**Week 6 – Python Data Structures**

There are four basic data structures in the Python programming language:

1. **List** is a collection which is ordered and changeable. Allows duplicate members. (like a vector)
2. **Dictionary** is a collection which is ordered\*\* and changeable. No duplicate members. (like lists)
3. **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
4. **Set** is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members. (good for removing duplicates, figuring out what is same/different between two groups)

**Lists**

Lists in Python are dynamically sized arrays much like the vectors we used in R. This essentially a collection of things enclosed in brackets `[ ]` and separated by commas. Lists are the simplest containers that are an integral part of the Python language. Lists need not be homogenous always which makes it the most powerful tool in Python. A single list may contain DataTypes like integers, strings, as well as objects. Lists are mutable, and hence, they can be altered even after their creation.

**Creating Lists**

Creating lists in Python is straight forward. You can create an empty list using brackets or create a list with elements as shown in the below example.

Text

Description automatically generated

**Combining Lists**

Python makes it easy to combine lists, and similarly other objects like strings with the `+` operator.

A screenshot of a computer

Description automatically generated with medium confidence

**Adding and Removing List Elements**

The append() method allows you to add elements to a list. The pop() method allows you to remove elements from a list by its index value and then returns the item in that index position. This is very useful for mixed object lists as shown in the below example.

**Text

Description automatically generated**

**Sorting Lists**

Once your lists are all of the same type you can use the sort method to re-arrange the values. By default, this works in ascending order but you can change that by specifying an argument. See the documentation for details.

Text

Description automatically generated

**Using Remove()**

We’ve shown that you can remove list elements using pop() but sometimes you want to remove them by name because you may be multiple occurrences of an item.

Text

Description automatically generated

**List Length**

Finding the length of lists is especially useful for iterating over lists or comparing the size of lists for unit tests or some other logical expression.

Text

Description automatically generated

**Inserting Elements at a Specific Index**

Append inserts list elements at the end or beginning of a list. But what if we wanted to insert an element in another location? The insert() method allows us to do that by providing an index, and the item to insert.

Text

Description automatically generated

**Subsetting Lists**

Subsetting lists in Python is identical to subsetting lists in R, the difference being the zero-based index. Python also uses the same bracket notation to allow you to subset strings which can be very powerful. In the below example we will correct the uncapitalize ‘b’ in the item we just added to our list in place.

Text

Description automatically generated

**Dictionaries & Defaultdict**

Dictionaries are one of the most powerful data structures in Python. They are a key-value pair-based structure that is easily serializable to JSON (JavaScript Object Notation), a data format that is widely used in big data and web-based applications. In a Dictionary, the key must be unique and immutable. This means that a Python Tuple can be a key whereas a Python List cannot. A Dictionary can be created by placing a sequence of elements within curly {} braces, separated by ‘comma’.

Text

Description automatically generated

Notice in that example that we received an error that the key 4 was not present in our dictionary. For some scenarios that may become a problem (hashing). To overcome this Python introduces another dictionary like container known as Defaultdict which is present inside the collections module.

**Defaultdict**

Defaultdict is a container like dictionaries present in the module collections. Defaultdict is a sub-class of the dictionary class that returns a dictionary-like object. The functionality of both dictionaries and defaultdict are almost same except for the fact that defaultdict never raises a KeyError. It provides a default value for the key that does not exists which means that you can use defaultdict for hash tables.

Text

Description automatically generated

**Merging dictionaries with the .update() method in Python**

Given two dictionaries that need to be combined, Python makes this easy with the .update() function.

For dict1.update(dict2), the key-value pairs of dict2 will be written into the dict1 dictionary.

For keys in both dict1 and dict2, the value in dict1 will be overwritten by the corresponding value in dict2.

Text

Description automatically generated with medium confidence

**Dictionary Key-Value Methods**

When trying to look at the information in a Python dictionary, there are multiple methods that return objects that contain the dictionary keys and values.

**.keys()** returns the keys through a dict\_keys object.

**.values()** returns the values through a dict\_values object.

**.items()** returns both the keys and values through a dict\_items object.

Text

Description automatically generated

**get() Method for Dictionary**

Python provides a **.get()** method to access a dictionary value if it exists. This method takes the key as the first argument and an optional default value as the second argument, and it returns the value for the specified key if key is in the dictionary. If the second argument is not specified and key is not found, then None is returned.

Text

Description automatically generated

**The .pop() Method for Dictionaries in Python**

Python dictionaries can remove key-value pairs with the **.pop()** method. The method takes a key as an argument and removes it from the dictionary. At the same time, it also returns the value that it removes from the dictionary.

Logo

Description automatically generated

**Tuples**

Tuples are used to store multiple items in a single variable. A tuple is a collection which is *ordered* and *immutable*. Tuples are written with round brackets. Since they are immutable there are no methods to be covered (.pop(), .append(), etc) aside from merging tuples. So why would you use a tuple and not a list? Well, if you want some data that isn’t changeable or doesn’t need to have its order changed, you could use a tuple. And when we talk about larger amounts of data, python executes tuples much faster.

Text

Description automatically generated

**Sets**

A set is an unordered collection with no duplicate elements. By unordered we mean it cannot be indexed. Basic uses include membership testing and eliminating duplicate entries. Set objects also support mathematical operations like union, intersection, difference, and symmetric difference. A practical use for a set is to quickly find all of the unique items of a list.

Text

Description automatically generated

**Set add() and update()**

Using the add() method we can add a single element to the set. With update() we can add multiple elements.

Text

Description automatically generated

**Set remove() and discard()**

Both methods will remove an element from the set but remove() will raise a key error if the value doesn't exist. Discard() will not raise any errors.

Text

Description automatically generated

**set union()**

union() or | will create a new set that contains all the elements from the sets provided.

Logo

Description automatically generated with low confidence

**set intersection**

Intersection or & will return a set containing only the elements that are common to all of them.

Text

Description automatically generated

**set difference**

difference or - will return only the elements that are unique to the first set (invoked set).

A screenshot of a computer

Description automatically generated with medium confidence

**set symmetric\_difference**

symmetric\_difference or ^ will return all the elements that are not common between them.

A screenshot of a computer

Description automatically generated with medium confidence