) [- Returns the count of the number of items passed as an argument](https://www.programiz.com/python-programming/methods/list/count) |
| [**sort()**](https://www.programiz.com/python-programming/methods/list/sort) [- Sort items in a list in ascending order](https://www.programiz.com/python-programming/methods/list/sort) |
| [**reverse()**](https://www.programiz.com/python-programming/methods/list/reverse) [- Reverse the order of items in the list](https://www.programiz.com/python-programming/methods/list/reverse) |
| [**copy()**](https://www.programiz.com/python-programming/methods/list/copy) [- Returns a shallow copy of the list](https://www.programiz.com/python-programming/methods/list/copy) |

# Some examples of Python list methods:

# Python list methods

my\_list = [3, 8, 1, 6, 0, 8, 4]

# Output: 1

print(my\_list.index(8))

# Output: 2

print(my\_list.count(8))

my\_list.sort()

# Output: [0, 1, 3, 4, 6, 8, 8]

print(my\_list)

my\_list.reverse()

# Output: [8, 8, 6, 4, 3, 1, 0]

print(my\_list)

**Output**

1

2

[0, 1, 3, 4, 6, 8, 8]

[8, 8, 6, 4, 3, 1, 0]

# List Comprehension: Elegant way to create Lists

List comprehension is an elegant and concise way to create a new list from an existing list in Python.

A list comprehension consists of an expression followed by [for statement](https://www.programiz.com/python-programming/for-loop) inside square brackets.

Here is an example to make a list with each item being increasing power of 2.

pow2 = [2 \*\* x for x in range(10)]

print(pow2)

**Output**

[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]

This code is equivalent to:

pow2 = []

for x in range(10):

   pow2.append(2 \*\* x)

A list comprehension can optionally contain more for or [if statements](https://www.programiz.com/python-programming/if-elif-else). An optional if statement can filter out items for the new list. Here are some examples.

>>> pow2 = [2 \*\* x for x in range(10) if x > 5]

>>> pow2

[64, 128, 256, 512]

>>> odd = [x for x in range(20) if x % 2 == 1]

>>> odd

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

>>> [x+y for x in ['Python ','C '] for y in ['Language','Programming']]

['Python Language', 'Python Programming', 'C Language', 'C Programming']

## 

## 

# Other List Operations in Python

### List Membership Test

We can test if an item exists in a list or not, using the keyword in.

my\_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']

# Output: True

print('p' in my\_list)

# Output: False

print('a' in my\_list)

# Output: True

print('c' not in my\_list)

**Output**

True

False

True

# Iterating Through a List

Using a for loop we can iterate through each item in a list.

for fruit in ['apple','banana','mango']:

    print("I like",fruit)

**Output**

I like apple

I like banana

I like mango