list comprehension

In set theory, we learn a couple of ways of creating a set.

a) enumeration : All the elements are listed.

Example: s = { 1, 3, 5, 7, 9}

b) rule based or builder method:

s = { x | x is odd and 1 <= x <= 10 }

So far we have learnt how to create lists by enumeration. In this section, we shall learn how to create lists using rules. This method of list creation is called list comprehension.

We shall see examples from the file list_comp.py.

```
l1 = [ 'hello' for x in range(5)]
```

This statement would create a list containing hello five times.

The above statement is equivalent to the following piece of code.

```
l1 = []
for x in range(5):
    l1.append(x)
```

The general form of list comprehension is

[<expr> for <variable> in <iterable>]

Semantically, this is equivalent the following.

Create an empty list. Execute the for part of the list comprehension. Evaluate the expr each time. Append that to the list. The result is the list so created.

Observe the similarity with map.

Let us look at some more examples.

```
# squares of numbers from 1 to 5
l2 = [ x * x for x in range(5)]
print(l2)
```

Is the above code self explanatory?

```
# list of tuples having a number and its square
l3 = [ (x, x * x) for x in range(5)]
print(l3)
```

```
# list of strings and its length
l4 = [ (x, len(x)) for x in ['bangalore', 'mysore', 'hubballi', 'shivamogga']]
print(l4)
```

The one below is an example of nested loops in list comprehension.

```
# cartesian product
l5 = [ (x, y) for x in range(4) for y in range(4)]
print(l5)
```

The one below is an example of selection amongst the many produced by the loops. Observe the similarity with flter.

In fact it is a combination of map and flter.

```
# relation: partial order
l6 = [ (x, y) for x in range(4) for y in range(4) if x < y]
print(l6)
# convert all words to uppercase
# map
a = ['bangalore', 'mysore', 'hubballi', 'shivamogga' ]
b = [ x.upper() for x in a ]
print(b)

# flter
# fnd all words whose len exceeds 7
b = [ x for x in a if len(x) > 7]
print(b)
# convert all words to uppercase if len exceeds 7
# combine
b = [ x.upper() for x in a if len(x) > 7]
print(b)
```

list comprehension provides an alternate mechanism to functional programming constructs map and flter.

We also have set, dict comprehension. These are not part of the course – hence not discussed.