SEQUENCE

Lists, tuples and strings are examples of sequences, but what are sequences and what is so special about them?

The major features are **membership tests**, (i.e. the `in` and `not in` expressions) and **indexing operations**, which allow us to fetch a particular item in the sequence directly.

The three types of sequences mentioned above - lists, tuples and strings, also have a **slicing** operation which allows us to retrieve a slice of the sequence i.e. a part of the sequence.

Example (save as `ds_seq.py`):

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
name = 'swaroop'


# Indexing or 'Subscription' operation #
print 'Item 0 is', shoplist[0]
print 'Item 1 is', shoplist[1]
```

```
print 'Item 2 is', shoplist[2]

print 'Item 3 is', shoplist[3]

print 'Item -1 is', shoplist[-1]

print 'Item -2 is', shoplist[-2]

print 'Character 0 is', name[0]


# Slicing on a list #

print 'Item 1 to 3 is', shoplist[1:3]

print 'Item 2 to end is', shoplist[2:]

print 'Item 1 to -1 is', shoplist[1:-1]

print 'Item start to end is', shoplist[:]


# Slicing on a string #

print 'characters 1 to 3 is', name[1:3]

print 'characters 2 to end is', name[2:]

print 'characters 1 to -1 is', name[1:-1]

print 'characters start to end is', name[:]
```

Output:

```
$ python ds_seq.py

Item 0 is apple

Item 1 is mango

Item 2 is carrot

Item 3 is banana

Item -1 is banana

Item -2 is carrot

Character 0 is s

Item 1 to 3 is ['mango', 'carrot']

Item 2 to end is ['carrot', 'banana']

Item 1 to -1 is ['mango', 'carrot']

Item start to end is ['apple', 'mango', 'carrot', 'banana']

characters 1 to 3 is wa

characters 2 to end is aroop

characters 1 to -1 is waroo

characters start to end is swaroop
```

How It Works

First, we see how to use indexes to get individual items of a sequence. This is also referred to as the *subscription operation*. Whenever you specify a number to a sequence within square brackets as shown above, Python will fetch you the item corresponding to that position in the sequence. Remember that Python starts counting numbers from 0. Hence, `shoplist[0]` fetches the first item and `shoplist[3]` fetches the fourth item in the `shoplist`sequence.

The index can also be a negative number, in which case, the position is calculated from the end of the sequence. Therefore, `shoplist[-1]` refers to the last item in the sequence and `shoplist[-2]` fetches the second last item in the sequence.

The slicing operation is used by specifying the name of the sequence followed by an optional pair of numbers separated by a colon within square brackets. Note that this is very similar to the indexing operation you have been using till now. Remember the numbers are optional but the colon isn't.

The first number (before the colon) in the slicing operation refers to the position from where the slice starts and the second number (after the colon) indicates where the slice will stop at. If the first number is not specified, Python will start at the beginning of the sequence.

If the second number is left out, Python will stop at the end of the sequence. Note that the slice returned *starts* at the start position and will end just before the *end* position i.e. the start position is included but the end position is excluded from the sequence slice.

Thus, `shoplist[1:3]` returns a slice of the sequence starting at position 1, includes position 2 but stops at position 3 and therefore a **slice** of two items is returned. Similarly, `shoplist[:]` returns a copy of the whole sequence.

You can also do slicing with negative positions. Negative numbers are used for positions from the end of the sequence. For example, `shoplist[:-1]` will return a slice of the sequence which excludes the last item of the sequence but contains everything else.

You can also provide a third argument for the slice, which is the *step* for the slicing (by default, the step size is 1):

```
>>> shoplist = ['apple', 'mango', 'carrot', 'banana']
>>> shoplist[::1]
['apple', 'mango', 'carrot', 'banana']
>>> shoplist[::2]
['apple', 'carrot']
```

```
>>> shoplist[::3]
['apple', 'banana']
>>> shoplist[::-1]
['banana', 'carrot', 'mango', 'apple']
```

Notice that when the step is 2, we get the items with position 0, 2,… When the step size is 3, we get the items with position 0, 3, etc.

Try various combinations of such slice specifications using the Python interpreter interactively i.e. the prompt so that you can see the results immediately. The great thing about sequences is that you can access tuples, lists and strings all in the same way!

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