Sequence types:
Tuples, Lists, and Strings

Sequence Types

1. Tuple
   - A simple ***immutable*** ordered sequence of items
   - Items can be of mixed types, including collection types

2. Strings
   - ***Immutable***
   - **Conceptually very much like a tuple**

3. List
   - ***Mutable*** ordered sequence of items of mixed types

Similar Syntax

- **All three sequence types (tuples, strings, and lists) share much of the same syntax and functionality.**

- Key difference:
  - **Tuples and strings are *immutable***
  - **Lists are *mutable***
- The operations shown in this section can be applied to *all* sequence types
  - **most examples will just show the operation performed on one**

Sequence Types 1

- **Tuples are defined using parentheses (and commas).**

```
>>> tu = (23, 'abc', 4.56, (2,3), 'def')
```

- **Lists are defined using square brackets (and commas).**

```
>>> li = ["abc", 34, 4.34, 23]
```

- **Strings are defined using quotes (", ', or """").**

```
>>> st = "Hello World"
>>> st = 'Hello World'
>>> st = """This is a multi-line
string that uses triple quotes."""
```

Sequence Types 2

- **We can access individual members of a tuple, list, or string using square bracket “array” notation.**
- ***Note that all are 0 based…***

```
>>> tu = (23, 'abc', 4.56, (2,3), 'def')
>>> tu[1]      # Second item in the tuple.
 'abc'

>>> li = ["abc", 34, 4.34, 23]
>>> li[1]       # Second item in the list.
 34

>>> st = "Hello World"
>>> st[1]    # Second character in string.
 'e'
```

Positive and negative indices

```
>>> t = (23, 'abc', 4.56, (2,3), 'def')
```

Positive index: count from the left, starting with 0.

```
>>> t[1]
'abc'
```

Negative lookup: count from right, starting with –1.

```
>>> t[-3]
4.56
```

Slicing: Return Copy of a Subset 1

```
>>> t = (23, 'abc', 4.56, (2,3), 'def')
```

Return a copy of the container with a subset of the original members. Start copying at the first index, and stop copying *before* the second index.

```
>>> t[1:4]
('abc', 4.56, (2,3))
```

You can also use negative indices when slicing.

```
>>> t[1:-1]
('abc', 4.56, (2,3))
```

Slicing: Return Copy of a Subset 2

```
>>> t = (23, 'abc', 4.56, (2,3), 'def')
```

Omit the first index to make a copy starting from the beginning of the container.

```
>>> t[:2]
(23, 'abc')
```

Omit the second index to make a copy starting at the first index and going to the end of the container.

```
>>> t[2:]
(4.56, (2,3), 'def')
```