Monroe Township Library Coding Bootcamp

Class 4 Notes

* Complex data types
* Lists
* Tuples
* Sets
* Dictionaries

**Complex data types:**

* There are several more data types in Python that can be comprised of other primitive data types
  + For example, a list is a data type itself but it can contain many other data types including strings, integers, Booleans, and even other lists
* Complex data types are **iterable**meaning we can loop through each element using a for loop

**Lists:**

* A list is simply a collection of data surrounded by square brackets and separated by a comma
* Lists can contain values of *any* type (strings, integers, booleans, etc.)
  + Lists can also contain other lists, this is called nesting
* Lists are also **ordered,** which means we can access specific values by using their index (starting from 0)
  + To access the first element of the list, use the syntax list\_name[0]
* You can add items to a list using the append() method, and remove items using the pop() method (by index) or remove() method (by element)
* Lists are **mutable**, meaning the values within can be changed
  + This is important to remember since there are cases where you may change values in a list unintentionally; if you are working with a list and you want to keep the original values, you can make a copy of the list using the copy() method
* You can also use the in keyword to construct conditional statements
  + If the element is present in the list, will evaluate to True, otherwise False

**Tuples:**

* Tuples function very similarly to lists, they can contain values of any type and are created using parentheses with values separated by a comma
* Unlike lists though, tuples are **immutable,** meaning its contents cannot be changed
  + This also means that elements can’t be added or removed, so there is no append() or pop() method for tuples
* Tuples are still **ordered** and can be indexed the same way as lists

**Sets:**

* Sets also store a collection of elements like lists or tuples, however sets do not allow for duplicate values
* Sets are initialized using curly brackets, elements are separated with a comma
  + If you initialize a set that contains any number of duplicate values, those duplicates will be removed immediately but will not cause an error
* Sets are **unordered,** which means that elements of a set are not necessarily stored in the order you initialized them, and they also cannot be accessed by indexing
* You can add or remove elements from a set using the add() and discard() methods
* You can also combine two separate sets using the union() method, duplicates will be removed from the new set
* If you need to make sure there are no duplicates in a list, converting it to a set using the set() function will automatically remove any duplicates

**Dictionaries:**

* Dictionaries store a collection of key-value pairs
* Dictionaries are initialized using curly braces, where each pair consists of a key followed by a colon followed by a value (key : value), each pair is separated by a comma
* Values in a dictionary are accessed by referencing their key
  + So dictionary[key] would return the associated value in the pair
  + We can also use the get() method to either retrieve the associated key if it is in the dictionary, or return something else if it’s not found
  + Dictionaries are technically **ordered** (in current Python versions) but cannot be indexed like lists or tuples
* Dictionaries are **mutable**, meaning their values can be changed like lists, you can also make a copy using the copy() method
* You can add new key-value pairs using the update() method, and pairs can be removed using the pop() method

**Project: Employee Database**

**Check class files at** [**github.com/monroecoding**](https://github.com/monroecoding)

Chris DiFazio

Adult Services Librarian

cdifazio@monroetpl.org