For this module we will be covering Lists. We will look at how to create a list, finding items in a list to process, examine the various list methods and built-in functions, and how to process lists.

Sequence

A sequence is an object that holds multiple items of data, stored one after the other. You can perform operations on a sequence to examine and manipulate the items stored in it.

Lists

A list is an object that contains multiple data items. They are "mutable" - which means their contents can be changed during a program's execution. These are dynamic data structures. Each item in the list is called an "element". All of the elements in the list can have different data types. We reference things in the list by their position in the list.

```
even_numbers = [2, 4, 6, 8]

info = ['Alicia', 27, 17.25]

names = ['Mary', 'John', 'Phil', 'Joe']
```

There is a built in 'list' function as well:

```
numbers = list(range(5))   calls the range function with '5' as the argument and returns s list [0, 1, 2, 3, 4]
```

Repetition Operator

```
list * n        list is your list and n is the number of copies to make

numbers = [0] * 5

print(numbers)

[0, 0, 0, 0, 0]
```

Iterating over a List with for Loop

Iterating (repeating a number of times) can be done with a list also using the for loop:

```
numbers = [99, 100, 101, 102]

for n in numbers:

    print (n)
```

This will produce:

99

100

101

102

Indexing a List

We can also use an index to access individual elements of the list. The index is the specific position in the list. The index MUST start with 0 as it's position

```
my_list = [10, 20, 30, 40]

print(my_list[0], my_list[1], my_list[2], my_list[3]
```

or

```
index = 0

while index < 4

    print(my_list[index])

    index += 1
```

You can also use negative indexes - BUT remember index [-1] is the LAST element of the list.

The len Function

This built in function returns the length of a sequence (how many elements in the list).

Remember - we can change values in the list that has already been created with data - Mutable!

Concatenating Lists

To concatenate means to join two things together. We use the + operator to concatenate two lists.

List Slicing

Slicing expression selects a range of elements from a sequence in the list.

list_name[start : end] *****important - the "end" value is NOT included in the*

The in Operator

This allows you to search for an item in the list.

item in list

Methods and built-in Functions

append(item)
Method is commonly used to add items to a list.

index(item)

Returns the index of the first element whose value is equal to item.

insert (index, item)

Inserts item into the list at the specified index

sort()

Sorts the items in he list so they appear in ascending order (lowest to highest)

remove (item)

removes the first occurrence of the item from the list (the item value must exist in the list to remove)

reverse()

reverses the order of the items in the list

del my_list[2]

deletes the element at the index location

min and max functions

min will accept a list as an argument and returns the item that has the lowest value.

max will accept a list as an argument and returns the items that has the highest value.

Copying a List

To make a copy o f a list, you must copy the list's elements. Depending on your code - you might have two lists with different names - but actually point to the same memory location. You might also have two lists with different names that point to separate memory locations, but have the exact same data.

Processing Lists

Some common ways to process lists include: 1) total the values in a list; average the values in a list

Passing a List as an argument to a Function

There are built in functions that accept a complete list as an argument into the function. You can also return a list from a function.

Lists and Files

You can use a list to store the data that will be written out to a file.

2-Dimensional Lists

This type of list has other lists as its elements (rows and columns). When we have 2-dimensional lists we MUST have two indexes - one for the column position and one for the row position.

Tuple

A tuple is an immutable sequence - its contents CANNOT be changed. We have the built in *list* function to convert a tuple to a list, and the built-in *tuple* function to convert a list to a tuple.