

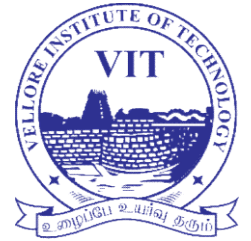


ITA6017

Python Programming

Dr. Arun Pandian J

Module 3: Python's List, Tuples, Dictionaries & Sets



Lists and its operations, Ranges: Iterators and its purpose, Tuples: Operation and usage, Python Dictionaries, examples on Dictionaries, Sets and its operations

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
```



List

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 all 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(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(range(len(friends)))
[0, 1, 2]
>>>
```

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

Concatenating Lists Using +



We can create a new list by adding two existing lists together

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Lists Can Be Sliced Using :



```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41, 12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

Remember: Just like in strings, the second number is “up to but not including”



List Methods

```
>>> x = list()
>>> type(x)
<type 'list'>
>>> dir(x)
['append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

<http://docs.python.org/tutorial/datastructures.html>



Building a List from Scratch

- We can create an empty list and then add elements using the append method
- The list stays in order and new elements are added at the end of the list

```
>>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print(stuff)
['book', 99]
>>> stuff.append('cookie')
>>> print(stuff)
['book', 99, 'cookie']
```




Is Something in a List?

- Python provides two operators that let you check if an item is in a list
- These are logical operators that return True or False
- They do not modify the list

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
>>>
```



Lists are in Order

A list can hold many items and keeps those items in the order until we do something to change the order

A list can be sorted (i.e., change its order)

The sort method (unlike in strings) means “sort yourself”

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



Built-in Functions and Lists

- There are a number of functions built into Python that take lists as parameters
- Remember the loops we built? These are much simpler.

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print(len(nums))
6
>>> print(max(nums))
74
>>> print(min(nums))
3
>>> print(sum(nums))
154
>>> print(sum(nums)/len(nums))
25.6
```



```
total = 0
count = 0
while True :
    inp = input('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    total = total + value
    count = count + 1
```

```
average = total / count
print('Average:', average)
```

```
Enter a number: 3
Enter a number: 9
Enter a number: 5
Enter a number: done
Average: 5.66666666667
```

```
numlist = list()
while True :
    inp = input('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    numlist.append(value)

average = sum(numlist) / len(numlist)
print('Average:', average)
```



Task:

Consider a list (`list = []`). You can perform the following commands:

- Insert `e` at position `i`.
- Print the list.
- Delete the first occurrence of integer `e`.
- Insert integer `e` at the end of the list.
- Sort the list.
- Pop the last element from the list.
- Reverse the list.
- Find occurrence of integer `e` in the list.



Task:

- Given a list of ten numbers, the task is to write a Python program to find the second largest and second smallest number in the given list.



Task:

write a Python program to split the prime and composite number as separate list and order it.

Example:

Input:

[4, 2, 12, 3, 16, 5, 7, 64]

Output:

[2, 3, 5, 7]

[4, 12, 16, 64]