

Python Lists

Chapter 8

Python for Everybody
www.py4e.com



Programming

Algorithms

- A set of rules or steps used to solve a problem

Data Structures

- A particular way of organizing data in a computer

<https://en.wikipedia.org/wiki/Algorithm>

https://en.wikipedia.org/wiki/Data_structure

What is Not A “Collection”?

Most of our **variables** have one value in them - when we put a new value in the **variable**, the old value is overwritten

```
$ python
>>> x = 2
>>> x = 4
>>> print(x)
4
```

A List Is a Kind of Collection



- A **collection** allows us to put many values in a single “**variable**”
- A **collection** is nice because we can carry **many values** around in one convenient package.

```
friends = [ 'Joseph', 'Glenn', 'Sally' ]
```

```
carryon = [ 'socks', 'shirt', 'perfume' ]
```

List Constants

- **List** constants are surrounded by square brackets and the elements in the list are separated by commas
- A **list** element can be any Python object - even **another list**
- A **list** can be empty

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['red', 'yellow',
'blue'])
['red', 'yellow', 'blue']
>>> print(['red', 24, 98.6])
['red', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

We Already Use Lists!

```
for i in [5, 4, 3, 2, 1] :  
    print(i)  
print('Blastoff!')
```

5

4

3


2

1

Blastoff!

Lists and Definite Loops – Best Pals

```
friends = ['Joseph', 'Glenn', 'Sally']  
for friend in friends :  
    print('Happy New Year:', friend)  
print('Done!')
```



Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!

```
z = ['Joseph', 'Glenn', 'Sally']  
for x in z:  
    print('Happy New Year:', x)  
print('Done!')
```



Looking Inside Lists

Just like strings, we can get at any single element in a list using an index specified in **square brackets**

Joseph	Glenn	Sally
0	1	2

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]  
>>> print(friends[1])  
Glenn  
>>>
```


Lists Are Mutable

- Strings are “immutable” - we cannot change the contents of a string - we must make a new string to make any change
- Lists are “mutable” - we can change an element of a list using the index operator

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not support item assignment
>>> x = fruit.lower()
>>> print(x)
banana
>>> lotto = [2, 14, 26, 41, 63]
>>> print(lotto)
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print(lotto)
[2, 14, 28, 41, 63]
```

How Long is a List?

- The `len()` function takes a `list` as a parameter and returns the number of `elements` in the `list`
- Actually `len()` tells us the number of elements of any set or sequence (such as a string...)

```
>>> greet = 'Hello Bob'
>>> print(len(greet))
9
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
>>>
```

Using the Range Function

- The `range` function returns a list of numbers that range from zero to one less than the `parameter`
- We can construct an index loop using `for` and an integer `iterator`

```
>>> print(range(4))
[0, 1, 2, 3]
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(len(friends))
3
>>> print(list(range(len(friends))))
[0, 1, 2]
>>>
```

A Tale of Two Loops...

```
friends = ['Joseph', 'Glenn', 'Sally']
```

```
for friend in friends :  
    print('Happy New Year:', friend)
```

```
for i in range(len(friends)) :  
    friend = friends[i]  
    print('Happy New Year:', friend)
```

```
>>> friends = ['Joseph', 'Glenn', 'Sally']  
>>> print(len(friends))  
3  
>>> print(list(range(len(friends))))  
[0, 1, 2]  
>>>
```

```
Happy New Year: Joseph  
Happy New Year: Glenn  
Happy New Year: Sally
```