Dictionaries

Dictionaries

So far we have only looked at sequence data structures, where elements are referred to by their position in the sequence. In dictionaries, however, the objects stored are referred to by a key. Keys must be an immutable type, for example a string, number or tuple containing only immutable types.

You can create a dictionary using the `dict` function; and assign values using the subscript notation:

```
dictionary[key] = value
```

[2]:
```
d = dict()

d[1] = 'a'
d['lst'] = [1, 2, 3]

print(d)
```

```
{1: 'a', 'lst': [1, 2, 3]}
```

You can also access dictionary values using the subscript notation:

[4]:
```
print(d[1])
```

```
a
```

An alternative way to initialize a dictionary with key-value pairs is:

```
{key1 : value1, key2 : value2}
```

much like it appears in the print output:

[6]:
```
d = {1 : 'a',  'lst' : [1, 2, 3]}

print(d)
```

```
{1: 'a', 'lst': [1, 2, 3]}
```

Note that using a key that doesn't exist in the dictionary will give you a `KeyError`:

[5]:
```
print(d[2])
```

```
␣
↪---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call␣
↪last)

<ipython-input-5-c8f93a31d4a2> in <module>
----> 1 print(d[2])


KeyError: 2
```

Listing the Keys Which Exist

Often you will want a list of the keys which a dictionary has. For this you can use the `dict.keys()` method:

[7]:
```
print(d.keys())
```

```
dict_keys([1, 'lst'])
```

One use for this is to check if a dictionary has the key you're looking for if you want to avoid an error:

[8]:
```
if 2 in d.keys():
    print(d[2])
else:
    print(2, 'not a key in d')
```

```
2 not a key in d
```