02_BooleansTuplesDictionaries

November 13, 2019

1 Booleans, Tuples, and Dictionaries

1.1 Booleans

A `boolean` is one of the simplest Python types, and it can have two values: `True` and `False` (with uppercase `T` and `F`):

```
[2]: a = True
     b = False
```

Booleans can be combined with logical operators to give other booleans:

```
[3]: True and False
```

```
[3]: False
```

```
[4]: True and True
```

```
[4]: True
```

```
[5]: False and False
```

```
[5]: False
```

```
[6]: True or False
```

```
[6]: True
```

```
[4]: (False and (True or False)) or (False and True)
```

```
[7]:   False or False
```

```
[7]: False
```

Standard comparison operators can also produce booleans:

```
[8]: 1 == 3
```

```
[8]: False
```

```
[9]: 1 != 3
```

```
[9]: True
```

```
[10]: 3 > 2
```

```
[10]: True
```

```
[11]: 3 <= 3.4
```

```
[11]: True
```

```
[13]: 3<3
```

```
[13]: False
```

1.2 Exercise 1

Write an expression that returns `True` if `x` is strictly greater than 3.4 and smaller or equal to 6.6, or if it is 2, and try changing `x` to see if it works:

```
[17]: x =2

      # your solution here
      ((x>3.4) and (x<=6.6)) or (x ==2)
```

```
[17]: True
```

1.3 Tuples

Tuples are, like lists, a type of sequence, but they use round parentheses rather than square brackets:

```
[18]: t = (1, 2, 3)
```

They can contain heterogeneous types like lists:

```
[19]: t = (1, 2.3, 'spam')
```

and also support item access and slicing like lists:

```
[20]: t[1]
```

```
[20]: 2.3
```

```
[21]: t[:2]
```

```
[21]: (1, 2.3)
```

The main difference is that they are **immutable**, like strings:

```
[22]: t[1] = 2
```

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

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-22-b9aa97991a78> in <module>
    ----> 1 t[1] = 2


        TypeError: 'tuple' object does not support item assignment
```

We will not go into the details right now of why this is useful, but you should know that these exist as you may encounter them in examples.

1.4 Dictionaries

One of the data types that we have not talked about yet is called *dictionaries* (`dict`). If you think about what a 'real' dictionary is, it is a list of words, and for each word is a definition. Similarly, in Python, we can assign definitions (or 'values'), to words (or 'keywords').

Dictionaries are defined using curly brackets `{}`:

```
[23]: d = {'a':1, 'b':2, 'c':3}
```

Items are accessed using square brackets and the 'key':

```
[24]: d['a']
```

```
[24]: 1
```

```
[25]: d['c']
```

```
[25]: 3
```

Values can also be set this way:

```
[26]: d['r'] = 2.2
```

```
[27]: print(d)
```

```
{'a': 1, 'b': 2, 'c': 3, 'r': 2.2}
```

The keywords don't have to be strings, they can be many (but not all) Python objects:

```
[28]: e = {}
      e['a_string'] = 3.3
      e[3445] = 2.2
```

```
[30]: print(e)
      f={'a_string': 3.3, 3445: 2.2}
```

```
{'a_string': 3.3, 3445: 2.2}
```

```
[31]: e[3445]
```

```
[31]: 2.2
```

If you try and access an element that does not exist, you will get a KeyError:

```
[32]: e[4]
```

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

        KeyError                                  Traceback (most recent call␣
 ↪last)

        <ipython-input-32-5c9c950b18e0> in <module>
    ----> 1 e[4]


        KeyError: 4
```

Also, note that dictionaries do *not* know about order, so there is no 'first' or 'last' element.

It is easy to check if a specific key is in a dictionary, using the in operator:

```
[33]: "a" in d
```

```
[33]: True
```

```
[34]: "t" in d
```

```
[34]: False
```

Note that this also works for lists:

```
[35]: 3 in [1,2,3]
```

```
[35]: True
```

1.5 Exercise 2

Try making a dictionary to translate a few English words into German and try using it!

```
[36]: # your solution here
```

```
[2]: a = {'hello':'Hallo', 'bye':'Auf Wiedersehen'}
     a['hello']
```

```
[2]: 'Hallo'
```