0508 Paper - Loops

July 23, 2018

1 Paper: Loops

Loops allow you to ensure that certain parts of the program are executed multiple times. This is useful, for example, if you have a list of students and want to make sure that a print() function is executed once for each student.

A loop allows you to express that in very little code!

1.1 The for loop

Besides the while loop, which we have already got to know, there is also the **for loop**. Here, a loop variable runs through the values one after the other in a sequence that is also to be specified.

This sequence can be a list, for example:

```
In [3]: list1 = [5, 8, 10]
        for i in list1:
            print(i)
```

```
5
8
10
```

```
In [1]: list1 = ["Max", "Moritz", "Monika"]
        for i in list1:
            print(i)
```

```
Max
Moritz
Monika
```

We see that our loop variable i takes the values from the list one after the other and automatically.

1.1.1 The range object

You do not necessarily need a list as a sequence for a for loop. Often an **range** object is used instead:

```
In [1]: print(range(0,10))
```

```
range(0, 10)
```

```
In [2]: for i in range(0, 10):
            print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
In [2]: # Here we sum up all numbers from 1 to 10 using a for loop and a range object
        sum = 0
        for i in range(1, 11):
            sum += i
        print(sum)
```

```
55
```

1.2 The while loop

A code block within an if-elif-else structure is executed only once. In loops such as the **while loop**, a code block is executed several times in succession until a termination condition is fulfilled:

```
In [3]: counter = 0

        while counter < 10:
            print(counter)
            counter = counter + 1

        print("Hello World")
```

```
0
1
2
```

```
3
4
5
6
7
8
9
Hello World
```

Within a loop, a state **must** change in each step so that the loop condition is not fulfilled permanently and the program can exit the loop again.

```
In [3]: students = ["Moritz", "Klara", "Monika", "Max"]
        i = 0
        while i < len(students):
            print(students[i])
            i = i + 1
```

```
Moritz
Klara
Monika
Max
```

2 Continue & Break

During a loop pass, we can abort the current pass prematurely and continue immediately with the next loop pass (**continue**) or abort the entire loop (**break**).

2.0.1 Continue

We simply need to write the word **continue** in a loop if we want to jump to the new loop pass at a certain point:

```
In [5]: for i in range(0, 10):
            if i == 3:
                continue
            print(i)
```

```
0
1
2
4
5
6
7
8
9
```

In the above example, the print() function is skipped for the value 3.

```
In [3]: for i in range(1, 10):
            print(i)
```

```
1
2
3
4
5
6
7
8
9
```

2.0.2 Break

We also simply write **break** in one line and the whole loop is aborted when the program reaches this point:

```
In [4]: for i in range(0, 10):
            if i == 3:
                break
            print(i)
```

```
0
1
2
```

```
In [4]: list1 = [4, 6, 7, 2, 4, 6, 7]

        s = 0

        for element in list1:
            s = s + element
            if s > 10:
                break

        print(s)
```

```
17
```