Dictionaries Defined

Dictionaries are another data structure in Python. They're similar to a list in that they can be used to organize data into collections. However, data in a dictionary isn't accessed based on its position. Data in a dictionary is organized into pairs of keys and values. You use the key to access the corresponding value. Where a list index is always a number, a dictionary key can be a different data type, like a string, integer, float, or even tuples.

When creating a dictionary, you use curly brackets: **{}**. When storing values in a dictionary, the key is specified first, followed by the corresponding value, separated by a colon. For example, **animals = { "bears":10, "lions":1, "tigers":2 }** creates a dictionary with three key value pairs, stored in the variable animals. The key "bears" points to the integer value 10, while the key "lions" points to the integer value 1, and "tigers" points to the integer 2. You can access the values by referencing the key, like this: **animals["bears"]**. This would return the integer 10, since that's the corresponding value for this key.

You can also check if a key is contained in a dictionary using the **in** keyword. Just like other uses of this keyword, it will return True if the key is found in the dictionary; otherwise it will return False.

Dictionaries are mutable, meaning they can be modified by adding, removing, and replacing elements in a dictionary, similar to lists. You can add a new key value pair to a dictionary by assigning a value to the key, like this: **animals["zebras"] = 2**. This creates the new key in the animal dictionary called zebras, and stores the value 2. You can modify the value of an existing key by doing the same thing. So **animals["bears"] = 11** would change the value stored in the bears key from 10 to 11. Lastly, you can remove elements from a dictionary by using the **del** keyword. By doing **del animals["lions"]** you would remove the key value pair from the animals dictionary.