Tuple

Need for tuple:

Lists work well for storing sets of items that can change throughout the life of a program. The ability to modify lists is particularly important when you're working with a list of users on a website or a list of characters in a game. However, sometimes you'll want to create a list of items that cannot change. Tuples allow you to do just that. Python refers to values that cannot change as *immutable*, and an immutable list is called a *tuple*.

Defining a tuple:

A tuple looks just like a list except you use parentheses instead of square brackets. Once you define a tuple, you can access individual elements by using each item's index, just as you would for a list.
Example:

```
colors = ('pink', 'red')
print(colors[0])
print(colors[1])
```

OUTPUT:

```
pink
red
```

In the above code if we try to change the contents of the tuple this will give us an error.

```
colors[0] = 'green'
```

OUTPUT:

```
Traceback (most recent call last):
File "dimensions.py", line 3, in <module>
dimensions[0] = 250
TypeError: 'tuple' object does not support item assignment
```

Looping Through All Values in a Tuple

You can loop over all the values in a tuple using a for loop, just as you did with a list:

```
dimensions = (200, 50)
for dimension in dimensions:
        print(dimension)
```

OUTPUT:
200
50

Writing over a Tuple

Although you can't modify a tuple, you can assign a new value to a variable that holds a tuple. So if we wanted to change our dimensions, we could redefine the entire tuple:

```
dimensions = (200, 50)
print("Original dimensions:")
for dimension in dimensions:
        print(dimension)

dimensions = (400, 100)
print("\nModified dimensions:")
for dimension in dimensions:
        print(dimension)
```

OUTPUT:
Original dimensions:
200
50
Modified dimensions:
400
100

SETS

A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed).

However, a set itself is mutable. We can add or remove items from it.

Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc

Creating Python Sets

A set is created by placing all the items (elements) inside curly braces `{}`, separated by comma, or by using the built-in `set()` function.

It can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have mutable elements like lists, sets of dictionaries as its elements.

```
# Different types of sets in Python
# set of integers
my_set = {1, 2, 3}
print(my_set)


# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

OUTPUT:

```
{1, 2, 3}
{1.0, (1, 2, 3), 'Hello'}
```

PLEASE GO THROUGH THE LINKS IN THE RESOURCE DOCUMENT TO FIND MORE ABOUT SETS AND TUPLE.