CS-500 Module Four Journal

Isaac Raymond

CS-500-11462-M01

Professor Ronak Nouri

The Use of Data Structures in Programming

Our programs have to store information at some point. That information might be data that needs to be held long-term, or it might be something used short-term in order to make the logic of a program function correctly. If there are several similar pieces of information that are similar, we will want to store that information in a particular structure. The particular data structure we choose for each situation depends on how that data will be used.

Lists are data structures that order the values and allow us to change these values in our program. Lists are usually employed when downloading a data set from a database, then perhaps cleaning that data and then using it in a statistical analysis or sending it to another program for further use. An example of the use of lists would be when a data set of temperature readings for certain days would be downloaded then loaded into a python program. For example:

temperature = [87.3, 90.3, 90.1, ...]

The second temperature in the list, 90.3, could be accessed as follows:

temperature = [1]

Tuples are data structures that are also ordered like lists, but they are unchangeable. Tuples should be used in circumstances where we do not want to modify the values of the data structure. The above example of temperature readings would not be a good example for a tuple because we often want to clean a data set downloaded from a database. An example of a tuple might be the geographic coordinates of each region of the temperature database downloaded.

Those regions won't change. The tuple can store their GPS coordinates. An example:

```
gps_coordinates = (10.324, 32.323)
```

We could access the longitude and latitude values of these coordinates similar to the syntax of a list:

```
gps_coordinates[0]
```

would give us 10.324

A dictionary should be employed in circumstances similar to a list, except a dictionary allows for fast lookup of values in the data structure. Unlike lists, where values can be accessed using indexing, dictionaries allow us to store data through key-value pairs. A specific string for a key can quickly allow a programmer to get the value they are looking for. For example:

```
temperature_by_city = {
"Boston" : 73,
"New York" : 78,
"Miami" : 88,
...
}
```

If a user wanted their code to get the temperature of Boston, they just need to reference the key "Boston" in the dictionary:

```
temperature_by_city["Boston"]
```

If the temperatures were stored in an "n" by 1 array, we would have to remember which index boston was in the array in order to get the temperature.

The choice of a data structure requires foresight. We need to know how our data will be used later on as our project develops. A lot of code has to be rewritten if we decide that our temperature data set needs to be stored as key-value pairs as in a dictionary. It can be difficult to know how these data structures will be used as our program expands. As a programmer works on more projects, they will develop an intuition for what data structure is needed for each situation and will be able to plan accordingly.