DICTIONARIES

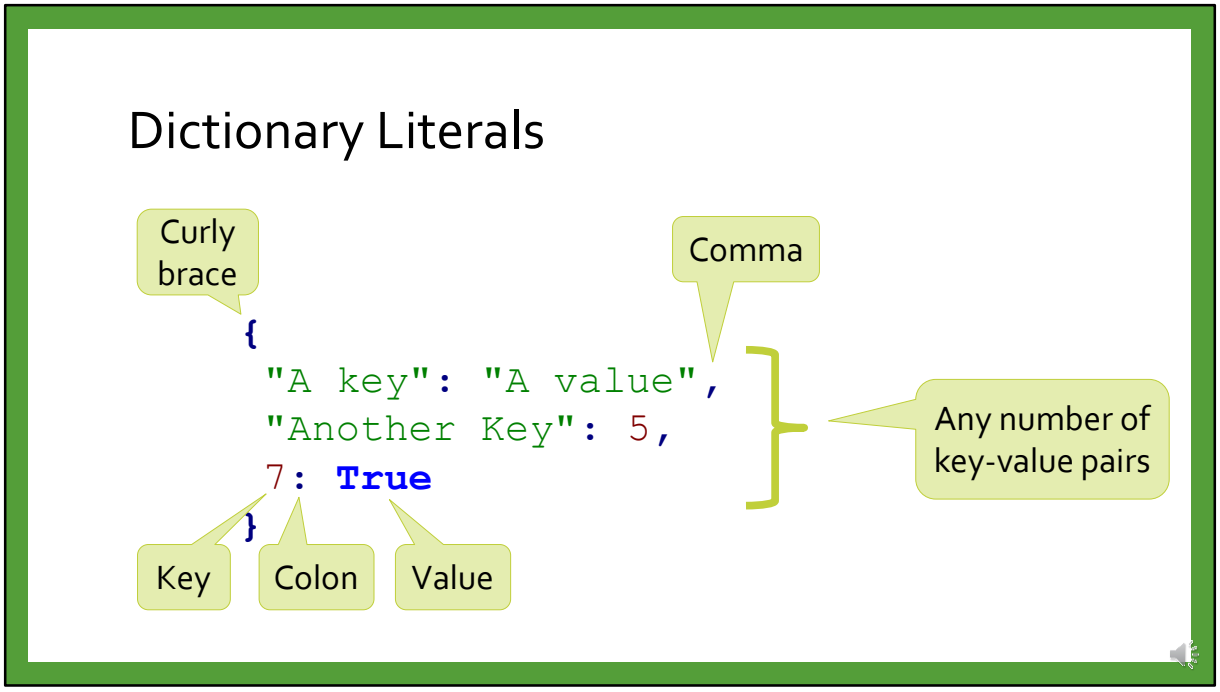
An Introduction to Computer Science

Let's learn about Dictionaries.

Dictionaries

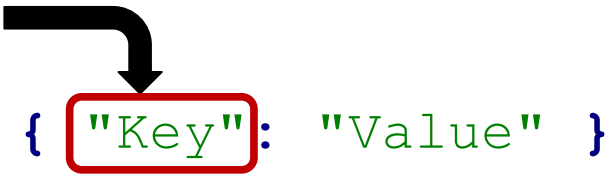
Aardvark	➔	A nocturnal burrowing mammal with long ears, a tubular snout, and a long extensible tongue, feeding on ants and termites.
Able	➔	having the power, skill, means, or opportunity to do something.
About	➔	on the subject of; concerning.
…	➔	…

Dictionaries are one of the most flexible and useful data structures in programming.
The idea of a dictionary is that they map from keys to values.
Think of it like an actual dictionary: given a word, you can look up its definition.
This lookup has a large number of applications.

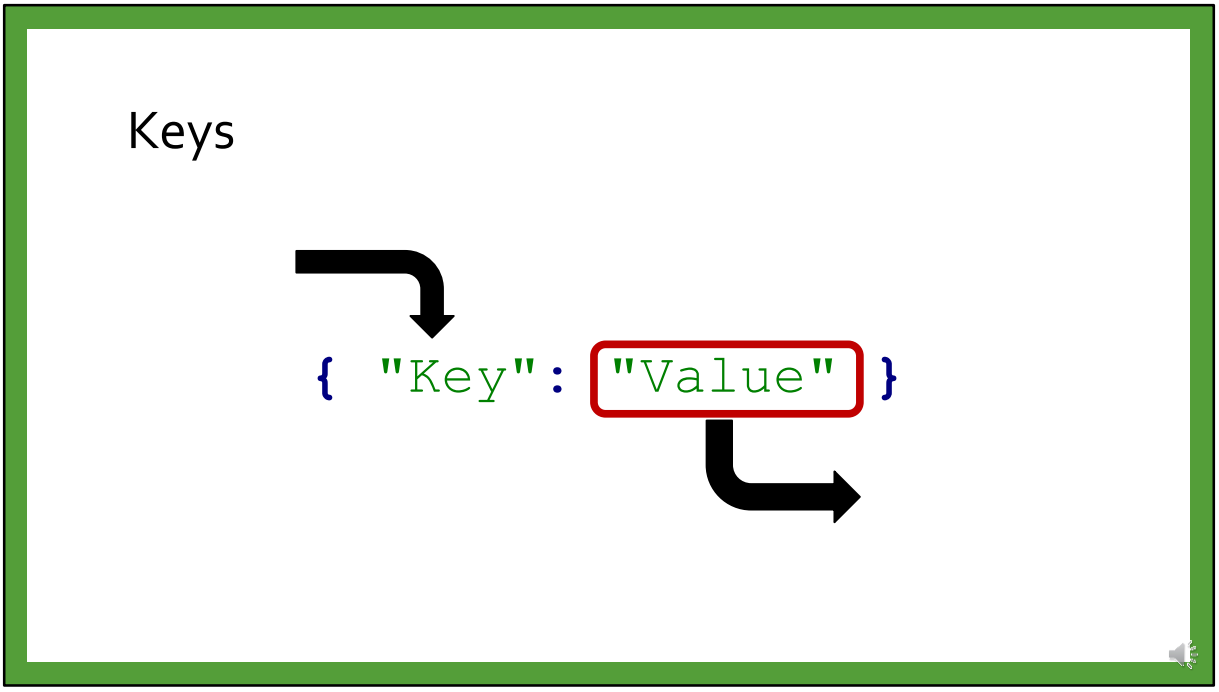


The syntax for a dictionary literal begins with a pair of curly braces.
Inside of those braces, you will find key/value pairs separated by commas.
Each key is separated from its value with a colon.
The key and the value can be any expression, but are often literal values.
Although not required, each key-value pair is often placed on its own new line.

Keys



The keys of a dictionary are what we use to look up information.
The keys of a dictionary are unique.
Typically, the keys will be of the same type, although this is not a requirement.

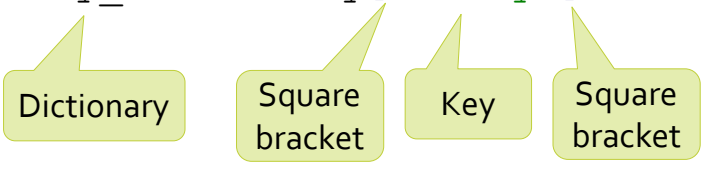


Each key is mapped to a value.
With the appropriate key, we can get a desired value.
Values do not have to be unique, unlike keys.
Like keys, the values can be of any type, even if they are not the same.

Dictionary Access

```
my_dictionary = {"A key": "A value",
                 "Another Key": 5,
                 7: True}
```

```
my_dictionary["A key"]
```



The values of a dictionary can be retrieved using dictionary access.
Another similarity between lists and dictionaries is how both use square brackets.
With lists, we used numeric indexes to access elements.
With dictionaries, we use the keys.
Often, the keys will be strings, so often we will see strings inside the square brackets.

Lookups

```
>>> pets = {"Klaus": "dog",
            "Pumpkin": "cat",
            "Wrex": "hamster"}

>>> pets["Klaus"]
"dog"

>>> pets["Spot"]
```

KeyError!

```
>>> pets["dog"]
```

KeyError!

When we perform dictionary access, we give a key, and the dictionary access expression is replaced with the corresponding value.
If the key is not in the dictionary, a "KeyError" is raised.
A common mistake is to use a value instead of a key, which will also cause a "KeyError".

Not Always Strings

```
>>> levels = {3: "High",
              2: "Medium",
              1: "Low"}

>>> levels[2]
"Medium"

>>> levels[0]
```

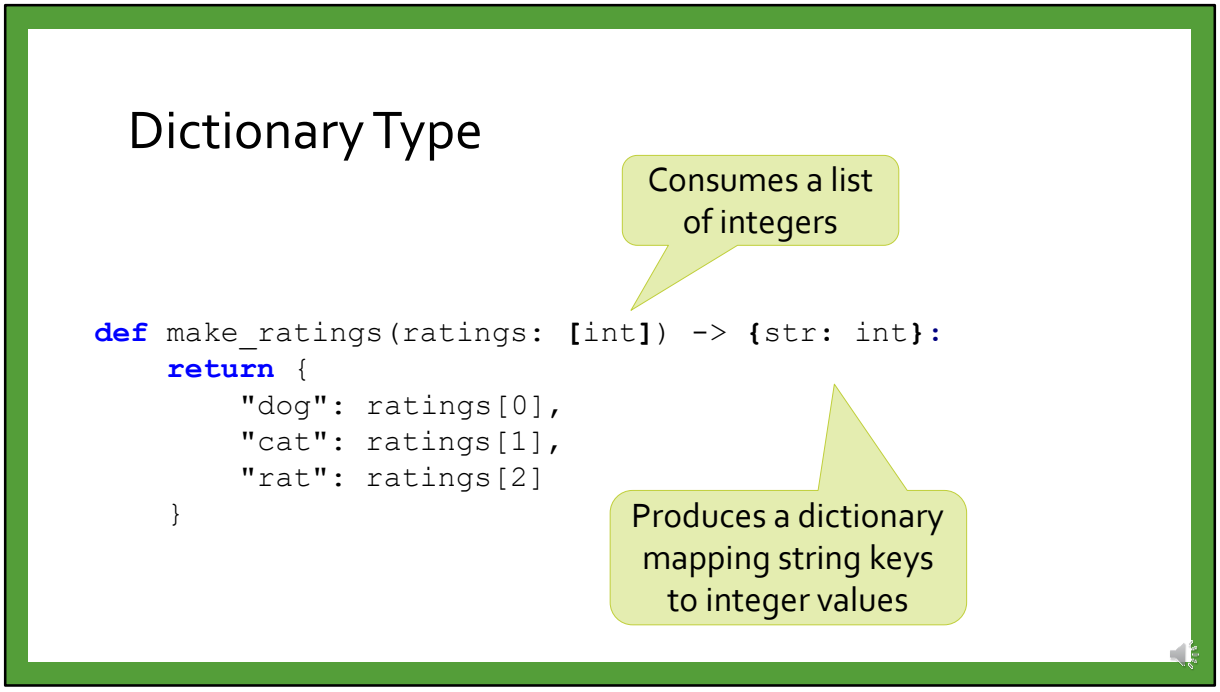
KeyError!

```
>>> levels["High"]
```

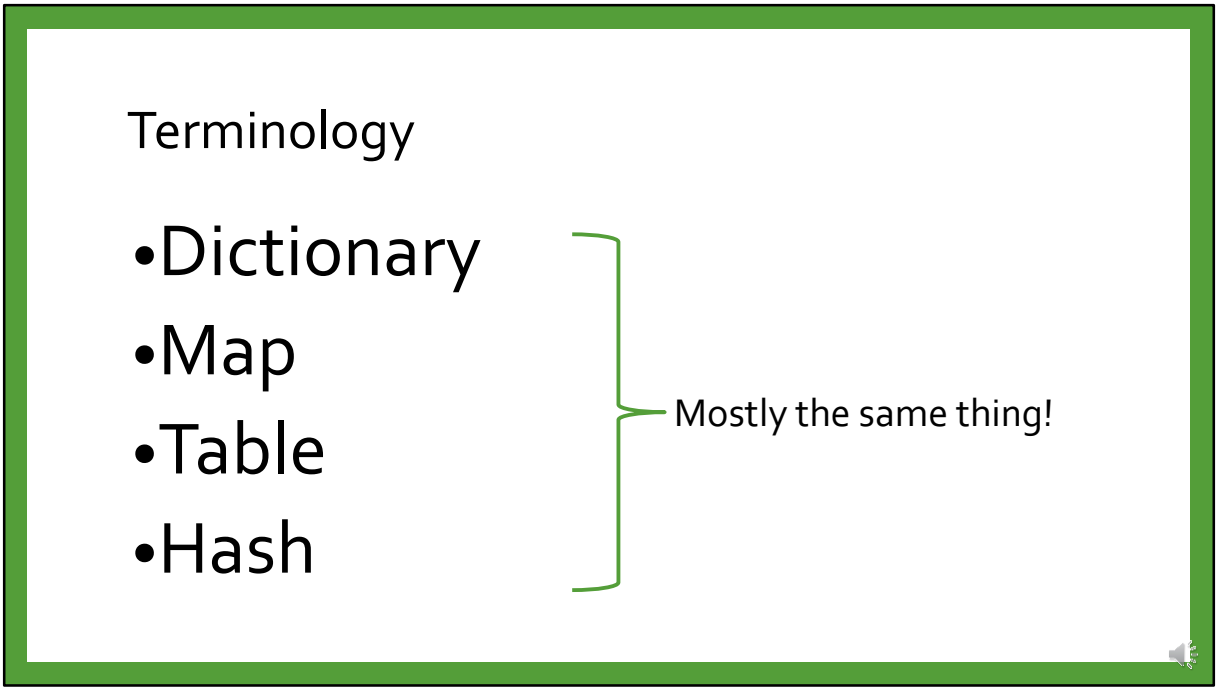
KeyError!

Sometimes keys will be strings, but it turns out that anything can be a key.
If a dictionary maps integers to strings, for instance, then the keys will be integers.
In fact, it is possible for a dictionary lookup to appear just like a list lookup!

Dictionary Type



When you define a function that consumes or produces a dictionary, we need to specify the type.
If we don't know anything specific about the dictionary's keys and values, we can use the the `dict` type.
However, if we know that the dictionary maps, say, strings to integers, then we can use a dictionary as the type.
The function shown here uses a list of integers to produce a dictionary that maps string keys to integer values.



A final minor note about dictionaries: dictionaries are also sometimes known as maps, tables, or hashes.
Although there are subtle differences in this terminology, the idea is still roughly the same each time.