`if` statements

In Python we can control the execution of code based on conditions with `if` statements.

`if` statements

Using `if`, we can execute part of a program conditional on some statement being true.

```python
traffic_light = 'green'
if traffic_light == 'green':
    print("move car!")
```

Note on indentation

In Python, blocks of code are defined using indentation.

This means that everything indented after an `if` statement is only executed if the statement is `True`.

If the statement is `False`, the program skips all indented code and resumes at the first line of unindented code.

```python
statement = False
if statement:
    # if statement is True, then all code here
    # gets executed but not if statement is False
    print("The statement is true")
    print("Else, this would not be printed")
# the next lines get executed either way
print("Hello, world,")
print("Bye, world!")
```

`if-else` statements

We can add more conditions to the `if` statement using `else` and `elif` (short for else if):

```python
traffic_light = 'red'
if traffic_light == 'green':
    print("drive")
elif traffic_light == 'orange':
    # executed if conditional statement is True and previous [el]if statements
    # are false
    print("better drive faster")
else:
    # executed if all other [el]if statements return False
    print("maybe stop")
```

What is `True`?

Non-boolean values can be used in an `if` statement. It is important to know which values are considered `True` versus `False`.

```python
if "am I True?":
    print("Yes! :)")
```

```
else:
    print("No :(")
```

Non-empty strings are considered `True` while empty strings are `False`.

In Python the following values are interpreted as `False`: * `False` * `None` * numeric zero of all numeric types * empty strings * empty containers (including strings, tuples, lists, dictionaries, sets and frozensets)

All other values are interpreted as `True`.