



List, Tuples, Set, Dictionaries

By Aksadur Rahman

aksadur@yahoo.com



Agenda

Lists

List Methods

Range Function in a list

2D Lists/Matrix

Tuples

Unpacking/Comparing

Difference between List and Tuple

Set (Union/Intersection/Difference)

Difference between List and Set

Dictionaries

Lists

Lists are used to store multiple items in a single variable. Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage. Lists are created using square brackets:

```
thislist = ["apple", "banana", "cherry", "apple", "cherry"]  
print(thislist)
```

```
list1 = ["apple", "banana", "cherry"]  
list2 = [1, 5, 7, 9, 3]  
list3 = [True, False, False]  
list4 = ["abc", 34, True, 40, "male"]  
list5 = ["apple"]  
string1 = "apple"  
print(list1[0])  
print(string1[0])
```

```
list1 = ["Apple", "Banana", "Cherry", "Mango", "Guava"]  
  
print(list1[0:2])  
print(list1[2:])  
print(list1[-1])  
print(list1[-3:-1])  
print(list1[0:5])  
print(list1[0:5:2])  
print(list1[-1:-2])  
print(list1[-1:-2:-1])
```

Lists

```
list1 = ["apple", "banana", "cherry", "mango", "orange"]  
print(list1 + ["tomato", 50])  
print(list1 * 3)  
print(list1)
```

```
list1 = ["apple", "banana", "cherry", "mango", "orange"]  
list1[0] = "tomato"  
print(list1)
```

List Methods

Python has a set of built-in methods that you can use on lists/arrays

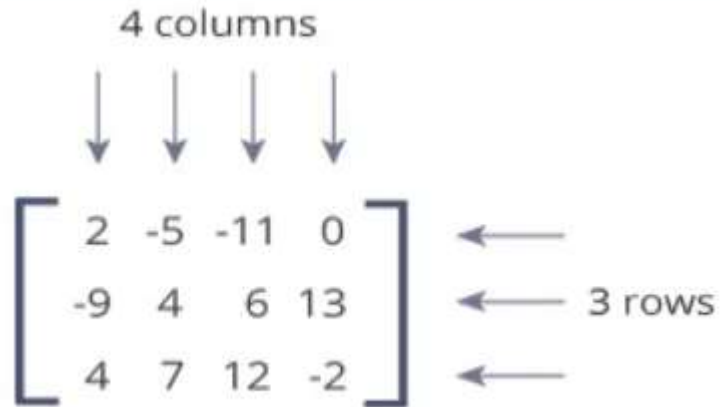
Method		Description
<u>append()</u>	<pre>fruits = ['apple', 'banana', 'cherry'] fruits.append("orange") print(fruits)</pre>	Adds an element at the end of the list
<u>clear()</u>	<pre>fruits = ['apple', 'banana', 'cherry', 'orange'] fruits.clear() print(fruits)</pre>	Removes all the elements from the list
<u>count()</u>	<pre>fruits = ["cherry", "apple", "banana", "cherry"] x = fruits.count("cherry") print(x)</pre>	Returns the number of elements with the specified value
<u>extend()</u>	<pre>fruits = ['apple', 'banana', 'cherry'] cars = ['Ford', 'BMW', 'Volvo'] fruits.extend(cars) print(fruits)</pre>	Add the elements of a list (or any iterable), to the end of the current list

List Methods

Method		Description
<u>index()</u>	<pre>fruits = ['apple', 'banana', 'cherry'] x = fruits.index("cherry") print(x)</pre>	Returns the index of the specified value
<u>insert()</u>	<pre>fruits = ['apple', 'banana', 'cherry'] fruits.insert(1, "orange") print(fruits)</pre>	Adds an element at the specified position
<u>pop()</u>	<pre>fruits = ['apple', 'banana', 'cherry'] fruits.pop(1) print(fruits)</pre>	Removes the element at the specified position
<u>remove()</u>	<pre>fruits = ['apple', 'banana', 'cherry'] fruits.remove("banana") print(fruits)</pre>	Removes the first item with the specified value
<u>reverse()</u>	<pre>fruits = ['apple', 'banana', 'cherry'] fruits.reverse() print(fruits)</pre>	Reverses the order of the list
<u>sort()</u>	<pre>cars = ['Ford', 'BMW', 'Volvo'] cars.sort() print(cars) cars.sort(reverse=False)</pre>	Sorts the list

2D Lists/Matrix

A Python matrix is a specialized two-dimensional rectangular array of data stored in rows and columns. The data in a matrix can be numbers, strings, expressions, symbols, etc.



A diagram illustrating a 2D list (matrix) with 3 rows and 4 columns. The matrix is enclosed in large square brackets. Above the matrix, four downward-pointing arrows are labeled "4 columns". To the right of the matrix, three leftward-pointing arrows are labeled "3 rows".

2	-5	-11	0
-9	4	6	13
4	7	12	-2

```
A = [[1, 4, 5, 12],  
      [-5, 8, 9, 0],  
      [-6, 7, 11, 19]]
```

```
print("A =", A)  
print("A[1] =", A[1])  
print("A[1][2] =", A[1][2])  
print("A[0][-1] =", A[0][-1])
```

```
column = []      # empty list  
for row in A:  
    column.append(row[2])  
  
print("3rd column =", column)
```

Tuples

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with **round brackets**.

```
thistuple = ("apple", "banana", "cherry")  
print(thistuple)
```


Difference between List and Tuple

SR.NO.	LIST	TUPLE
1	[] Square Braces is used	() First Braces is used
2	Lists are mutable	Tuples are immutable
3	Lists have several built-in methods	Tuple does not have many built-in methods.
4	Lists consume more memory	Tuple consume less memory as compared to the list

Unpacking/Comparing

We are allowed to extract the values back into variables of a tuples. This is called "unpacking":

```
fruits = ("apple", "banana", "cherry")
```

```
green, yellow, red = fruits
```

```
print(green)
```

```
print(yellow)
```

```
print(red)
```

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
```

```
green, yellow, *red = fruits
```

```
print(green)
```

```
print(yellow)
```

```
print(red)
```

```
a=(8,9)
```

```
b=(9)
```

```
if a>b:
```

```
    print("a is bigger")
```

```
else:
```

```
    print("b is bigger")
```

Set(Union/Intersection/Difference)

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.

A set is a collection which is unordered, unchangeable*, and unindexed.

* Note: Set items are unchangeable, but you can remove items and add new items.

```
# Different types of sets in Python
# set of integers
my_set = {1, 2, 3}
print(my_set)

# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

```
# set cannot have duplicates
my_set = {1, 2, 3, 4, 3, 2}
print(my_set)

# we can make set from a list
my_set = set([1, 2, 3, 2])
print(my_set)
```

```
num1 = {1, 2, 3, 4, 5}
num2 = {4, 5, 6, 7}

print(num1 | num2) #union
print(num1 & num2) #intersection
print(num1 - num2) #difference
```

```
# sorting
aa = {100, 200, 300, 400}
print(sorted(aa))
```

Set(Method)

https://www.w3schools.com/python/python_ref_set.asp

Difference between List and Set

S.No	List	Set
1.	The list implementation allows us to add the same or duplicate elements.	The set implementation doesn't allow us to add the same or duplicate elements.
2.	The insertion order is maintained by the List.	It doesn't maintain the insertion order of elements.
3.	We can get the element of a specified index	We cannot find the element from the Set based on the index
4.	It is used when we want to frequently access the elements by using the index.	It is used when we want to design a collection of distinct elements.

Dictionaries

Dictionaries are used to store data values in **key:value pairs**.

A dictionary is a collection which is ordered, changeable and do not allow duplicates.

Dictionaries are written with curly brackets, and have keys and values:

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
print(thisdict)
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
print(thisdict["brand"])
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
print(thisdict.get("Name", "Invalid Key"))
```

```
thisdict = {  
    "brand": "Ford",  
    "electric": False,  
    "year": 1964,  
    "colors": ["red", "white", "blue"]  
}  
print(thisdict.items())  
print(thisdict.keys())  
print(thisdict.values())
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964,  
    "year": 2020  
}  
print(thisdict)
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
  
for x, y in thisdict.items():  
    print(x, y)
```

```
text = "The banana is common fruit of Bangladesh. It is grown  
in all the districts and all the seasons. Munsigonj and Narsingdi  
are famous for banana . banana has several varieties such as  
Sagore, Shabri, Chapa, Agniswar etc. It is very nutritious and  
sweet.Papaw is a delicious fruit.It keeps the liver function  
active. The papaw is grown in all the districts and in all seasons.  
The coconut is a common fruit of Bangladesh. It grows in all  
seasons. Its water is a sweet drink and its kernel is a tasty food.  
The mango , the orange , the lichi , the black-berry , the jack  
fruits, the pineapple etc. grow in different seasons."  
fruits = ["banana", "Papaw", "coconut", "black-berry",  
"pineapple", "lichi", "orange", "mango"]
```

```
text = text.split(" ")  
fruitList = []  
for x in text:  
    if x in fruits:  
        fruitList.append(x)  
print(fruitList)
```

```
fruitList = set({})
for x in text:
    if x in fruits:
        fruitList.add(x)
print(fruitList)
```

```
fruitList = {}
for x in text:
    if x in fruits:
        fruitList[x] = 1
print(fruitList)
```

```
fruitList = {}
for x in text:
    if x in fruits:
        if x not in fruitList.keys():
            fruitList[x] = 1
        else:
            fruitList[x] += 1
print(fruitList)
```




Functions

By Aksadur Rahman

aksadur@yahoo.com



Functions

Functions

Parameters/Arguments

Keyword Parameters/Arguments

Default Parameter Value

Return Statement

Lambda Function

Map and Filter function

Zip Function

Recursion

Functions

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

Built in function

```
print("Hello World")
```

```
mylist = ["apple", "banana", "cherry"]  
x = len(mylist)
```

```
x = max(5, 10)
```

```
print('Enter your name:')  
x = input()  
print('Hello, ' + x)
```

User define function

```
def my_function():  
    print("Hello from a function")  
  
my_function()
```

Parameters/Arguments

```
def my_function(fname): #Parameters  
    print("Good morning!", fname)
```

```
my_function("Emil") #Arguments  
my_function("Tobias")  
my_function("Linus")
```

#This function expects 2 arguments, and gets 2 arguments:

```
def my_function(fname, lname):  
    print(fname, lname)
```

```
my_function("Emil", "Refsnes")
```

Keyword Parameters/Arguments

#You can also send arguments with the key = value syntax.
#This way the order of the arguments does not matter.

```
def my_function(child3, child2, child1):  
    print("The youngest child is", child3)
```

```
my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

Default Parameter Value

```
def my_function(country = "Norway"):  
    print("I am from", country)  
  
my_function("Sweden")  
my_function("India")  
my_function()  
my_function("Brazil")
```

Return Statement

```
def my_function(x):  
    return 5 * x
```

```
print(my_function(3))  
print(my_function(5))  
print(my_function(9))
```

```
def add_and_multiple(n1,n2):  
    sum = n1 + n2  
    mult = n1 * n2  
    return sum, mult
```

```
x, y = add_and_multiple(2,3)  
print(x, y)
```

Docstring

```
def Add(a, b):  
    """  
    This will take two parameter  
    Return value is integer  
    """  
    return a + b  
  
print(Add(10, 20))  
print(Add.__doc__)  
print(help(Add))
```


Lambda Function

Python Lambda Functions are anonymous function means that the function is without a name.

(lambda parameters: expression) (arguments)

```
print((lambda a : a + 10) (5))
```

```
x = lambda a : a + 10  
print(x(5))
```

```
x = lambda a, b, c : a + b + c  
print(x(5, 6, 2))
```

```
def cube(y):  
    return y * y * y
```

```
print(cube(5))
```

```
# using the lambda function  
lambda_cube = lambda y: y * y * y  
print(lambda_cube(5))
```

Map and Filter function

```
def myfunc(a):  
    return len(a)  
  
x = map(myfunc, ('apple', 'banana', 'cherry'))  
print(list(x))
```

```
def myFunc(x):  
    if x < 18:  
        return False  
    else:  
        return True  
  
ages = [5, 12, 17, 18, 24, 32]  
  
adults = list(filter(myFunc, ages))  
  
print(adults)
```

```
def square(x):  
    return x*x  
  
num = [1, 2, 3, 4, 5]  
result = list(map(square, num))  
  
print(result)
```

```
num = [5, 12, 17, 18, 25, 32]  
  
result = list(filter(lambda x: x%2==0, num))  
  
print(result)  
print(num)
```