Python has four basic inbuilt data structures namely Lists, Dictionary, Tuple and Set. These almost cover 80% of the our real world data structures. This article will cover the above mentioned topics.

Above mentioned topics are divided into four sections below.

1. **Lists :**Lists in Python are one of the most versatile collection object types available. The other two types are dictionaries and tuples, but they are really more like variations of lists.
   * Python lists do the work of most of the collection data structures found in other languages and since they are built-in, you don’t have to worry about manually creating them.
   * Lists can be used for any type of object, from numbers and strings to more lists.
   * They are accessed just like strings (e.g. slicing and concatenation) so they are simple to use and they’re variable length, i.e. they grow and shrink automatically as they’re used.
   * In reality, Python lists are C arrays inside the Python interpreter and act just like an array of pointers.

# Python program to illustrate a simple **list**

**# Declaring a list**

L = [1, "a" , "string" , 1+2]

print L

**# add 6 to the above list**

L.append(6)

print L

**# pop deletes the last element of the list**

L.pop()

**print L**

print L[1]

Output:

[1, 'a', 'string', 3]

[1, 'a', 'string', 3, 6]

[1, 'a', 'string', 3]

a

There are various Functions that can be carried out on lists like append(), extend(), reverse(), pop() etc.

1. **Dictionary:** In python, dictionary is similar to hash or maps in other languages. It consists of key value pairs. The value can be accessed by unique key in the dictionary.

* Keys are unique & immutable objects.
* Syntax: dictionary = {"key name": value}

# Python program to illustrate **dictionary**

**# Create a new dictionary**

d = dict() # or d = {}

**# Add a key - value pairs to dictionary**

d['xyz'] = 123

d['abc'] = 345

**# print the whole dictionary**

print d

**# print only the keys**

print d.keys()

**# print only values**

print d.values()

**# iterate over dictionary**

for i in d :

    print "%s %d" %(i, d[i])

**# another method of iteration**

for index, value in enumerate(d):

    print index, value , d[value]

**# check if key exist**

print 'xyz' in d

**# delete the key-value pair**

del d['xyz']

**# check again**

print "xyz" in d

Output:

{'xyz': 123, 'abc': 345}

['xyz', 'abc']

[123, 345]

xyz 123

abc 345

0 xyz 123

1 abc 345

True

False

1. **Sets:**Unordered collection of unique objects.

* Set operations such as union (|) , intersection(&), difference(-) can be applied on a set.
* Sets are immutable i.e once created further data can’t be added to them
* () are used to represent a set.Objects placed inside these brackets would be treated as a set.

# Python program to demonstrate working of **Sets** in Python

**# Creating two sets**

set1 = set()

set2 = set()

**# Adding elements to set1**

for i in range(1, 6):

    set1.add(i)

**# Adding elements to set2**

for i in range(3, 8):

    set2.add(i)

print("Set1 = ", set1)

print("Set2 = ", set2)

print("\n")

Output:

('Set1 = ', set([1, 2, 3, 4, 5]))

('Set2 = ', set([3, 4, 5, 6, 7]))

1. **Tuple:** Python tuples work exactly like Python lists except they are immutable, i.e. they can’t be changed in place. They are normally written inside parentheses to distinguish them from lists (which use square brackets), but as you’ll see, parentheses aren’t always necessary. Since tuples are immutable, their length is fixed. To grow or shrink a tuple, a new tuple must be created.

# Python program to illustrate **tuple**

tup = (1, "a", "string", 1+2)

print tup

print tup[1]

Output:

(1, 'a', 'string', 3)

a