Lists and Tuples
Today’s Topics

- Sequences
- Introduction to Lists
- List Slicing
- Finding Items in Lists with the in Operator
- List Methods and Useful Built-in Functions
- Copying Lists
- Processing Lists
- Two-Dimensional Lists
- Tuples
- Plotting List Data with the matplotlib Package
Lists

- We’ve learned about lists already. We now talk about it in more detail, and adds some new things as well.
Sequences

• **Sequence**: an object that contains **multiple items** of data. For instance:
  
  • `my_list = [6, 78, 9]` is an example of a sequence.
    
    • The distinctive name of this sequence is `list`.
    
    • So list is a type of sequence.
    
    • The items are stored in sequence one after another.

• Python provides different types of sequences, including lists and tuples.
  
  • The difference between these is that:
    
    • a list is **mutable**
    
    • a tuple is **immutable**
Lists

- **List**: an object that contains multiple data items separated by a comma.
  - An data item in a list is called an **Element**.
  - **Format**: `list = [item1, item2, etc.]`
  - A list can hold items of different types.
    - `my_list = [7, "Ted", [56, 78]]`
      - Contains three elements of type int, str and list.

- The **print** function can be used to display an entire list.
- The **list()** function can convert certain types of objects to lists.
  - For instance, to convert a tuple into a list.
The Repetition Operator and Iterating over a List

- **Repetition operator**: makes multiple copies of a list and joins them together
  - The * symbol is a repetition operator when applied to a sequence and an integer.
    - Sequence is left operand, number is right
  - General format: `list * n`
- You can iterate over a list using a `for` loop
  - Format: `for x in list:`
Indexing

- **Index**: a number specifying the position of an element in a list
  - Enables access to individual element in list
  - Index of first element in the list is 0, second element is 1, and n’th element is n-1
  - Negative indexes identify positions relative to the end of the list
    - The index -1 identifies the last element, -2 identifies the next to last element, etc.
The `len` function

- An `IndexError` exception is raised if an invalid index is used.
- `len` function: returns the length of a sequence such as a list
  - Example: `size = len(my_list)`
  - Returns the number of elements in the list, so the index of last element is `len(list) - 1`
  - Can be used to prevent an `IndexError` exception when iterating over a list with a loop.
    - `for i in range(len(my_list)):`

Lists Are Mutable

- Mutable sequence: the items in the sequence can be changed
  - Lists are mutable, and so their elements can be changed

- An expression such as

- `list[1] = new_value` can be used to assign a new value to a list element.

  - Must use a valid index to prevent raising of an `IndexError` exception
Concatenating Lists

- **Concatenate**: join two things together.
- The + operator can be used to concatenate two lists.
  - Cannot concatenate a list with another data type, such as a number.
- The += augmented assignment operator can also be used to concatenate lists.
To be continued on Monday...