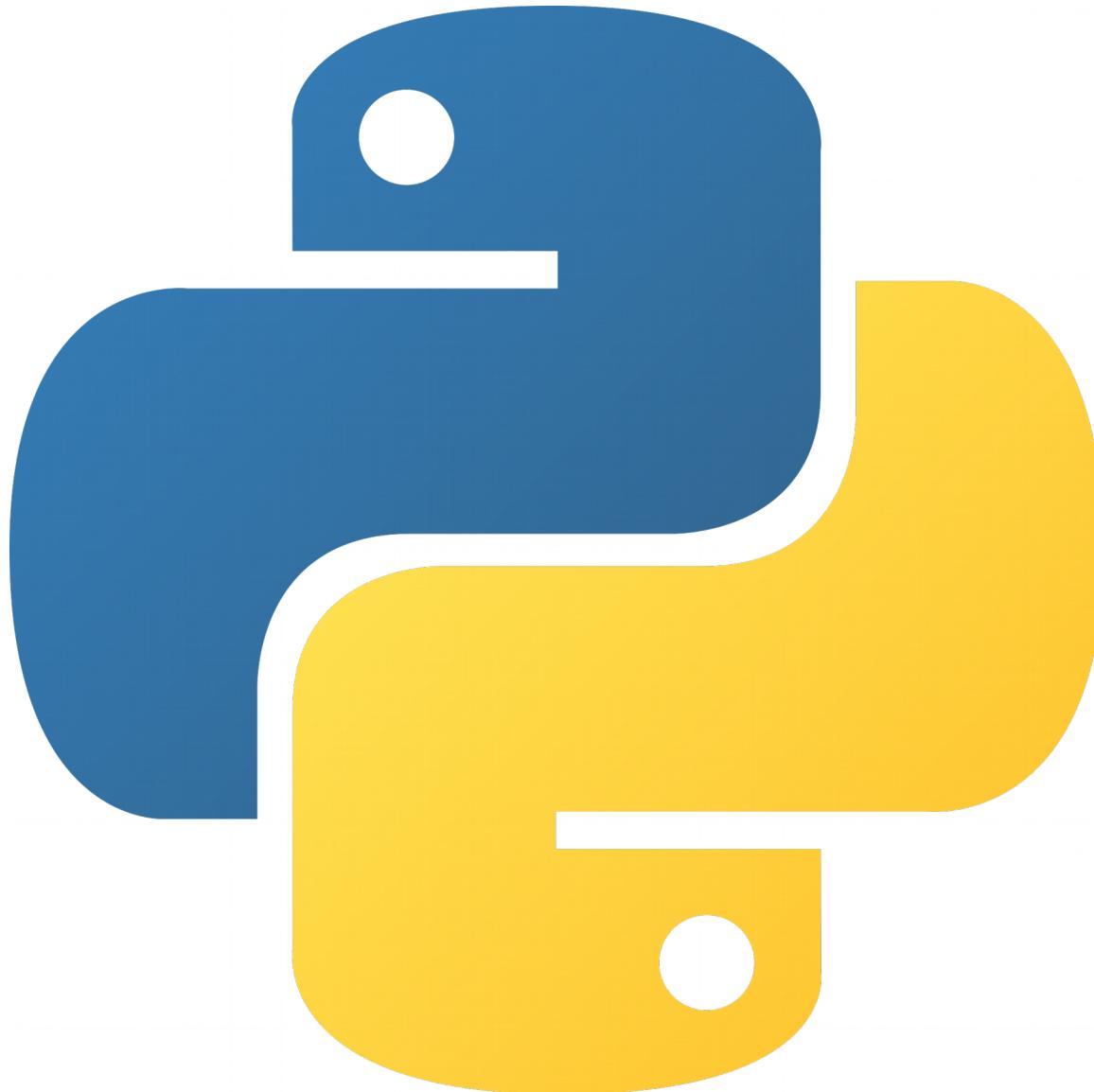


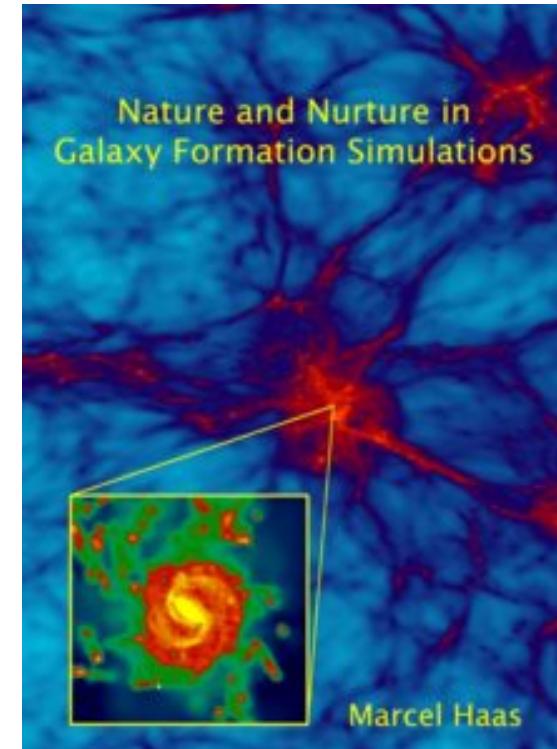
# Intro Python voor Data Science



Jansen en Van der Zee  
Marcel Haas, 2018

# Wie ben ik?

- Getraind in de sterrenkunde  
(Python sinds 2010)
- Switch naar data science in 2013



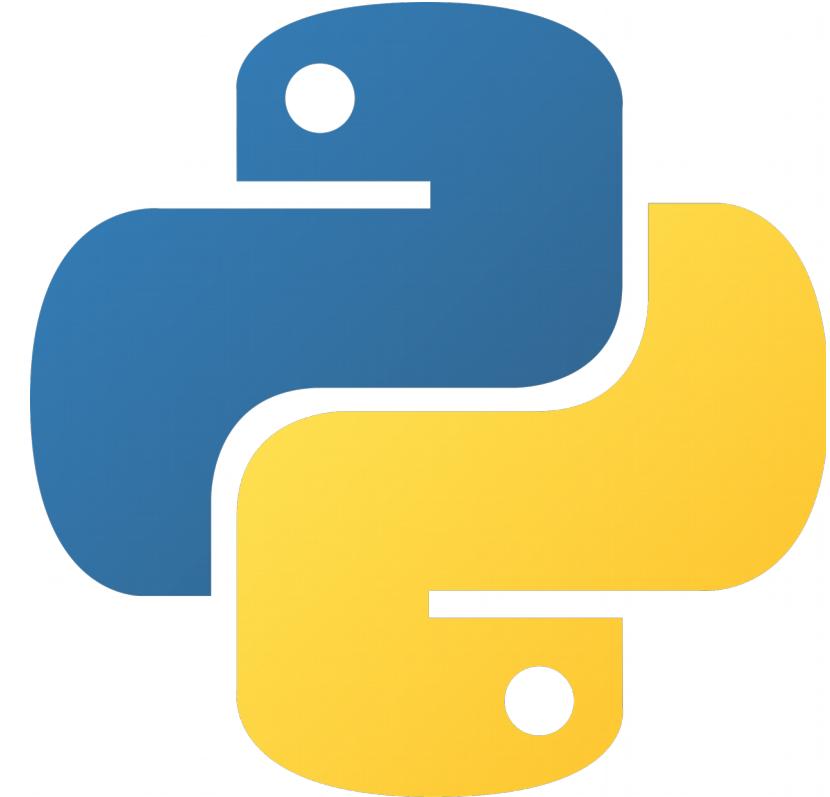
**DSW**  
zorgverzekeraar

DataScience@MarcelHaas

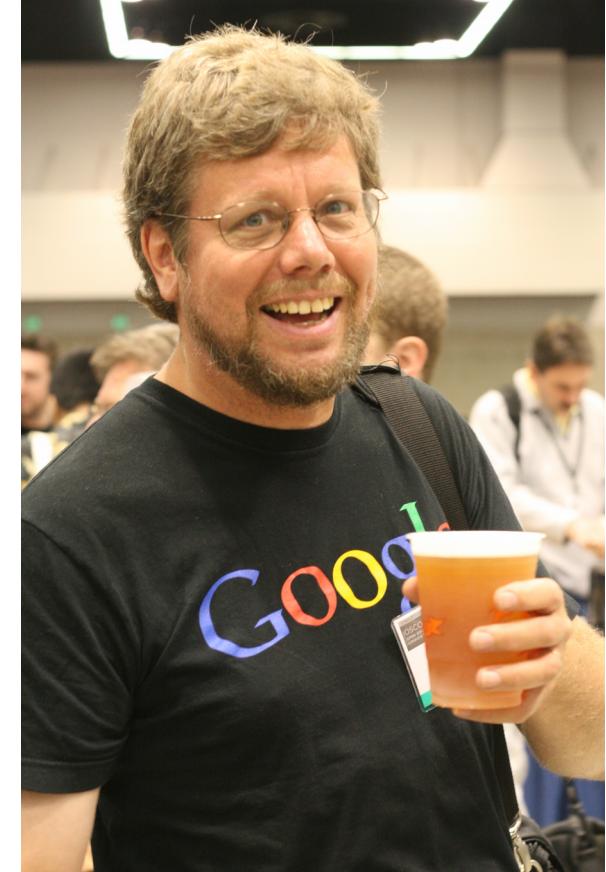
# En wie zijn jullie?

- Achtergrond (en voorkennis?)
- Verwachtingen
- Doel van de opleiding





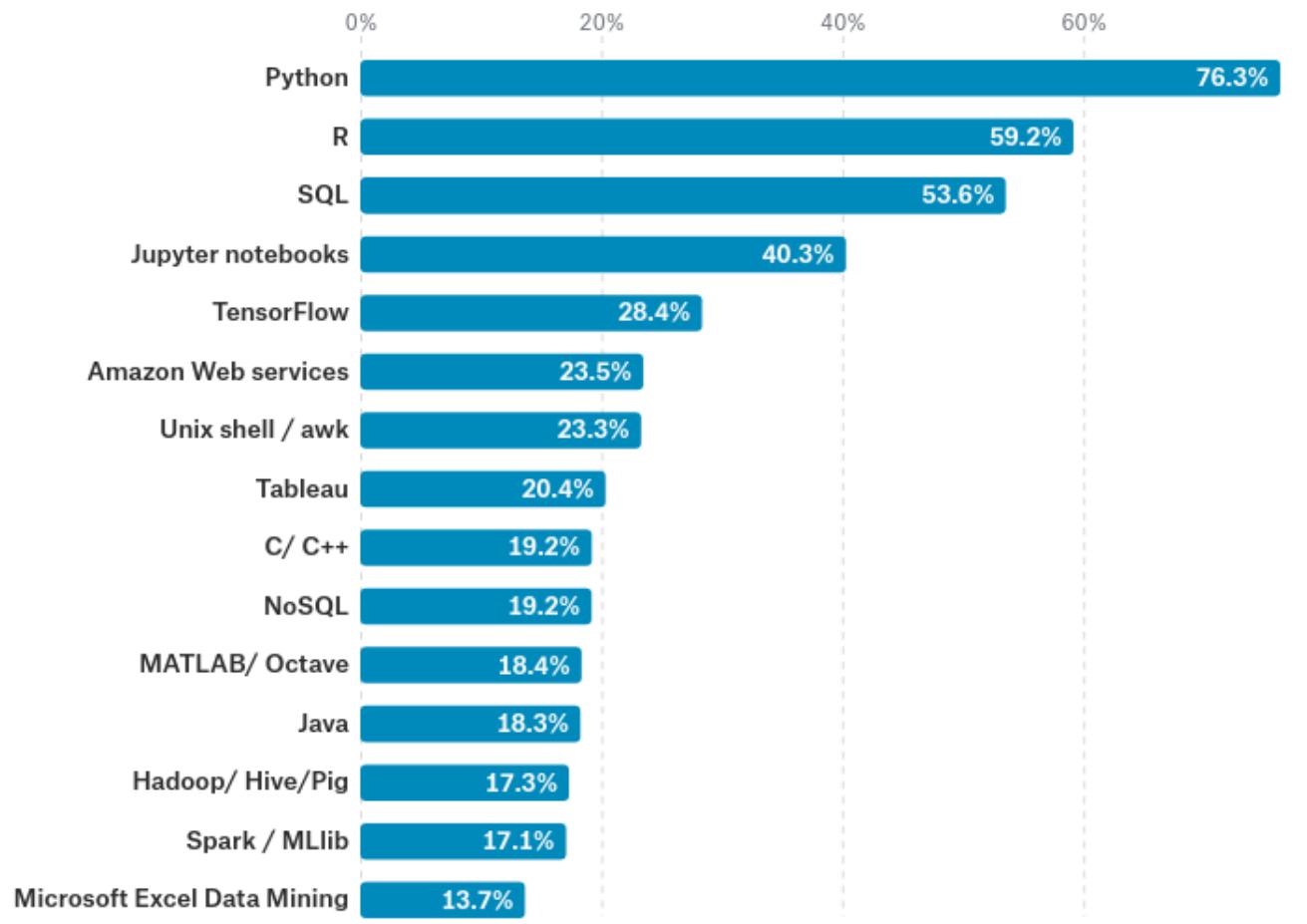
# Python!



- Huidige versie 3.7
  - Hier gebruikt: 3.6.6
  - Niet compatible met versies 2.X!
- Anaconda distributie met data science functionaliteiten: <https://www.anaconda.com/>

# Waarom Python?

- Geoptimaliseerd voor de programmeur



Kaggle:

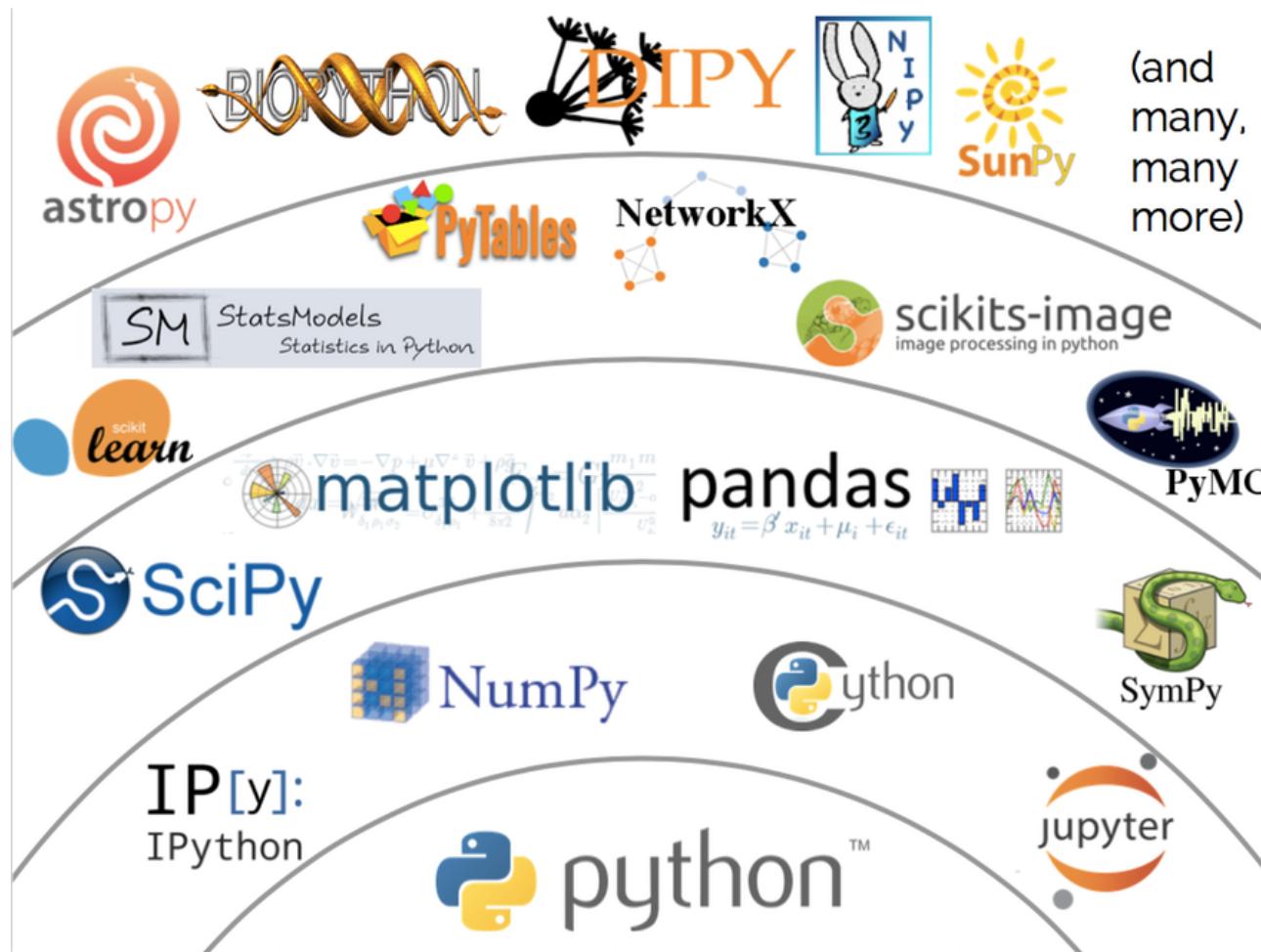
<https://www.kaggle.com/surveys/2017>

En dus:

- StackOverflow
- Open Source community

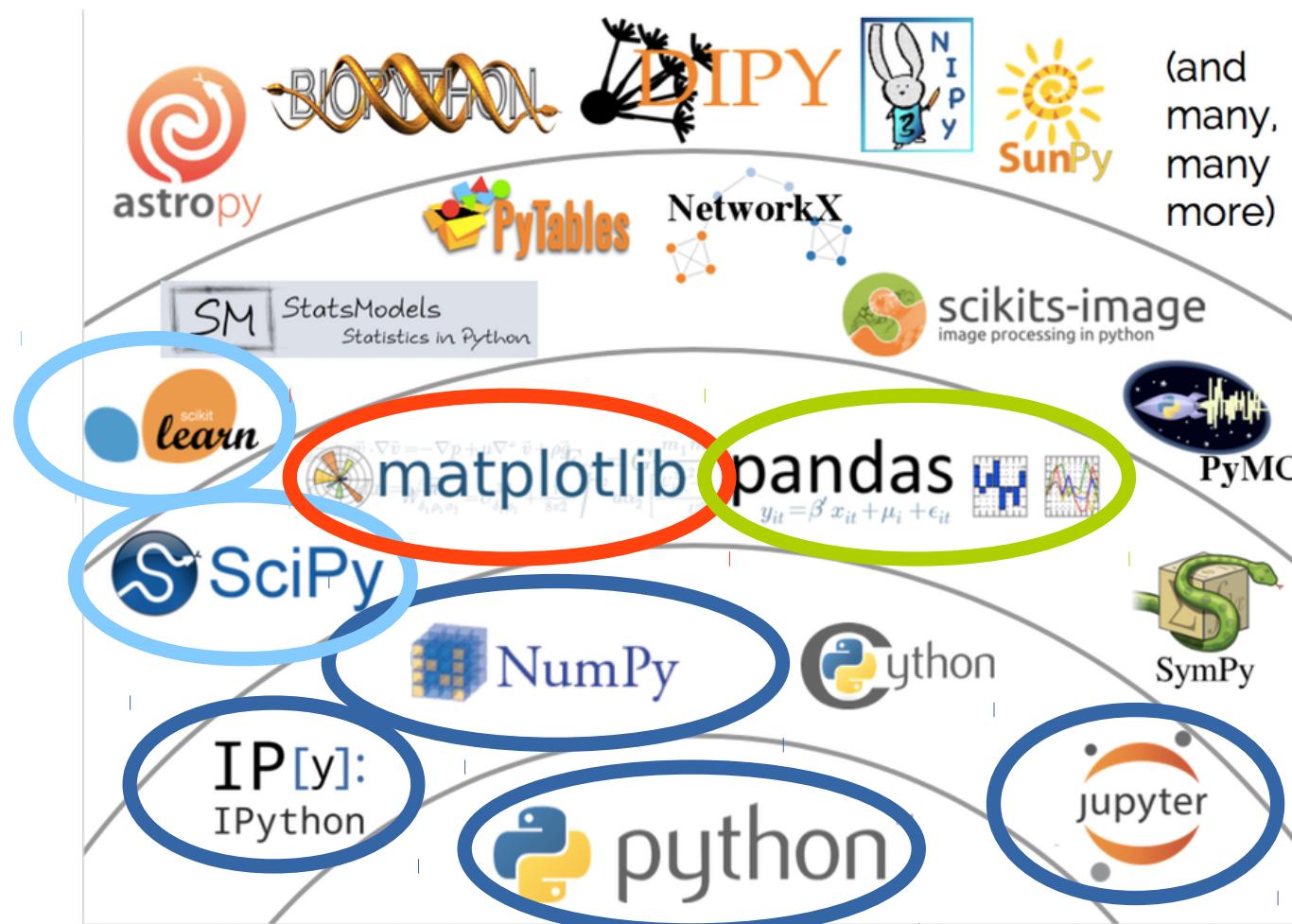
# Waarom Python?

- Python kan veel meer dan alleen data science
- Maar binnen de data science:



# Waarom Python?

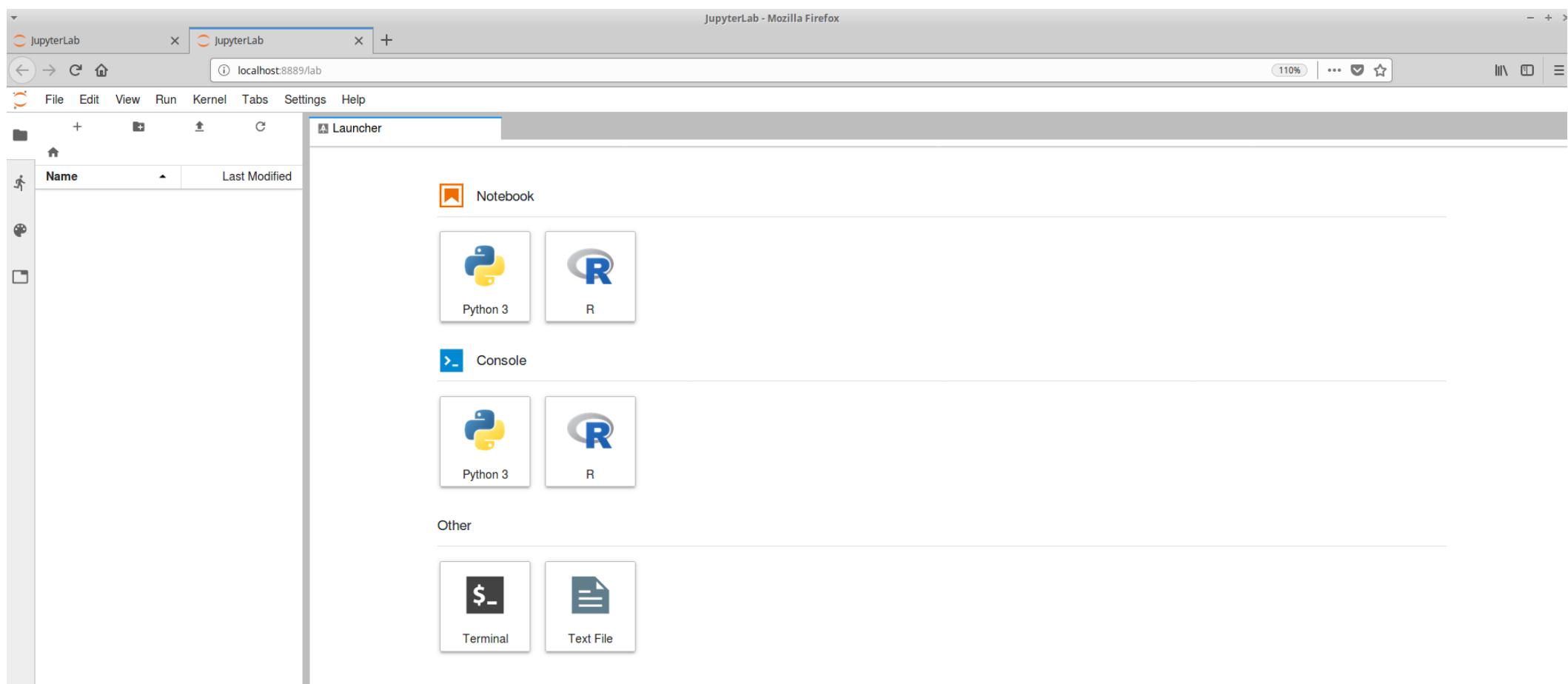
- Python kan veel meer dan alleen data science
- Maar binnen de data science:



# So.... Let's go!

- Anaconda distributie (Windows, Mac, Linux/unix):
  - Python 3.6+
  - Alle packages (dus geen miniconda, niet alleen conda)
- Jupyter Labs (of alleen Notebooks)
- Cursusmateriaal op github:
  - <https://github.com/harcel/PyDataScienceIntroNL>
  - Presentatie, “instructie”, opgaven en huiswerk

# Jupyter Labs



In een browser

# Jupyter Labs

The screenshot shows the JupyterLab interface running in Mozilla Firefox. On the left, there's a sidebar with a 'Files a la Explorer' view. The main area has a 'Launcher' tab selected, displaying three sections: 'Notebook' (with Python 3 and R icons), 'Console' (with Python 3 and R icons), and 'Other' (with Terminal and Text File icons). To the right of each section is a descriptive text block.

Files  
a la  
Explorer

JupyterLab - Mozilla Firefox  
localhost:8889/lab

File Edit View Run Kernel Tabs Settings Help

Launcher

Notebook

Python 3 R

Console

Python 3 R

Other

Terminal Text File

Notebooks met verschillende “kernels”

Consoles met dezelfde kernels

Extra’s. Default: terminal en text editor

# Jupyter Labs

Demo met cursusmateriaal:

- Beschikbare files
- Running notebooks
- Wat is...
  - Een notebook?
  - Een console?
  - Een terminal?
- Scherm lay-out

# Python! In een notebook

- Live coding

# Python! In een notebook

- Live coding

```
[1]: ?list
Init . . . . . . . . . . . . . . . . .
Docst Init signature: list(self, /, *args, **kwargs)
list( Docstring:
      list() -> new empty list
      list(iterable) -> new list initialized from iterable's
      Type: items
[ ]: list( Type: type
```

# Indices

In zowel strings, als lists, als andere ‘iterables’:

```
[ ]: mijn_iterable[start:stop:stap]
```

The diagram illustrates the components of the slice notation `mijn_iterable[start:stop:stap]`:

- naam variabele ('pointer') → points to `mijn_iterable`
- startpunt [0] → points to `start`
- eindpunt (exclusief!) [einde] → points to `stop`
- stapgrootte [1] → points to `stap`

[]: default waarden, specificatie optioneel

# Lists en dictionaries

Rechte haken geven bereik aan

```
[ ]: mijn_lijst = [item1, item2, item3, ...]
```

Willekeurige items gescheiden door komma's

Key:value-pairs gescheiden door komma's

```
[ ]: mijn_dictionary = {key1:value1, key2:value2, key3:value3, ...}
```

Gedefinieerd door curly brackets

Keys: int, string, tuple

Values: wat je maar wil.

# Objecten, instanties, pointers, attributen en methoden

Assignment

```
[ ]: naam_pointer = object
```

Handig voor jou      Instantie van object in geheugen

```
[ ]: object_pointer.attribuut
```

```
[ ]: object_pointer.methode()
```

```
[ ]: zomaar_een_functie(object_pointer)
```

# Loops

- For/while: definitie loop
- Itereren over ‘iterable’, met (locale) hulpvariable
- Wat te doen na “:”

```
[1]: for ii in [1, 7, 10]: print(ii)
```

```
1
```

```
7
```

```
10
```

# Loops en condities: INDENTATION

```
[1]: for jj in range(3):
      k = 3*jj
      print(jj, k)
print("Buiten de loop!")
```

```
0 0
1 3
2 6
Buiten de loop!
```

# if, elif, else

```
[1]: x = 2
      if x > 5:
          print("Groot getal")
      elif x < 0:
          print("Negatief getal")
      else:
          print("Weinig indrukwekkend...")
```

Weinig indrukwekkend...

‘then’ is dus vervangen door een dubbele punt en indents.

# PEP8

- 1 indent == 1 “tab” == 4 spaties
- UTF-8 karakters in Py3+
- 72 karakters of minder per regel
- Nog veel meer stijlconventies. Bekijk eens

<https://development.robinwinslow.uk/2014/01/05/summary-of-python-code-style-conventions/>

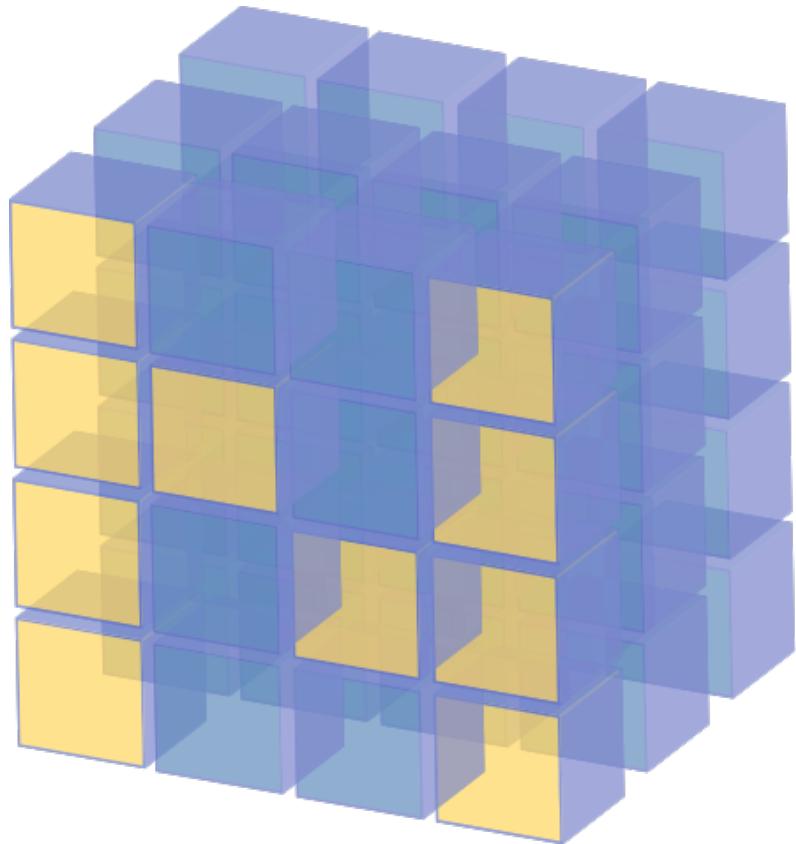
# Importeren

```
[1]: # Importeer 1 functie (sqrt()) uit een package (math):
from math import sqrt

# Importeer alles wat in de package math zit:
# from math import *

# Importeer een heel package:
import math
# Alle functionaliteit is nu beschikbaar in de math namespace:
wortel9 = math.sqrt(9)

# Importeer een package met kortere naam:
import math as m
# Alle functionaliteit is nu beschikbaar in de "m" namespace:
wortel2 = m.sqrt(2)
```



# NumPy

# Referenties

- Python Data Science Handbook  
(Jake VanderPlas, ook zijn andere materiaal)  
<https://github.com/jakevdp/PythonDataScienceHandbook>
- PyData, (Euro)SciPy, Enthought, PyCon op youtube
- Kaggle