1 Administrative Topics

- Sign up for csstu
- Bruce will be teaching lab today, but I will be teaching lab tomorrow.

2 Lists

A list is a sequence of values of any type (including lists!).

And lists can have items added, replaced, or deleted. That means they can change - the Python term for that is mutable. So that means: A list is a mutable sequence of values of any type.

2.1 Creating Lists

We can create a list explicitly, like this:

```python
myList = [0,1,2,3]
```

The square brackets indicate the beginning and ending. The commas separate the different items. In this case the items are all ints, but that does not have to be the case.

```python
myOtherList = [‘CS’,1,’Cool!’,3]
```
2.2 List Operations

Just like there are operators that work on numbers, there are operators that work on lists:

There are several operators that shown in Table 1.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x in s</code></td>
<td>True if an item of <code>s</code> is equal to <code>x</code>, else False</td>
</tr>
<tr>
<td><code>x not in s</code></td>
<td>False if an item of <code>s</code> is equal to <code>x</code>, else True</td>
</tr>
<tr>
<td><code>s + t</code></td>
<td>the concatenation of <code>s</code> and <code>t</code></td>
</tr>
<tr>
<td><code>s * n, n * s</code></td>
<td><code>n</code> shallow copies of <code>s</code> concatenated</td>
</tr>
<tr>
<td><code>s[i]</code></td>
<td><code>i</code>th item of <code>s</code>, origin 0</td>
</tr>
<tr>
<td><code>s[i:j]</code></td>
<td>slice of <code>s</code> from <code>i</code> to <code>j</code> (but not including the <code>j</code>th item)</td>
</tr>
<tr>
<td><code>s[i:j:k]</code></td>
<td>slice of <code>s</code> from <code>i</code> to <code>j</code> with step <code>k</code> (but not including the <code>j</code>th item)</td>
</tr>
<tr>
<td><code>len(s)</code></td>
<td>length of <code>s</code></td>
</tr>
<tr>
<td><code>s[i] = x</code></td>
<td>item <code>i</code> of <code>s</code> is replaced by <code>x</code></td>
</tr>
<tr>
<td><code>s[i:j] = t</code></td>
<td>slice of <code>s</code> from <code>i</code> to <code>j</code> is replaced by the contents of the iterable <code>t</code></td>
</tr>
<tr>
<td><code>s[i:j:k] = t</code></td>
<td>the elements of <code>s[i:j:k]</code> are replaced by those of <code>t</code></td>
</tr>
</tbody>
</table>

When indexing into a list, it is acceptable to use negative integers. -1 indicates the last item in the last. -2 indicates the penultimate item in the list, etc.

When using non-negative integers to index, remember that the first item is at index 0, the second item is at index 1, etc.
Here are examples with print statements (and their outputs) using Python in interactive mode:

```python
>>> temperatures = [63, 65, 68, 70, 74, 75]
>>> print(temperatures[0])
63
>>> print(temperatures[-1])
75
>>> print(temperatures[-2])
74
>>> print(temperatures[1])
65
>>> print(temperatures[1:])
[65, 68, 70, 74, 75]
>>> print(temperatures[1:2])
[65]
>>> print(temperatures[:2])
[63, 65]
>>> print(2*temperatures)
[63, 65, 68, 70, 74, 75, 63, 65, 68, 70, 74, 75]
>>> print(75 in temperatures)
True
>>> print(30 in temperatures)
False
>>> print(30 not in temperatures)
True
>>> print(temperatures[0:2:5])
[63]
>>> print(temperatures[0:2:])
[63, 65]
>>> len(temperatures)
6
>>> print(temperatures[1:3] + temperatures[3:5])
[65, 68, 70, 74]
>>> print(temperatures[1:2] + temperatures[3:4])
[65, 70]
>>> print(temperatures[1] + temperatures[3])
135
>>> print(temperatures)
[63, 65, 68, 70, 74, 75]
>>> temperatures[1] = 0
>>> print(temperatures)
[63, 0, 68, 70, 74, 75]
>>> temperatures[1:3] = [1, 2]
>>> print(temperatures)
[63, 1, 2, 70, 74, 75]
```
2.3 Lists Methods

One of the most useful list methods is `append`, which appends an item to the end of a list. Its syntax, along with that of several other list methods is in Table 2.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s.append(x)</code></td>
<td>append the item <code>x</code> to the end of list <code>s</code></td>
</tr>
<tr>
<td><code>s.extend(t)</code></td>
<td>append each item of list <code>t</code> to the end of list <code>s</code></td>
</tr>
</tbody>
</table>

Here are a couple of examples. We will return to these functions when we talk about lists of lists.

```python
>>> temperatures = [63,65,68,70,74,75]
>>> temperatures.append(75)
>>> print temperatures
[63, 65, 68, 70, 74, 75, 75]
>>> temperatures.extend([77, 78])
>>> print temperatures
[63, 65, 68, 70, 74, 75, 75, 77, 78]
```

For a complete list of list methods, see
http://docs.python.org/library/stdtypes.html#mutable-sequence-types

3 For loops

We can write code to “loop over” a list - that means we want the loop body to execute once for each element in the list.

To compute print the Celsius to Fahrenheit mapping for a specific set of temperatures we could make a list of temperatures and then loop over the list.

We will do it three ways. First, we loop through the temperature list directly:

ctemps = [10, 15, 20, 25, 30]
for ctemp in ctemps:
    print ctemp, "deg C is", convertCelsius2Fahrenheit(ctemp), "deg F"

The “loop control variable” temp has the value 10 the first time through the
loop, 15 the second time through, 20 the third time through, etc. Notice that
we give it a name that reflects its meaning. We have a list of temperatures.
So each item of that list is a temperature. We are assuming units of Celsius,
so we call it ctemp.

Now we loop through the indexes of the list. This list has 5 elements, so its
indices are 0, 1, 2, 3, and 4.

ctemps = [10, 15, 20, 25, 30]
for i in [0, 1, 2, 3, 4]:
    print ctemps[i], "deg C is", convertCelsius2Fahrenheit(ctemps[i]), "deg F"

But that involves writing out a lot of numbers. We can do this more easily,
if we use a handy function called range.

- range(x) - where x is an int return a list 0, 1, 2, ... x-1 (i.e. the list of
  ints beginning at 0, stopping before x)
  >>> range(3)
  [0, 1, 2]

- range(b,e) - where b and e are ints, returns a list b, b+1, ... e-2, e-1.
  (i.e. the list of ints beginning at b, stopping before e)
  >>> range(1,5)
  [1, 2, 3, 4]

- range(b,e,s) - where b, e, and s are ints, returns a list of ints beginning
  at b, stopping before e, using step s)
  >>> range(1,6,2)
  [1, 3, 5]

If we use the range function, passing in the length of the list, we can easily
loop through the indices:

ctemps = [10, 15, 20, 25, 30]
for i in range(len(ctemps)):
    print ctemps[i], "deg C is", convertCelsius2Fahrenheit(ctemps[i]), "deg F"
The “loop control variable” \( i \) has the value 0 the first time through the loop, 1 the second time through, 2 the third time through, and 3 the fourth time through. Notice that we name it \( i \) because \( i \) is a commonly used symbol for an index. In math, we often use the subscript \( i \).

There is a way to access both the index and the value. We don’t need to for this example, but I will show it to you now:

```python
ctemps = [10, 15, 20, 25, 30]
for i, ctemp in enumerate(ctemps):
    print ctemp, "deg C is", convertCelsius2Fahrenheit(ctemp), "deg F"
```