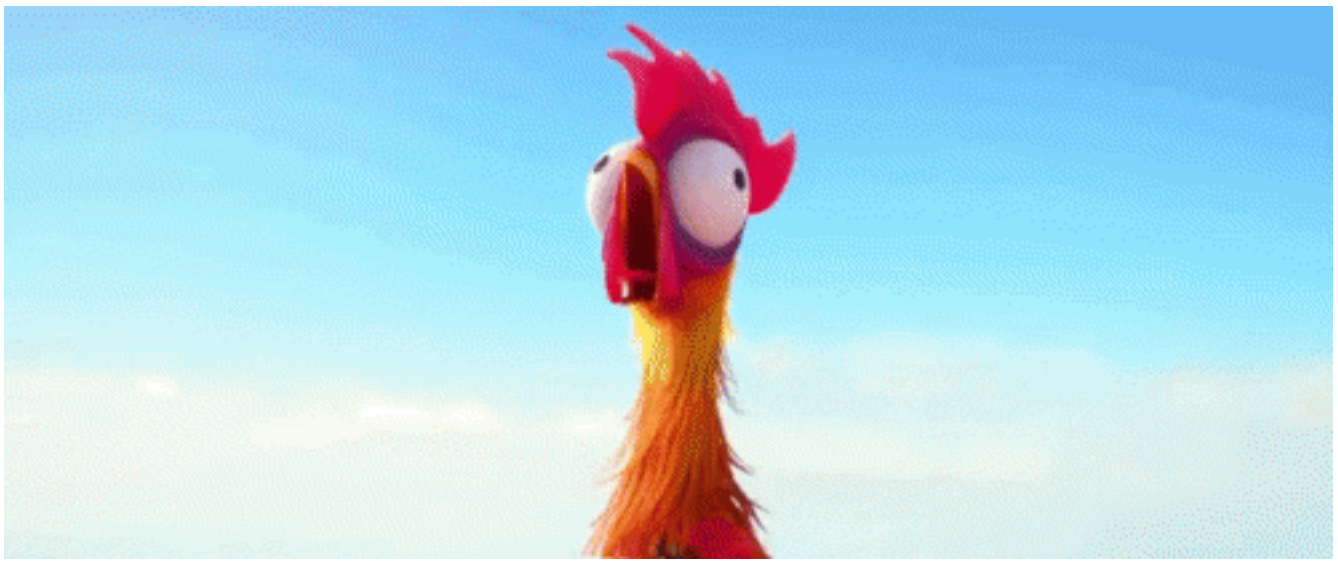
Good Morning!



How are you?

I will skip this now as we will have an evaluation before lunch where we try to collect feedback on that :)

I need help! I do not understand a certain function...

Option A - The Official Documentation

For your workshops/homework, you might want to search in the official manual: https://docs.python.org/3/ (https://docs.python.org/3/) It is the authoritative source of information. **Drawback**, you cannot use it during the test as you cannot connect to the internet during the test.

For example: "*I do not understand why the find method returns -1 for the following code:*"

```
'Waldo is not here'.find('waldo')
```

In [1]:

```
from IPython.display import IFrame

doc_url = 'https://docs.python.org/3/library/stdtypes.html?highlight=find#str
IFrame(src=doc_url, width='100%', height='60%')
```

Out[1]:

Option B - The inbuilt help system

For example: "I do not understand why the find method returns -1 for the following code:"

```
'Waldo is not here'.find('waldo')
```

The write a small program with a single line saying:

In [9]:

```
print(help('Waldo is not here'.find))
```

```
Help on built-in function find:

find(...) method of builtins.str instance
    S.find(sub[, start[, end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.

None
```

- **Advantage**, you can use this during the test as everything is stored on your computer.
- **Drawback**, you have to type a bit more and you have to remember to not type any parenthesis for the function for which you need help.

Practical Tip from Morten!

Keep a file `doodle.py` open in your Mu editor, in which you type in these help queries and in which you can experiment with small fragments of code.

I need more small exercise to recap what I learned

Nice, that sounds like a good idea.

The book https://automatetheboringstuff.com (https://automatetheboringstuff.com) contains a lot of exercises in the end of every chapter. From last year's students we know that they liked those!

In [16]:

```
from IPython.display import IFrame

doc_url = 'https://automatetheboringstuff.com/chapter4/'
IFrame(src=doc_url, width='100%', height=500)
```

Out[16]:

Keep up the good work!

Unfortunately, with the setup of the seminar, there is not really time to have a longer mental break.

But after today's session we are halfway through the contents of the seminar and done with all the Python basics.

In essence, after today you know everything to be a programmer!

What did you learn so far?

Variables as "vaults"

```
my_variable = 100
```

... later ...

```
print(my_variable)
```

Conditionals as control logic

```
if age < 18:
    print('You cannot buy cigarettes!')
else:
    print('You may smoke.')
```

Lists

```
numbers = [1, 2, 3, 5, 8]
print(numbers[0])
```

Loops

`for` loops and `while` loops

They can express the same. Remember Morten's slide!: This:

In [ ]:

```
my_list = [1, 'elephant', 4, 'rats']

index = 0
while index < len(my_list):
    element = my_list[index]
    print(element)
    index += 1
```

is the same as

In [ ]:

```
my_list = [1, 'elephant', 4, 'rats']

for element in my_list:
    print(element)
```

len(my_list) != last_index

Remember, the length of a list is always one bigger than its last index number!

That is, what will the following program do?

In [2]:

```
my_list = [1, 'elephant', 4, 'rats']

length = len(my_list)
last_element = my_list[length]
print(last_element)
```

```
---------------------------------------------------------------------
-------
IndexError                                Traceback (most recent cal
l last)
<ipython-input-2-dc219e102709> in <module>
      2
      3 length = len(my_list)
----> 4 last_element = my_list[length]
      5 print(last_element)

IndexError: list index out of range
```

It can be fixed as:

In [ ]:

```
my_list = [1, 'elephant', 4, 'rats']

length = len(my_list)
last_index = length - 1
last_element = my_list[last_index]
print(last_element)
```

Deconstructing Lists

Let's say we have the following list:

In [ ]:

```
animals = ['elephant', 4500, 'rat', 0.2, 'bat', 0.057]
```

You can deconstruct that to variables like so:

In [ ]:

```
first_element = animals[0]
second_element = animals[1]
third_element = animals[2]
fourth_element = animals[3]
fifth_element = animals[4]
sixth_element = animals[5]
```

Do you remember a shorter notation for this?

In [3]:

```
animals = ['elephant', 4500, 'rat', 0.2, 'bat', 0.057]

first_el, second_el, third_el, fourth_el, fifth_el, sixth_el = animals
print(first_el)
print(sixth_el)
```

```
elephant
0.057
```

That technique is usually called unpacking in Python and the book calls it the _The Multiple Assignment Trick _

Deconstruction requires the correct length

In [7]:

```
animals = ['elephant', 4500, 'rat', 0.2, 'bat', 0.057]

fst_animal, fst_animal_weight = animals
```

In [ ]:

```
animals = ['elephant', 4500, 'rat', 0.2, 'bat', 0.057]

fst, fst_weight, snd, snd_weight, thrd, thrd_weight = elements
```

But I do not want all elements when destructuring

You can use `_` as a placeholder for data that you are not interested in.

You can think of it as an *unnamed* variable.

In [5]:

```
animals = ['elephant', 4500, 'rat', 0.2, 'bat', 0.057]

fst_animal, _, second_animal, _, third_animal, _ = animals
print(fst_animal)
```

```
elephant
```