

DICTIONARY PATTERNS

An Introduction to Computer Science



Let's learn about Dictionary Patterns.

Using Dictionaries

- Lookup
- Record
- Counting



Dictionaries are very useful for a broad number of applications.
However, it can be tricky to understand how to use them.
In this lesson, you will see some common patterns for how to use Dictionaries.

Lookup Pattern

```
abbreviations = {"Virginia": "VA",  
                 "Delaware": "DE",  
                 "Maryland": "MD",  
                 "North Carolina": "NC"}  
  
>>> abbreviations["Virginia"]  
'VA'
```



Dictionaries can be used to quickly convert one value into another.

Consider the example here; this dictionary can be used to words to their abbreviation.

Note that this could have also been implemented as a function with an IF statement.

However, if you are only looking up a single value, then the dictionary will be much easier and faster.

Although this example uses strings, you can really map any type of data to any other type of data in a lookup dictionary.

Record Pattern

```
first_person = {"name": "Cory",
                "age": 27,
                "has a dog?": True}

second_person = {"name": "Adam",
                 "age": 45,
                 "has a dog?": False}

>>> first_person["name"]
'Cory'
>>> second_person["age"]
45
```



With the Record Pattern, dictionaries become a useful way to hold data that represents a single complex thing.

In the example below, we represent a person using a dictionary with 3 keys.

Notice that in this pattern, each key is a string.

The values can be whatever type of data is useful, but they will always have the same type as other records that represent that complex thing.

By convention, all records meant to represent the same type of real world thing (like a person) will have the same keys with the same spelling.

Counting Pattern

```
a_list = [1, 3, 1, 1, 3, 2, 1, 3, 2, 3]
counts = {}
for item in a_list:
    if item in counts:
        counts[item] = counts[item] + 1
    else:
        counts[item] = 1
```

Empty dictionary

Setting a
dictionary

Using a variable
as a key



We have previously used the Counting pattern to count the number of occurrences of an item in a list.

But what if we want to count the occurrences for each possible item in a list, without knowing the items before hand?

By adding in a dictionary, we can do just that.

The counting pattern results in a dictionary of any kind of keys mapped to integers.

Notice that dictionary access can be used to update a value, in addition to getting a value. When the dictionary access occurs on the left side of an assignment, the value associated with that key will be replaced.

Moreover, you can see that we are using a variable to access elements, instead of a string literal.