KAAS & MULVAD
RESEARCH & ANALYSE

CHEAT SHEET: BASIC PYTHON CODING FOR JOURNALISTS | TOMMY KAAS, KAAS & MULVAD
Find me on Twitter: @tbkaas **Find this sheet online here:** http://kortlink.dk/rvrx

GLOBAL INVESTIGATIVE JOURNALISM CONFERENCE – JOHANNESBURG, SOUTH AFRICA 2017

The screenshots below are almost all from the interactive shell at PythonAnywhere.com, which works without installing anything on your own computer. You only need a login, and the free account is great for testing. In all examples Python 2.7 has been used.

If you want to install Python on your computer, go to www.python.org. Find more links on the last page of this cheat sheet.

Numbers

Integers and floats

Integers (without decimal points) e.g 1 or 5 or 2500 or 1000000

Floats (numbers with decimal points) e.g 0.5 or 3.1427 or 55.000001

If you divide an integer with another integer, the result will be an integer:

```
>>> 10/3
3
```

Unless you use a decimal point this way:

```
>>> 10/3.
3.3333333333333335
>>>
```

Or this way:

```
>>> 10./3
3.3333333333333335
>>>
```

Or just tell Python, that your Integer is a float:

```
>>> float(10)/3
3.3333333333333335
>>>
```

You can always change a float to an integer and an integer to a float:

int()

float()

Variables

A variable is used as a container – basically a name that represents (or refers to) some value.

Some examples, where we assign the values of 3 and 3.14 to x and y:

```
>>> x = 3
>>> y = 3.14
```

Variables are easy to work with in Python. After we have assigned a value, we can use it in calculations (Python would not say “calculations”, but “expressions):

```
>>> x = 5
>>> y = 0.5
>>> x + y
5.5
>>>
```

A variables name can consist of letters, digits and underscore characters. A variable can’t begin with a digit.

Strings

Perhaps you began your career as a programmer writing:

```
>>> print "Hello World!"
Hello World!
>>>
```

In that case, you already worked with strings. The “print” part is called a statement and is simply us saying to Python: “print this string to the screen”. The “Hello World!” part is the string – like a string of characters.

We use strings a lot in Python.

You can use single quotes or double quotes – just use the same in both ends of the string.

If the string contains a single quote, you can do it like this: “Let’s go”

If the string contains a double quote, you can do it like this: ‘And then he typed ”Hello World!”’

Or use the backslash character to escape it: ‘Let\’s go’ or “And then he typed \”Hello World!\””

You can concatenate (combine) strings in several ways:

```
>>> first = "Tommy "
>>> last = "Kaas"
>>> full = first + last
>>> print full
Tommy Kaas
```

```
>>> k = 'Kaas'
>>> a = '&'
>>> m = 'Mulvad'
>>> print k,a,m
Kaas & Mulvad
```

Type() explore the type of a variable (or other object) and gives you the result. Here we explore a string:

```
>>> t = "text"
>>> type(t)
<type 'str'>
```

We often store strings in variables. If we are told, that the type is <type 'str'> we know that the content of the variable is a string.

```
>>> t = 'mystring'
>>> print t
mystring
>>> type(t)
<type 'str'>
>>> len(t)
8
>>>
```

Here the built-in len function is used. len() returns the length of a sequence – here we get the number of characters in the string. And yes, the string 'mystring' is 8 characters long.

len() and type() are both very useful functions to know when you check, why your script perhaps didn't work totally as planned.

Lists

Lists are sequences - series of values, e.g., ['John', 'Carol', 'Bob'] or [1, 2, 3]. A list appears with brackets [].

A list can be empty, and you can in your script add or remove content from a list. It is mutable – it means that it may be altered. (In Python we also have something called tuples – it's a bit like lists, but they can't be altered. We won't cover those in this sheet).

Adding content to a list is typically done with the functions append (an element at the end), extend (a sequence of elements at the end) or insert (one element at the beginning).

All elements in a sequence are numbered – from zero and upwards. You can access them individually with an index number, like in this example with a string:

```
>>> string = 'spam, spam, eggs, bacon and spam'
>>> string[0]
's'
>>> string[4]
','
>>> string[12]
'e'
```

The same goes for lists:

```
>>> list = ['spam', 'spam', 'eggs', 'bacon', 'spam']
>>> list[0]
'spam'
>>> list[2]
'eggs'
```

When we count in real life, we start with 1. But here we always start with zero.

We use indexing to access individual elements, and we use slicing to access ranges of elements – parts of a sequence:

```
>>> list = ['spam', 'spam', 'eggs', 'bacon', 'spam']
>>> list[1:3]
['spam', 'eggs']
>>> list[2:4]
['eggs', 'bacon']
>>> list[2:]
['eggs', 'bacon', 'spam']
>>> list[:2]
['spam', 'spam']
```

The *first* index is the number of the first element you want to include. The *last* index is the number of the first element *after* your slice. And that's why list[2:4] only gives us two elements from the list: index number 2 'eggs' and index number 3 'bacon'.

If the slice continues to the end of the sequence we can leave out the last index: list[2:]. As we see above the same thing works from the beginning: list[:2]

This will give you the entire sequence: list[:]

You can add an element to a list:

```
>>> mylist = ['spam', 'spam', 'spam']
>>> mylist.append('eggs')
>>> print mylist
['spam', 'spam', 'spam', 'eggs']
>>>
>>> mylist.insert(1, 'bacon')
>>> print mylist
['spam', 'bacon', 'spam', 'spam', 'eggs']
```

And you can delete elements:

```
>>> print mylist
['spam', 'bacon', 'spam', 'spam', 'eggs']
>>>
>>> del mylist[2]
>>> print mylist
['spam', 'bacon', 'spam', 'eggs']
```

Loop

A loop is a sequence of instructions that is continually repeated until a certain condition is reached. Typically, a certain process is done, such as getting an item of data and changing it, and then some condition is checked such as whether a counter has reached a prescribed number. If it hasn't, the next instruction in the sequence is an instruction to return to the first instruction in the sequence and repeat the sequence. If the condition has been reached, the next instruction "falls through" to the next sequential instruction or branches outside the loop. A loop is a fundamental programming idea that is commonly used in writing programs.

An example of a “while” loop: (each sequence is “print a number and add 1 to the number”).

```
>>> t = 1
>>> while t < 5:
...     print t
...     t = t + 1
...
1
2
3
4
>>>
```

An example of a “for” loop: (go through a list. Do something with each element)

```
>>> for name in ['bob', 'jim', 'sue', 'ann']:
...     print name
...
bob
jim
sue
ann
>>>
```

Import modules

Modules are extension that can be imported into Python to extend its capabilities. Python use the phrase "Batteries included" and it's true, that Python comes with a very large standard library – lots of modules are included. Still sometimes we must find new modules outside Python and install them. They only have to be installed once. But every time we want to use a module in Python, we must import it.

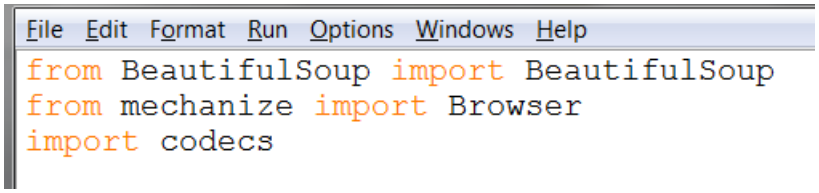
We import modules with the command import. When we do it in an interactive shell like pythonanywhere or IDLE, we don't want anything fancy to happen right after the import. The screen should look like this:

```
>>> import urllib2
>>>
```

Instead of importing the whole module you can import a single function with a given name from the module.

If you are writing a script, you will typically import the modules you may need at the beginning of the script. It could look like this.

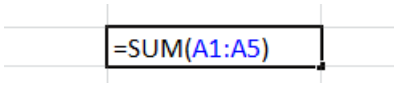
We import the module codecs – and two functions from two other modules.



Functions

A function is kind of like a mini-program inside your program. Functions contain several instructions to execute when the function is called. Python provides some built-in functions already, and you can create your own functions. The great thing about functions is that you only need to know what the function does, but not how it does it.

If it's difficult to understand the concept of a function, then think about the SUM function in a spreadsheet like Excel. We only need to know that this function:



will add the values in cell A1, A2, A3, A4 and A5 and give us the result. Exactly how the calculation is done, doesn't matter.

Where to learn more

This was the very basics of programming with Python. There are lots and lots of tutorials on the web. Some are extremely good. If you try one and don't like it or get it, then try another. But books can be great too.

I learned a lot from Magnus Lie Hetlands "Python from Beginning Python: From Novice to Professional".

This is the official Python Tutorial

From the Python Wiki: A long list of tutorials for non-programmers

In Youtube and many other sites, you'll find screencasts / video recordings of programming. Done the right way video tutorials can be excellent.

I have used the screencast service www.showmedo.com (paywall, but some stuff are free).

What do to if you get stuck?

Google for the answer. Ask a friend or user group. Read How to Ask Questions, before posting your questions online. It covers ways to help frame your questions so others can best help you

Mailing lists etc.

Python.org has lists of newsgroups and mailing lists for Python newbies:

http://mail.python.org/mailman/listinfo

One of the mailing lists "Tutor" is described like this: "This list is for folks who want to ask questions regarding how to learn computer programming with the Python language".

In short, one can send his question to the list, and soon there will be answers and those answers are usually very competent. They won't give you the final solution though. You still want to learn to code, right?

A group of working journalists has created the mailing list PythonJournos. It works the same way – if you have a problem (regarding Python) send it to the mailing list and count on good and quick answers. At the moment the group has 407 members, and it's open for everybody.

https://github.com/PythonJournos/LearningPython/wiki

https://groups.google.com/forum/?hl=en#!forum/PythonJournos

Journalism.co.uk: 4 ways journalists can get help with their code