

4.4 Dictionaries

Look Up Ability

A regular dictionary is one where the user inputs some value (a word), and receives some answer/translation (a definition). A pair of lists or a list of pairs could be used to maintain a dictionary, but in Python, a special dictionary type is included to simplify this sort of look-up ability. Whereas a list must be indexed by a 0-based integer, a dictionary is indexed with a word, or anything you want it to be indexed by! To retrieve an item, simply index it with its corresponding key.

Initializing a Dictionary

We create an empty dictionary as follows:

```
phonebook = {}
```

We can create an initial dictionary with entries as follows:

```
phonebook = {"Adam":5551234, "Carol":5559999}
```

Thus, in general, each item is separated with a comma, and within each item, the key (input value) comes first, followed by a colon, followed by the value (output value) to which it maps.

Adding an Entry into a Dictionary

We can add an entry as follows:

```
phonebook["Dave"] = 5553456
```

Accessing a Value

To access a value stored in a dictionary, simply index the dictionary with the appropriate key:

```
name = input("Whose phone number do you want to look up?\n")
print(name, "'s number is ", phonebook[name], sep="")
```

For example, if we enter Carol for the name, we get the following output:

```
Carol's number is 5559999
```

Invalid Key Error

If we attempt the following:

```
print(phonebook["Bob"])
```

We get the following error:

```
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    phonebook["Bob"]
KeyError: 'Bob'
```

Thus, it's important NOT to attempt to access a dictionary entry that isn't there. Here is a segment of code that makes sure an invalid access does not occur:

```
def main():

    phonebook = {"Adam":5551234, "Carol":5559999}
    name = input("Whose phone number do you want to look up?\n")

    if name in phonebook:
        print(name, "'s number is ", phonebook[name], sep="")
    else:
        print("Sorry, I do not have a number for ", name, ".",
              sep="")

main()
```

This sort of error is similar to an index out of bounds error for strings and lists. We can avoid such errors by checking our indexes, or in this case, our key, before we ever use it to access our dictionary.

It's important to note that although we just used a dictionary to link names to phone numbers in this extended example, we can use a dictionary to link any one type of object to another type of object.

Changing a Dictionary Entry

This can be done as expected, via the assignment operator:

```
phonebook["Adam"] = 5557654
```

When Python discovers that there's an entry for "Adam" already, it will simply replace the old corresponding value, 5551234, with this new one, 5557654.

Deleting a Dictionary Entry

While deletion in a regular dictionary doesn't make sense all that often because words don't usually cease to exist, in many contexts deleting items from a dictionary makes sense. For example, if we are no longer friends with someone, we may want to delete their entry from our phone book. We can accomplish this with the `del` command. We follow the keyword `del` with the name of our dictionary indexed at the entry we want to delete. This following trace through in IDLE illustrates the use of the `del` command:

```
>>> phonebook = {"Adam":5551234, "Carol":5559999}
>>> phonebook["Bob"] = 5556666
>>> phonebook
{'Bob': 5556666, 'Carol': 5559999, 'Adam': 5551234}
>>> phonebook["Adam"] = 5557654
>>> phonebook["Adam"]
5557654
>>> del phonebook["Adam"]
>>> phonebook
{'Bob': 5556666, 'Carol': 5559999}
```