DATA STRUCTURES

Data structures are basically just that - they are **structures** which can hold some **data** together. In other words, they are used to store a collection of related data.

There are four built-in data structures in Python - *list, tuple, dictionary and set.* We will see how to use each of them and how they make life easier for us.

List

A list is a data structure that holds an ordered collection of items i.e. you can store a **sequence** of items in a list. This is easy to imagine if you can think of a shopping list where you have a list of items to buy, except that you probably have each item on a separate line in your shopping list whereas in Python you put commas in between them.

The list of items should be enclosed in square brackets so that Python understands that you are specifying a list. Once you have created a list, you can add, remove or search for items in the list. Since we can add and remove items, we say that a list is a **mutable** data type i.e. this type can be altered.

Quick Introduction To Objects And Classes

Although I've been generally delaying the discussion of objects and classes till now, a little explanation is needed right now so that you can understand lists better. We will explore this topic in detail in a later chapter.

A list is an example of usage of objects and classes. When we use a variable `i` and assign a value to it, say integer `5` to it, you can think of it as creating an **object** (i.e. instance) `i` of **class** (i.e. type) `int`. In fact, you can read `help(int)` to understand this better.

A class can also have **methods** i.e. functions defined for use with respect to that class only. You can use these pieces of functionality only when you have an object of that class. For example, Python provides an `append` method for the `list`class which allows you to add an item to the end of the list. For example, `mylist.append('an item')` will add that string to the list `mylist`. Note the use of dotted notation for accessing methods of the objects.

A class can also have **fields** which are nothing but variables defined for use with respect to that class only. You can use these variables/names only when you have an object of that class. Fields are also accessed by the dotted notation, for example, `mylist.field`.

Example (save as ds_using_list.py):

```
# This is my shopping list
shoplist = ['apple', 'mango', 'carrot', 'banana']

print 'I have', len(shoplist), 'items to purchase.'

print 'These items are:',
for item in shoplist:
    print item,

print '\nI also have to buy rice.'
shoplist.append('rice')
print 'My shopping list is now', shoplist

print 'I will sort my list now'
shoplist.sort()
print 'Sorted shopping list is', shoplist

print 'The first item I will buy is', shoplist[0]
olditem = shoplist[0]
```

```
del shoplist[0]

print 'I bought the', olditem

print 'My shopping list is now', shoplist
```

Output:

```
$ python ds_using_list.py

I have 4 items to purchase.

These items are: apple mango carrot banana

I also have to buy rice.

My shopping list is now ['apple', 'mango', 'carrot', 'banana', 'rice']

I will sort my list now

Sorted shopping list is ['apple', 'banana', 'carrot', 'mango', 'rice']

The first item I will buy is apple

I bought the apple

My shopping list is now ['banana', 'carrot', 'mango', 'rice']
```

How It Works

The variable `shoplist` is a shopping list for someone who is going to the market.

In `shoplist`, we only store strings of the names of the items to buy but you can add *any kind of object* to a list including numbers and even other lists.

We have also used the `for..in` loop to iterate through the items of the list. By now, you must have realized that a list is also a sequence. The specialty of sequences will be discussed in a later section.

Notice the use of the trailing comma in the `print` statement to indicate that we want to end the output with a space instead of the usual line break. Think of the comma as telling Python that we have more items to print on the same line.

Next, we add an item to the list using the `append` method of the list object, as already discussed before. Then, we check that the item has been indeed added to the list by printing the contents of the list by simply passing the list to the `print` statement which prints it neatly.

Then, we sort the list by using the `sort` method of the list. It is important to understand that this method affects the list itself and does not return a modified list - this is different from the way strings work. This is what we mean by saying that lists are *mutable* and that strings are *immutable*.

Next, when we finish buying an item in the market, we want to remove it from the list. We achieve this by using the `del` statement. Here, we mention which item of the list we want to remove and the `del` statement removes it from the list for us.

We specify that we want to remove the first item from the list and hence we use `del shoplist[0]` (remember that Python starts counting from 0).

If you want to know all the methods defined by the list object, see `help(list)` for details.

Source: http://www.swaroopch.com/notes/python/