101 - Intro - Getting to know Python and using Jupyter

February 20, 2017

1 Hm, Let's get started, shall we?

This application is called Jupyter and can be used to execute Python code (where Python is a programming, language; a way of explaining to a computer what you want it to do for you).

Select the cell with the sum below by clicking in it (a green border will appear around it) en press Shift+Enter or the Play () button in the menu at the top of this page to execute the code in the cel:

```
In [ ]: 5+11
```

The code is organised in cells anf if you want you can change that code and execute it again. Adapt the code in the cell above and execute it again.

... go ahead, I'll wait here for a moment ...

"Pffff, that's something I can do with any simple calculator"
Correct, but we're just starting out here, let's try something else:

```
In [ ]: print("Hi all!")
```

As you noticed, Jupyter will, if the given command returns a result, display this "output" below the code cell.
And if you forget something or add a type, an error message will be shown instead.

```
In [ ]: print("So this does not work"
```

Python will then try to explain what the problem is, but this may not always be 100% clear (and, to be honest, may sometimes even point you in the wrong direction). Can you figure out what went wrong in the previous statement?

-Tip: comparing to the "Hi all!" command may help. It's only a small omission, but a computer can get really confused by that kind of stuff-

2 OK, what else can Python do?

2.0.1 Remember things

Computers are very good at that (as long as you keep feeding it electricity...) We call that those things variables and in Python you'd better give them a name or else they are kinda hard to find back:

```
In [ ]: a = 'This is a text'  # text goes between quotes '...'
        a = "This is a text"  # they can also be double quotes "..." (just don't start mixing th

        # Oh yeah, and everything that comes after a # is a comment, Python will just skip it as

        b = 13
        c = 273.15              # decimal numbers use points, not a comma!
```

You see, no result, so Jupyter does not print anything, but the values are in memory and we can get them back with their names, look:

```
In [ ]: print(a, b, c)
```

2.0.2 Methods and "dot notation"

Some "things" or objects you use in Python get some kind of superpowers in the form of methods that you can call. You use these powers by adding a dot after the object and typing the method (watch out, for calling a function you always have to add brackets after the function name, even if there is nothing between them):

```
In [ ]: # e.g. get an upper case version of the text we stored in memory before
        a.upper()
```

By typing a dot and pressing the button, Jupyter will show a list of available functions. Very handy if you need some functionality, but you don't quite recall its name.
Put your cursor after the a, then type a dot and to try it out (press to select a method):

```
In [ ]: a
```

2.0.3 Remember lists

If you want to store a list of objects, you can do that by using square brackets:

```
In [ ]: minions = ['Dave','Stuart','Jerry','Jorge']
        print(minions[2])
        #watch out, element index numbers start at 0, so the 3rd minion in the list is the one t
```

But if you want to create a small database of each minion's favorite ice cream, you're probably better off by using a dictionary, since it allows you to combine a key and a value and makes it easy and fast to find those values back later:

```
In [ ]: minion_icecreams = {
            'Dave':'strawberry',            # 'Dave' is the key here, 'strawberry' is the relate
            'Stuart':'vanille',
            'Carl':'vanille',
            'Jerry':['mokka', 'vanille'],   # Indeed, we we can make this way more complicated :
            'Jorge':'chocolate'
        }
        print(minion_icecreams['Jerry'])
```

```
In [ ]: # so, can you guess what this will print out?
        print(minion_icecreams['Stuart'])

In [ ]: #and this?
        print(minion_icecreams[minions[3]])

In [ ]: #and this?
        print(minion_icecreams[minions[2]][0])
```

3 Loopen

How to print all ice cream favorites with as little effort as possible?
Note: range() is a built-in function to make a list of numbers.

```
In [ ]: print(range(4))
```

We can use this to repeat the same print statement 4 times, but each time with a number that is 1 higher.

```
In [ ]: for nummer in range(4):
            print(minions[nummer])
```

Watch out! the "white space" before the print command is important! Without it, Python would not know what belongs to the loop and what doesn't:

```
In [ ]: for number in range(4):
            print(number)
            print(minions[number])
```

Is not the same as:

```
In [ ]: for nummer in range(4):
            print(nummer)
        print(minions[nummer])
```

One last time?

```
In [ ]: print("-- GO --")
        for number in range(4):
            print(minions[number])
            # this comment is skipped 4 times
        # this comment is skipped only once
        print("--done--")
```

We don't have to use numbers! In Python we can use the list directly:

```
In [ ]: for minion in minions:
            print(minion)
```

Which makes it possible to print the favorite flavour for each of the minions in our list from the minion_icecreams dictionary

```
In [ ]: for minion in minions:
            print(minion, minion_icecreams[minion])
```

While is a similar kind of loop. It means that instructions are executed "while" a certain condition is met.
You can for instance throw an imaginary dice until a 6 is thrown:

```
In [ ]: import random                    # in order to use the random module; we'll get back to

        throw = 0                        # a throw of 0 is impossible, but we don't really care
        while throw < 6:
            throw = random.randint(1,6)  # a random number between 1 and 6
            print(throw)
```

Try this a few times and you'll get different results each time (Ctrl+Enter executes the cell while your cursor stays on it)

As you can imagine, the "while True:" construct is a bit special; since True will never turn into False (never say never...) this loop will keep going eternally, unless you explicitly stop it by for instance clicking the Stop button () in the menu above or choosing Kernel > Interrupt from the dropdown menu.

```
In [ ]: import time                     # in order to use the time module; we'll -really- get ba

        while True:
            print('.'),                 # the comma after the print statement keeps the next pri
            time.sleep(0.2)             # 0.2 seconds pause
```

Click the Stop button () to end the execution. This method is sometimes use to start a program that needs to keep running indefinitely.

Advanced: You can "catch" errors like the one you see above. This is a way of telling Python you were expecting this error might occur and handling it in an elegant way, so Python does not have to show the ugly error messages to the end user. This happens with a "try except" construct, in the following manner:

```
In [ ]: import time

        while True:
            try:                                # execute some code...
                print('.'),
                time.sleep(0.2)
            except KeyboardInterrupt:           # ... and if a KeyboardInterrupt error would occ
                print('\nThe End')                 # print 'The End' (on a new line)
                break                           # leave the while loop
```

4 Conditions

With an "if" statement, we can influence the execution flow of our code.
It allows us to make the computer do something, not do something or do a different thing, all depending on the condition we impose.

```
In [ ]: score = 85
        if score > 90:
            print('Brilliant')
        elif score > 80:
            print('Well done')
        elif score > 60:
            print('OK')
        else:
            print('Hm')
```

For instance: that Jerry seems a bit greedy and we don't want any bad examples in our list, so:

```
In [ ]: for minion in minions:
            if minion == 'Jerry':
                print('--Glutton--')
            else:
                print(minion, minion_icecreams[minion])
```

5 Real programming

We can also write our own functions in Python and then use them; this helps to keep code tidy and prevents us from having to repeat the same piece of code over and over again.

```
In [ ]: def greet(name):
            print('Hi ' + name)
```

Shall we try out our new fuction?

```
In [ ]: greet('Mariette')
```

5.0.1 And a little extra

You'll often need to print stuff to the screen and in many cases this involves mixing fixed texts with variable values (strings, but also numbers or other kinds of data). In those circumstances we can use string templates and the String.format() function, which works in way similar to the Mail Merge functionality in a word processor.

The format() function is used as follows:
'Hi {}'.format(name)
This replaces the braces with a text representation of the value of the name passed into the function

```
In [ ]: def greet(name):
            print('Hi {}'.format(name))

        greet('Willy')
```

6 Python libraries

Of course, people have already written a lot of Python code that you could use and many of them have made that code available to you in the form of libraries that are ready to be installed and used. Once such a library is installed, you can import it and use the functions it provides (much like the random and time libraries we imported before):

```
In [ ]: import math
        print("PI: {}".format(math.pi))
        print("sin(PI/2): {}".format(math.cos(math.pi)))
```

requests is a library to load web pages; in the example below we visit openweathermap and print a part of the weather prediction for the city of Mechelen.

```
In [ ]: import requests
        r = requests.get('http://api.openweathermap.org/data/2.5/weather?q=Mechelen').json()
        print(r['weather'][0]['description'])
```

But we can just as easy get a quote from the online iheartquotes.com database:

```
In [ ]: import requests
        r = requests.get('http://www.iheartquotes.com/api/v1/random')
        print(r.text)
```

Well done! you completed your first Python steps!