

List

List is the same as array. We can store various values within the same variable name. Those values are organized by an index location. The index location starts from zero.

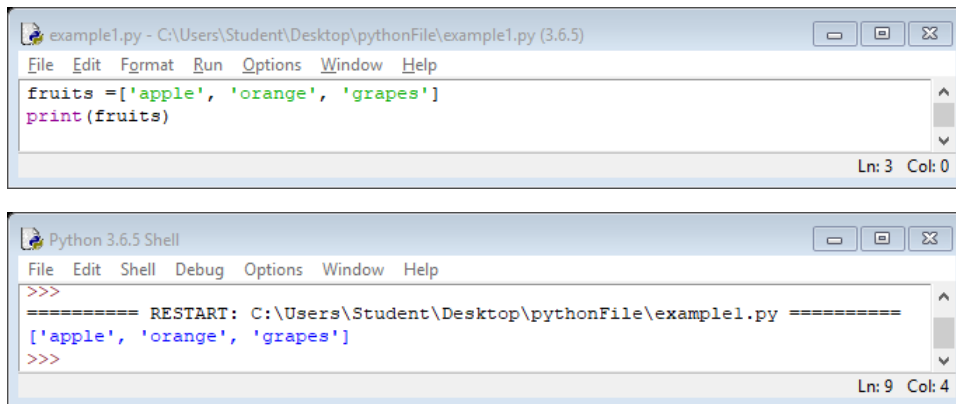
To create a list by using square brackets []

```
fruits=['apple', 'orange', 'grapes']
```

The lists *fruits* will be organized as the following:

fruits =	apple	orange	grapes
Index location	0	1	2

The Python code will look as the following:



The screenshot shows two windows from a Python IDE. The top window, titled 'example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)', contains the following code:

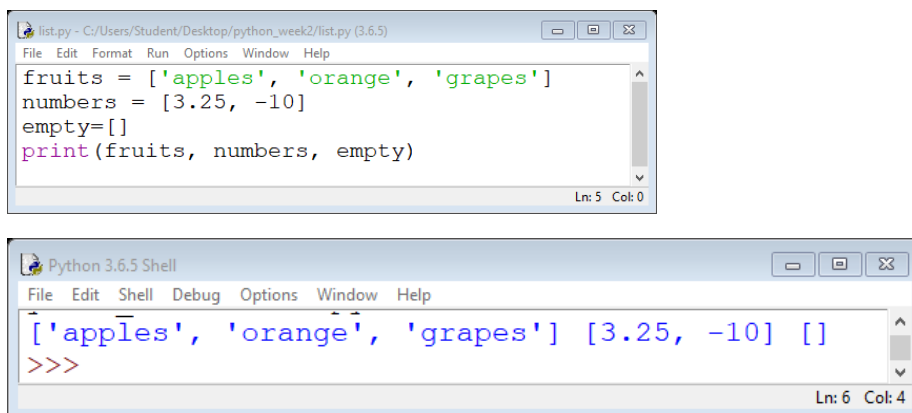
```
fruits=['apple', 'orange', 'grapes']
print(fruits)
```

The bottom window, titled 'Python 3.6.5 Shell', shows the execution output after pressing F5:

```
>>>
===== RESTART: C:\Users\Student\Desktop\pythonFile\example1.py =====
['apple', 'orange', 'grapes']
>>>
```

Empty list

A list that contains no elements is called an empty list; you can create one with empty brackets, []



The screenshot shows two windows from a Python IDE. The top window, titled 'list.py - C:\Users\Student\Desktop\python_week2\list.py (3.6.5)', contains the following code:

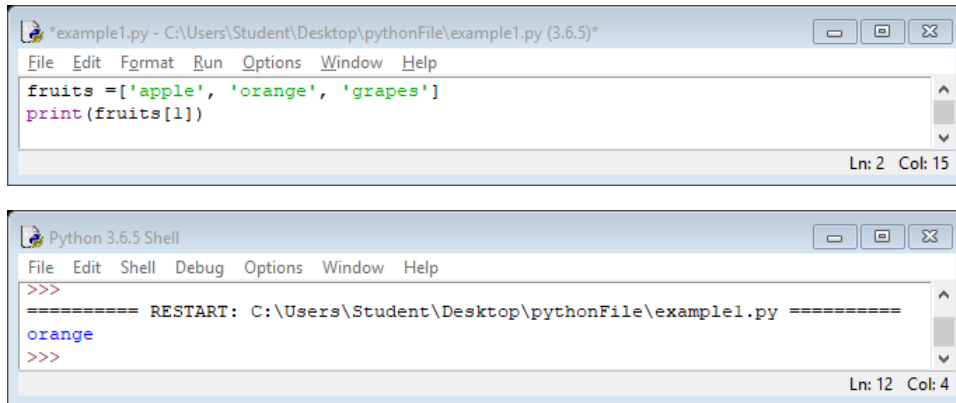
```
fruits = ['apples', 'orange', 'grapes']
numbers = [3.25, -10]
empty=[]
print(fruits, numbers, empty)
```

The bottom window, titled 'Python 3.6.5 Shell', shows the execution output after pressing F5:

```
>>>
['apples', 'orange', 'grapes'] [3.25, -10] []
>>>
```

Accessing to elements in the list

Creating a list takes a bit more typing than creating a string, but a list is more useful than a string because it can be manipulated. The syntax for accessing the elements of a list is the same as for accessing the characters of a string — the **bracket operator** `[]`. For example, we could print the second value in *fruits* list by entering its index position inside square brackets after the list name. Remember that the index starts from zero, therefore, if we want to print the second value, the index number should be 1 **`fruits[1]`**

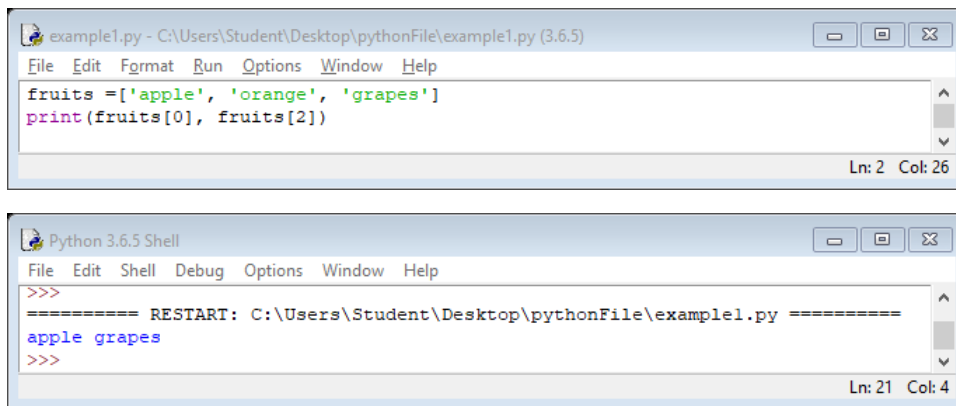


The screenshot shows two windows from a Python IDE. The top window, titled 'example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)*', contains the following code:

```
fruits = ['apple', 'orange', 'grapes']
print(fruits[1])
```

The bottom window, titled 'Python 3.6.5 Shell', shows the execution of the script. It displays a 'RESTART' message followed by the output 'orange'.

Also, if we want to print the first and third values of list *fruits*, we need to separate the index number with a comma:



The screenshot shows two windows from a Python IDE. The top window, titled 'example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)', contains the following code:

```
fruits = ['apple', 'orange', 'grapes']
print(fruits[0], fruits[2])
```

The bottom window, titled 'Python 3.6.5 Shell', shows the execution of the script. It displays a 'RESTART' message followed by the output 'apple grapes'.

Negative index

If an index has a negative value, it counts backward from the end of the list.

fruits =	apple	orange	grapes
Index location	-3	-2	-1

```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes']
print(fruits[-1])
Ln: 2 Col: 15
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
=====
grapes
>>>
Ln: 6 Col: 4
```

Range of indexes

You can specify a range of indexes by specifying where to start and where to end the range by using a colon operator. Remember that Python will not print the end index

Print from index 2
↓
fruits[2:4]
↑
List name

end index 4 but doesn't include value of index 4
↓

Example) Range of indexes: print list elements from index 2 up to index 4 (Exclusive)

```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes', 'cherry', 'kiwi', 'melon', 'mango']
print(fruits[2:4])
Ln: 2 Col: 16
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/Student/Desktop/python
week2/list.py =====
['grapes', 'cherry']
Ln: 9 Col: 4
```

Example) Negative range of indexes: print list elements from element -5 up to element -2

```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes', 'cherry', 'kiwi', 'melon', 'mango']
print(fruits[-5:-2])
Ln: 2 Col: 18
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
=====
esktop/python_week2/list.py =====
['grapes', 'cherry', 'kiwi']
>>>
Ln: 18 Col: 4
```

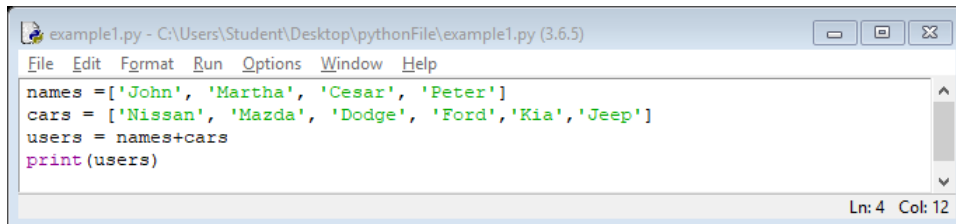
List operations

+ operator

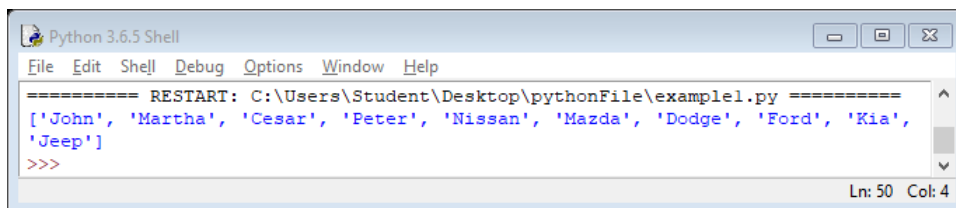
The + operator concatenates lists. For example, suppose we have two lists: names and cars:

```
names = ['John', 'Martha', 'Cesar', 'Peter']  
cars = ['Nissan', 'Mazda', 'Dodge', 'Ford', 'Kia', 'Jeep']
```

We can also add the two lists and set the result equal to another variable:

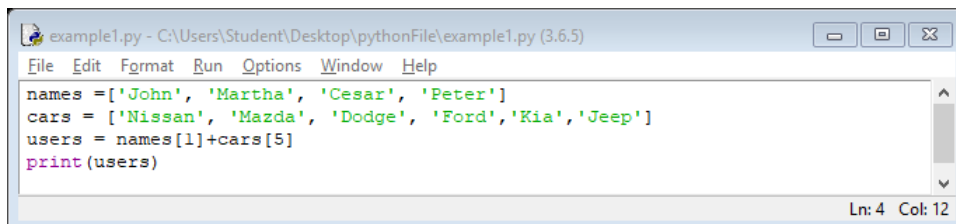


```
example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)  
File Edit Format Run Options Window Help  
names = ['John', 'Martha', 'Cesar', 'Peter']  
cars = ['Nissan', 'Mazda', 'Dodge', 'Ford', 'Kia', 'Jeep']  
users = names+cars  
print(users)  
Ln: 4 Col: 12
```

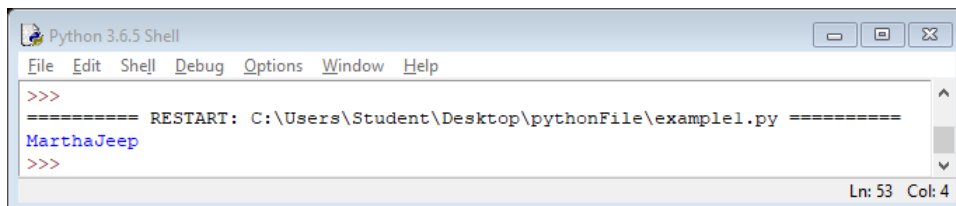


```
Python 3.6.5 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: C:\Users\Student\Desktop\pythonFile\example1.py =====  
['John', 'Martha', 'Cesar', 'Peter', 'Nissan', 'Mazda', 'Dodge', 'Ford', 'Kia',  
'Jeep']  
>>>
```

We can also add independent values from different lists. For example, if we want to join Martha with Jeep: users = names[1]+cars[5]



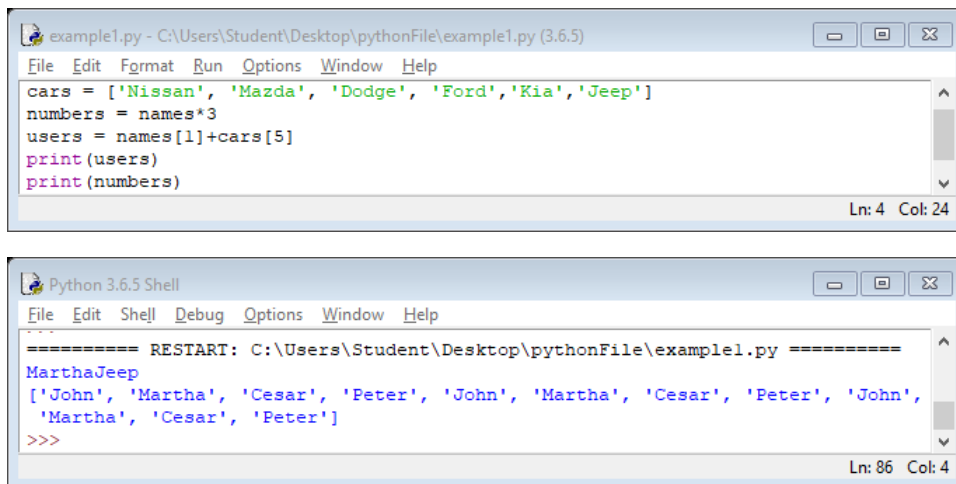
```
example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)  
File Edit Format Run Options Window Help  
names = ['John', 'Martha', 'Cesar', 'Peter']  
cars = ['Nissan', 'Mazda', 'Dodge', 'Ford', 'Kia', 'Jeep']  
users = names[1]+cars[5]  
print(users)  
Ln: 4 Col: 12
```



```
Python 3.6.5 Shell  
File Edit Shell Debug Options Window Help  
>>>  
===== RESTART: C:\Users\Student\Desktop\pythonFile\example1.py =====  
MarthaJeep  
>>>
```

* operator

Similarly, the * operator repeats a list a given number of times. In other words, we can also duplicate list values by using the multiplication sign *



The image shows two windows from a Python IDE. The top window, titled 'example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)', contains the following code:

```
cars = ['Nissan', 'Mazda', 'Dodge', 'Ford', 'Kia', 'Jeep']
numbers = names*3
users = names[1]+cars[5]
print(users)
print(numbers)
```

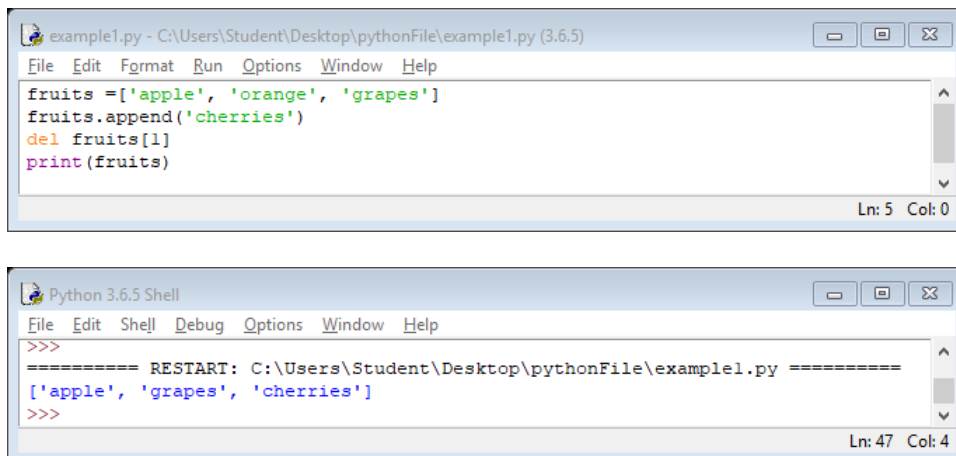
The bottom window, titled 'Python 3.6.5 Shell', shows the output of the script after a restart:

```
===== RESTART: C:\Users\Student\Desktop\pythonFile\example1.py =====
MarthaJeep
['John', 'Martha', 'Cesar', 'Peter', 'John', 'Martha', 'Cesar', 'Peter', 'John',
'Martha', 'Cesar', 'Peter']
>>>
```

List Methods

Removing Items from a list

To remove items from a list, use the *del* command (short for *delete*) followed by the list value. For example, to remove the second value from list *fruits*: `del fruits[1]`



The image shows two windows from a Python IDE. The top window, titled 'example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)', contains the following code:

```
fruits = ['apple', 'orange', 'grapes']
fruits.append('cherries')
del fruits[1]
print(fruits)
```

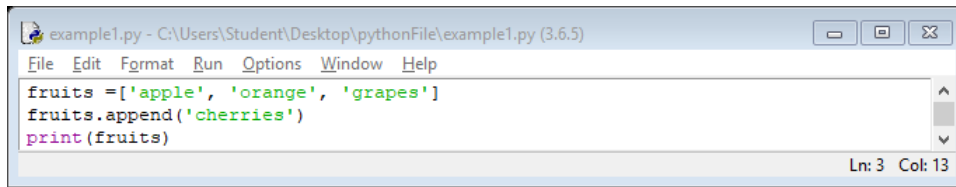
The bottom window, titled 'Python 3.6.5 Shell', shows the output of the script after a restart:

```
===== RESTART: C:\Users\Student\Desktop\pythonFile\example1.py =====
['apple', 'grapes', 'cherries']
>>>
```

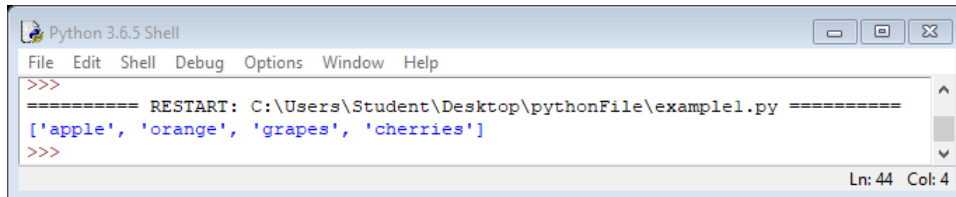
Adding items to a lists

To add items to a list, we use the Python function ***append()***. A function is a chunk of code that tells Python to do something. In this case, ***append()*** adds an item to the end of a list.

For example, to add *cherries* to the list *fruits*: `fruits.append('cherries')`



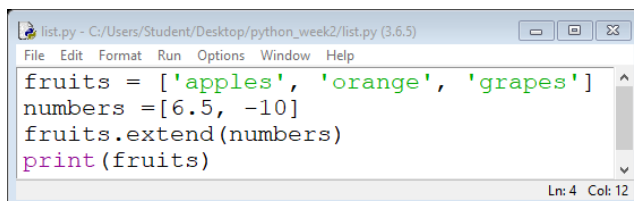
```
example1.py - C:\Users\Student\Desktop\pythonFile\example1.py (3.6.5)
File Edit Format Run Options Window Help
fruits=['apple', 'orange', 'grapes']
fruits.append('cherries')
print(fruits)
Ln: 3 Col: 13
```



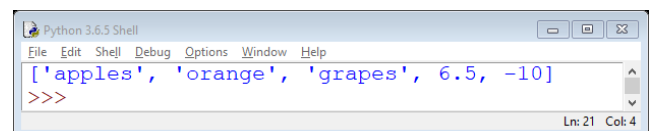
```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:\Users\Student\Desktop\pythonFile\example1.py =====
['apple', 'orange', 'grapes', 'cherries']
>>>
Ln: 44 Col: 4
```

Extend item in a list

You can use the **extend()** method, which purpose is to add elements from one list to another list:



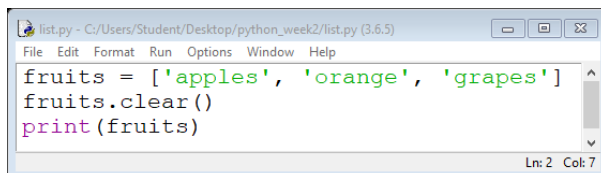
```
list.py - C:\Users\Student\Desktop\python_week2\list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes']
numbers = [6.5, -10]
fruits.extend(numbers)
print(fruits)
Ln: 4 Col: 12
```



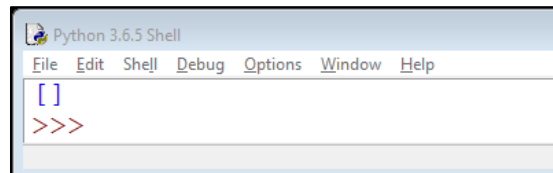
```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
['apples', 'orange', 'grapes', 6.5, -10]
>>>
Ln: 21 Col: 4
```

Clear elements from a list

Method **clear()** Removes all the elements from the list



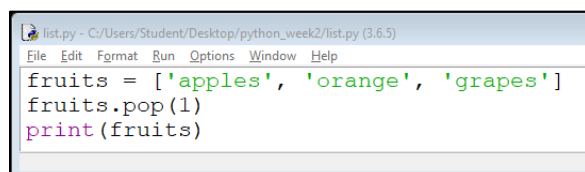
```
list.py - C:\Users\Student\Desktop\python_week2\list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes']
fruits.clear()
print(fruits)
Ln: 2 Col: 7
```



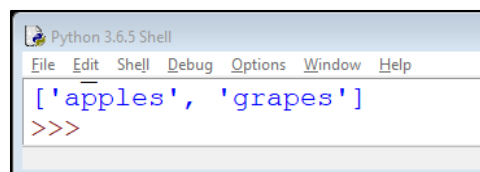
```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
[]
>>>
Ln: 21 Col: 4
```

Remove an element from the list with specific index

The **pop()** method removes the element at the specified position. **Method that returns a value**



```
list.py - C:\Users\Student\Desktop\python_week2\list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes']
fruits.pop(1)
print(fruits)
Ln: 2 Col: 7
```



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
['apples', 'grapes']
>>>
Ln: 21 Col: 4
```

Reverse a list

The **reverse()** method reverses the sorting order of the elements.

```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes']
fruits.reverse()
print(fruits)
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
['grapes', 'orange', 'apples']
>>>
```

Sort elements in an array

The **sort()** method sorts the list ascending by default. If it is string list, it sorts alphabetically

```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes']
fruits.sort()
print(fruits)
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
['apples', 'grapes', 'orange']
>>>
```

If it is number list, it sorts by increasing order.

```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
number=[2.8,2,-3,-5.8]
number.sort()
print(number)
Ln: 3 Col: 12
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
[-5.8, -3, 2, 2.8]
>>>
Ln: 70 Col: 0
```

copy a list

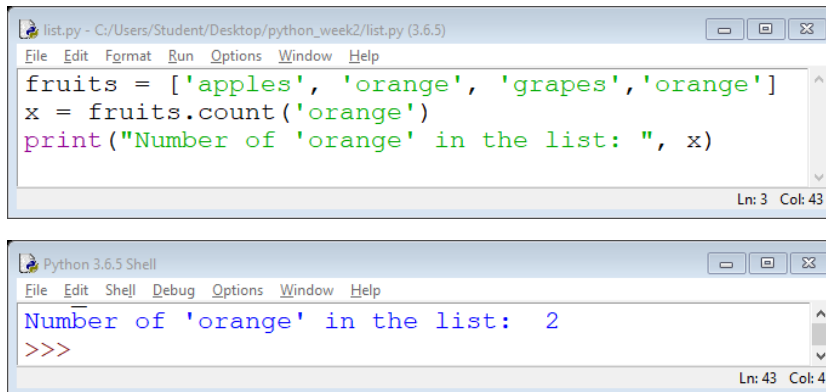
copy() returns a copy of the list

```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes']
copyFruits = fruits.copy()
print("fruits list: ", fruits)
print("copyFruits list: ", copyFruits)
Ln: 4 Col: 37
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
fruits list: ['apples', 'orange', 'grapes']
copyFruits list: ['apples', 'orange', 'grapes']
>>>
Ln: 40 Col: 4
```

Return number of a specific value

The `count()` method **returns** the number of elements with the specified value.

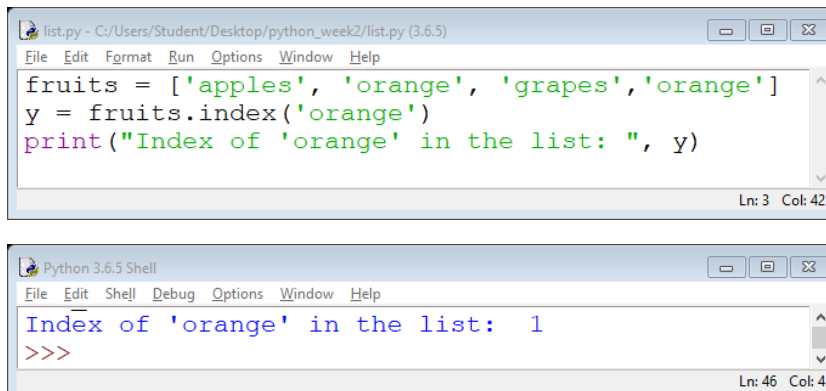


```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes', 'orange']
x = fruits.count('orange')
print("Number of 'orange' in the list: ", x)
Ln: 3 Col: 43
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Number of 'orange' in the list: 2
>>>
Ln: 43 Col: 4
```

Index of a value in the list

The `index()` method **returns** the position at the first occurrence of the specified value.



```
list.py - C:/Users/Student/Desktop/python_week2/list.py (3.6.5)
File Edit Format Run Options Window Help
fruits = ['apples', 'orange', 'grapes', 'orange']
y = fruits.index('orange')
print("Index of 'orange' in the list: ", y)
Ln: 3 Col: 42
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Index of 'orange' in the list: 1
>>>
Ln: 46 Col: 4
```