DICTIONARIES:

We've already used lists and strings to store and process data. Python also has a variable type called a **"dictionary"** that is similar to a list, but instead of having integer indices, you provide your own index, called a "key". You can assign data to a dictionary as follows: `phones = {'Zoe':'232-43-58', 'Alice':'165-88-56'}`. We can therefore think of a dictionary as a "function" that maps a collection of **keys (`'Zoe'`, `'Alice'`)** to **values(`'232-43-58'`, `'165-88-56'`)**.

As with lists, the **values** of the dictionary can be of any type: **strings, integers, floating point numbers, even lists or dictionaries** themselves. For **keys** you can use only **strings, numbers, floats and other immutable types**. Unlike lists, **accessing values of a dictionary is using the keys**:

```
phones = {'Zoe':'232-43-58', 'Alice':'165-88-56'}
print (phones['Zoe'])
```

Here, the output should be:

```
232-43-58
```

Adding new values to a dictionary or assigning a new value to an existing key can be done as follows:

```
phones['Zoe'] = '658-99-55'
phones['Bill'] = '342-18-25'
print (phones)
```

This should produce the following:

```
{'Bill': '342-18-25', 'Zoe': '658-99-55', 'Alice': '165-88-56'}
```

Note that the new `'Bill'` appeared in the beginning of the dictionary, not in the end, as you might expect. Dictionaries do not have an obvious ordering.

Remember that dictionaries are case-sensitive if you are using strings as keys. For example, 'key' and 'Key' are viewed as different keys:

```
d = {}
d['key'] = 1
d['Key'] = 2
d['KEY'] = 3
print (d)
```

Output:

```
{'KEY': 3, 'Key': 2, 'key': 1}
```

Note how we created an empty dictionary with `d = {}`. This could be useful in case you need to add values to dictionary dynamically (for example, when reading a file). If you need to check whether there a key in dictionary, you can use `key in d` syntax:

```
if 'Peter' in phones:
    print ("We know Peter's phone")
else:
    print ("We don't know Peter's phone")
```

Output:

```
We don't know Peter's phone
```

In case you need to delete a value from a dictionary, use the `del` command:

```
phones = {'Zoe':'232-43-58', 'Alice':'165-88-56'}
del phones['Zoe']
print (phones)
```

This produces the following output:

```
{'Alice': '165-88-56'}
```

For a pretty representation when outputting a dictionary, you can use the built in `.items()` function:

```
for key, value in phones.items():
    print (key)
    print (value)
```