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.

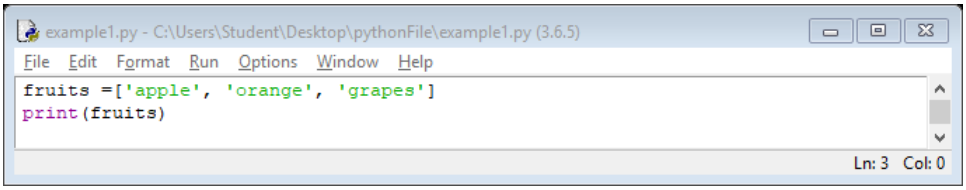
To create a list by using square brackets [ ]

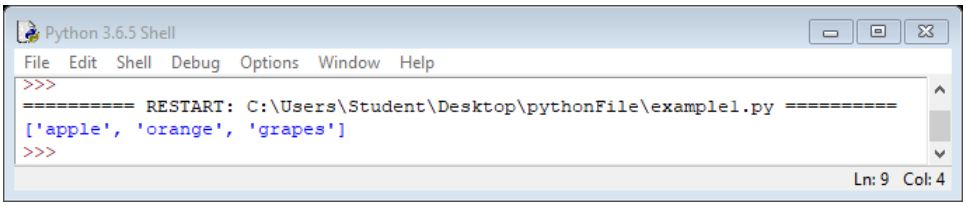
fruits =['apple', 'orange', 'grapes']

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

fruits =	apple	orange	grapes
Index location	0	1	2

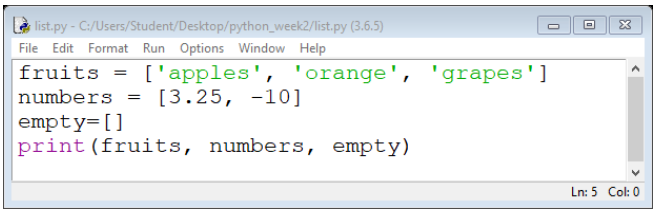
The Python code will look as the following:

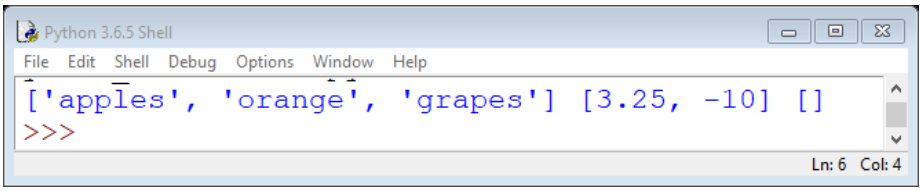




Empty list

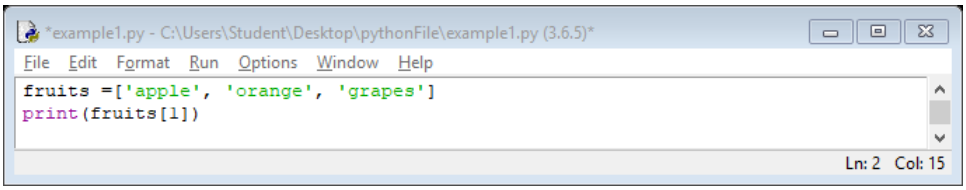
A list that contains no elements is called an empty list; you can create one with empty brackets, **[ ]**

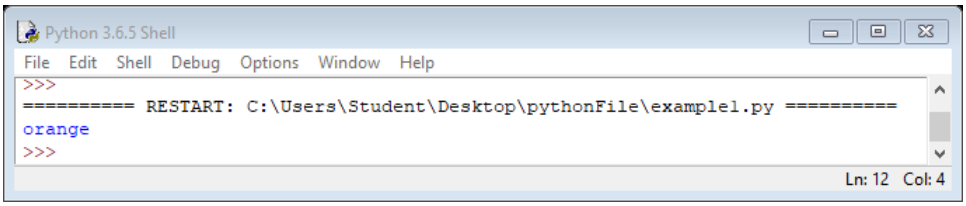




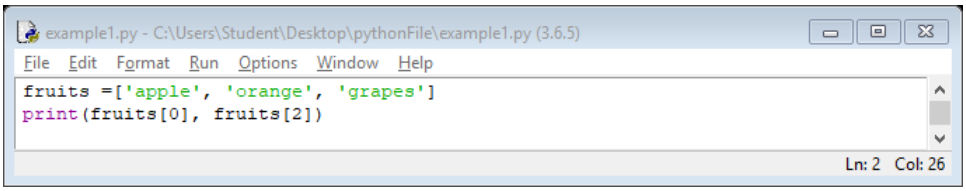
Accessing to elements in the list

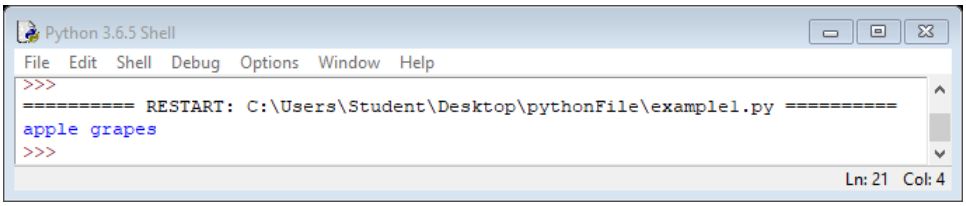
Creating a list takes a bit more typing than creating a string, but a list is more useful than a string because it can be manipulated. The syntax for accessing the elements of a list is the same as for accessing the characters of a string — the **bracket operator [ ]**. For example, we could print the 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]`**





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

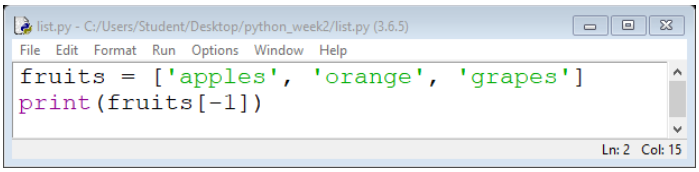


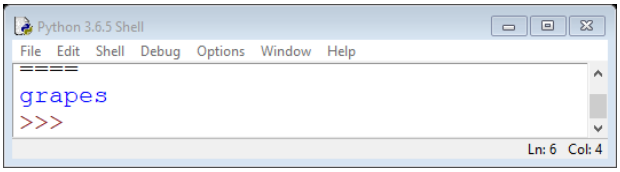


Negative index

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

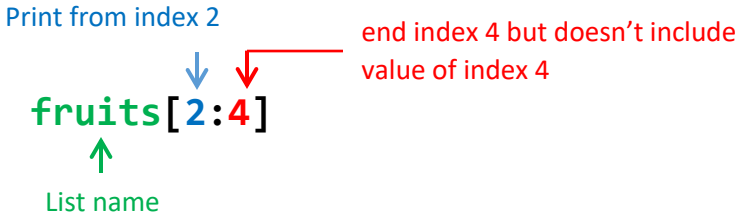
fruits =	apple	orange	grapes
Index location	-3	-2	-1



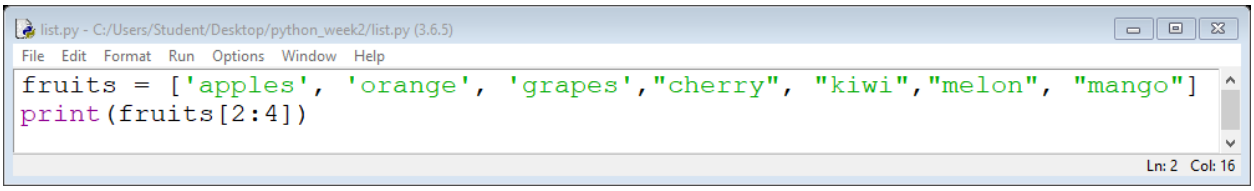


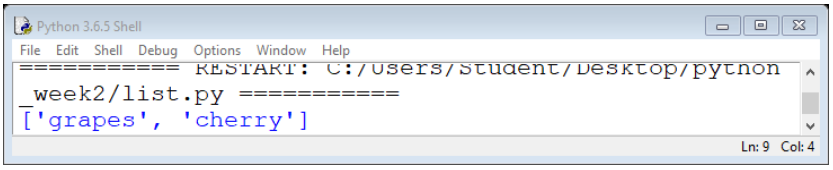
Range of indexes

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

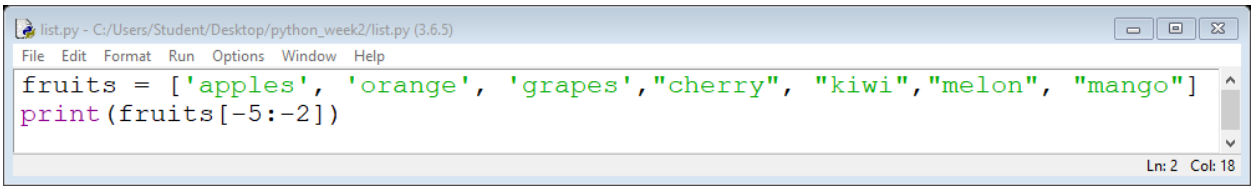


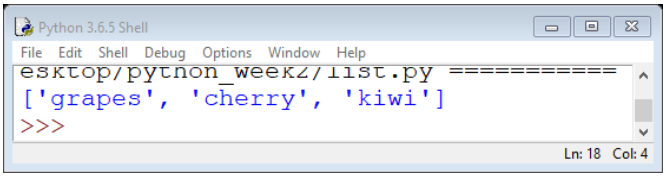
Example) Range of indexes: print list elements from index 2 up to index 4 (Exclusive)





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





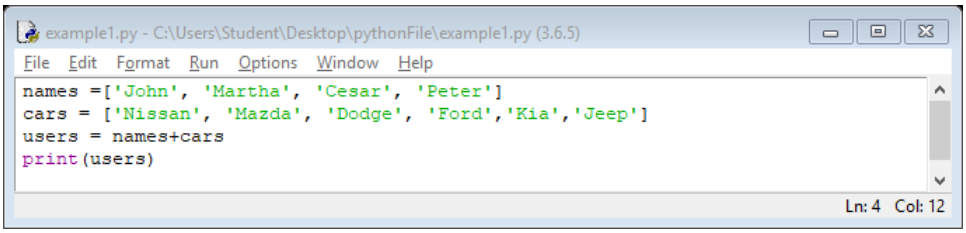
List operations

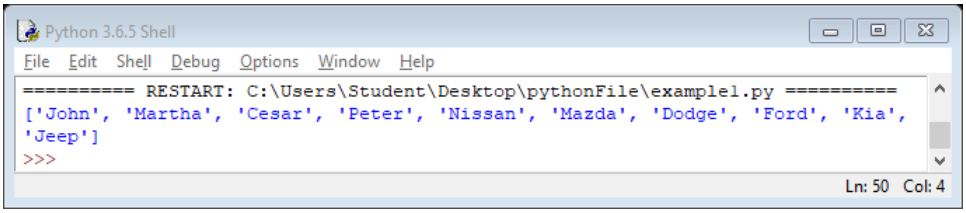
+ operator

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

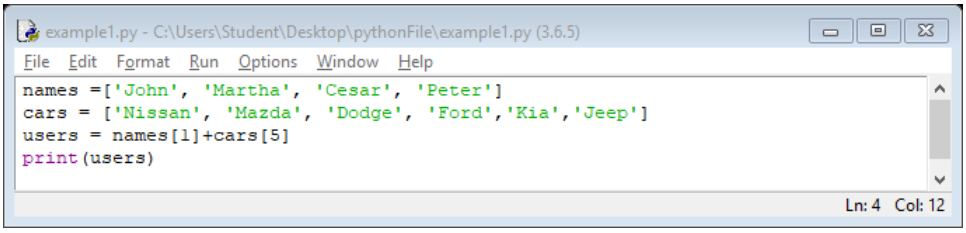
```
names =['John', 'Martha', 'Cesar', 'Peter']
cars = ['Nissan', 'Mazda', 'Dodge', 'Ford','Kia','Jeep']
```

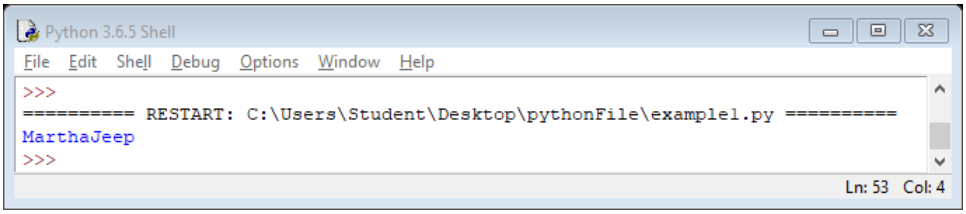
We can also add the two lists and set the result equal to another variable:





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





**** 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 *

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

```
========== 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*) follows by the list value. For example, to remove the second value from list *fruits*: `del fruits[1]`

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

```
>>>
========== 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 chuck 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')`

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

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

Extend item in a list

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

```
fruits = ['apples', 'orange', 'grapes']
numbers =[6.5, -10]
fruits.extend(numbers)
print(fruits)
```

```
['apples', 'orange', 'grapes', 6.5, -10]
>>>
```

Clear elements from a list

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

```
fruits = ['apples', 'orange', 'grapes']
fruits.clear()
print(fruits)
```

```
[]
>>>
```

Remove an element from the list with specific index

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

```
fruits = ['apples', 'orange', 'grapes']
fruits.pop(1)
print(fruits)
```

```
['apples', 'grapes']
>>>
```

Reverse a list

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

```
fruits = ['apples', 'orange', 'grapes']
fruits.reverse()
print(fruits)
```

```
['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

```
fruits = ['apples', 'orange', 'grapes']
fruits.sort()
print(fruits)
```

```
['apples', 'grapes', 'orange']
>>>
```

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

```
number=[2.8,2,-3,-5.8]
number.sort()
print(number)
```

```
[-5.8, -3, 2, 2.8]
>>>
```

copy a list

`copy()` returns a copy of the list

```
fruits = ['apples', 'orange', 'grapes']
copyFruits = fruits.copy()
print("fruits list: ", fruits)
print("copyFruits list: ", copyFruits)
```

```
fruits list:  ['apples', 'orange', 'grapes']
copyFruits list:  ['apples', 'orange', 'grapes']
>>>
```

Return number of a specific value

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

```
fruits = ['apples', 'orange', 'grapes','orange']
x = fruits.count('orange')
print("Number of 'orange' in the list: ", x)
```

```
Number of 'orange' in the list:  2
>>>
```

Index of a value in the list

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

```
fruits = ['apples', 'orange', 'grapes','orange']
y = fruits.index('orange')
print("Index of 'orange' in the list: ", y)
```

```
Index of 'orange' in the list:  1
>>>
```