LISTS []

Lists are mutable. Lists are like a sequence. It can hold items of various data types.

CREATING A LIST

```
In: myl = ['string', 23, 100.2, 'o']
```

LEN FUNCTION

```
In: len(myl)
o/p: 4
```

INDEXING & SLICING

```
In: myl = ['one', 'two', 'three', 4, 5]
In: myl[0]        # Indexing
o/p: 'one'
In: myl[1:3]      # Slicing
o/p: ['two', 'three']
```

CONCATENATION

qn: myl + ['new']

o/p: ['one', 'two', 'three', 4, 5, 'new']

Note: This doesn't actually change orginal list

REASSIGNMENT

qn: myl = myl + ['new']

qn: myl

o/p: ['one', 'two', 'three', 4, 5, 'new']

DUPLICATION

qn: myl * 2

o/p: ['one',
'two',
'three',
4,
5,
'new'

next Line

'one',
'two',
'three',
4,
5,
'new']

LIST METHODS

qn: l1 = [1, 2, 3]

APPEND → To permanently add an element to end

qn: l1.append('append')

qn: l1

o/p: [1, 2, 3, 'append']

POP - By default takes off the last index, but we can also specify index to pop

qn: l1.pop(0)

o/p: 1

qn: l1 # Show

o/p: [2, 3, 'append']

ASSIGN POPPED ELEMENT

qn: pi = l1.pop()

qn: pi

o/p: 'append'

qn: nl=['a','e','x','b','c']

qn: ~~nl.reverse()~~

qn: ~~n~~

REVERSE

qn: nl.reverse()

qn: nl

o/p: ['c','b','x','e','a']

SORT

qn: nl.sort()

qn: nl

o/p: ['a','b','c','e','x']

NESTING LISTS

This means we can have data structures within data structures.

qn: # Let's make three lists
l1=[1,2,3]
l2=[4,5,6]
l3=[7,8,9]

Make a Lists

In: matrix = [l1, l2, l3]

In: matrix

o/p: [[1,2,3], [4,5,6], [7,8,9]]

Grab first item in matrix object

In: matrix[0] # Grab 1st item in
o/p: [1,2,3] # matrix object

Grab 1st item of 1st item in the matrix object

In: matrix[0][0]
o/p: 1

LIST COMPREHENSIONS

They allow for quick constructions of lists.

In: first_col = [row[0] for row in matrix]

In: first_col

o/p: [1,4,7]

TUPLES ()

Tuples are very similar to lists, however unlike lists they are immutable.

Create a tuple

```
In: t = (1, 2, 3)
    len(t)
```

o/p: 3

Can also mix object types

```
In: t = ('one', 2)
In: t
```

o/p: ('one', 2)

Indexing

```
In: t[0]
```

o/p: 1

Slicing

```
In: t[-1]
```

o/p: 2

Methods (Object.method())

- INDEX → Enter a value & return the index

qn: t.index('one')

o/p: 0

- COUNT → count no. of times a value appears

qn: t.count('one')
o/p: 1

IMMUTABILITY

qn: t[0]='change'
o/p: TypeError: 'tuple' object doesn't allow item assignment

When to use tuples?

Tuples are not used as often as lists in programming, but used when immutability is necessary.