
Introduction to Python

Prof. Dr. Thomas Kopinski

why Python

- easy to accomplish things quickly, compare ,Hello World' in Python vs. Java

Python

```
>>> print ('Hello World')  
Hello World
```

Java

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
    }  
}
```


- lots of packages for data wrangling - numpy, pandas etc.
- easy to visualize data and results of analysis (matplotlib)
- popular industry-approved frameworks and kits to do *machine learning*, i.e. tensorflow, pybrain, scikitlearn and many more
- transfer code into C with Cython (runtime reduction)
- connect to MySQL databases easily (PyMySQL)
- BeautifulSoup for web scraping
- iPython for interactive programming , jupyter notebooks is a modern and easy to use environment

Python - Anaconda

- open-source distribution for doing data science / machine learning
- develop models with scikit-learn, tensorflow and other frameworks
- easy to:
 - collect, clean, transform and work with data
 - environments are set up
 - sharing / collecting projects with students / teachers
 - product deployment simple

Installation guide for Anaconda

- <https://www.anaconda.com/distribution/>

 for this course you will need to have a working Anaconda prepared on your own environment

Introduction: Jupyter Notebook

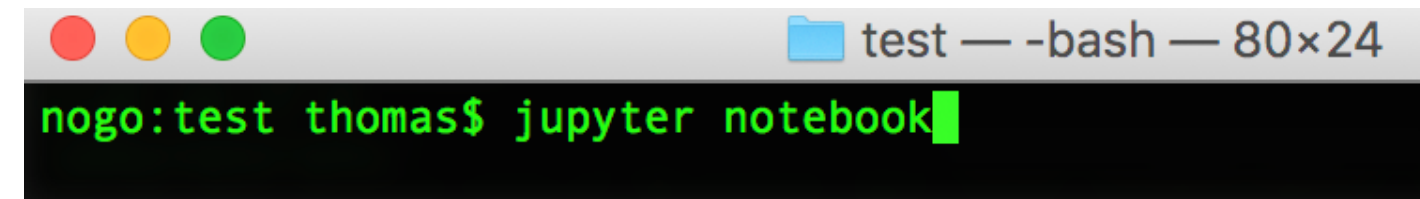
- Jupyter - loose acronym of Julia, Python, R
- working with notebook documents
- mix of code and elements like rich text, figures, tables etc...
- you can perform routines and analysis as well as see the results directly in a browser
- Jupyter Notebook App: local server-client App (remote is also possible)
- working with console and browser
- Notebook Dashboard shows local files and allows to shut down kernels
- for installation info refer to: <https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/install.html>

Notebook kernel

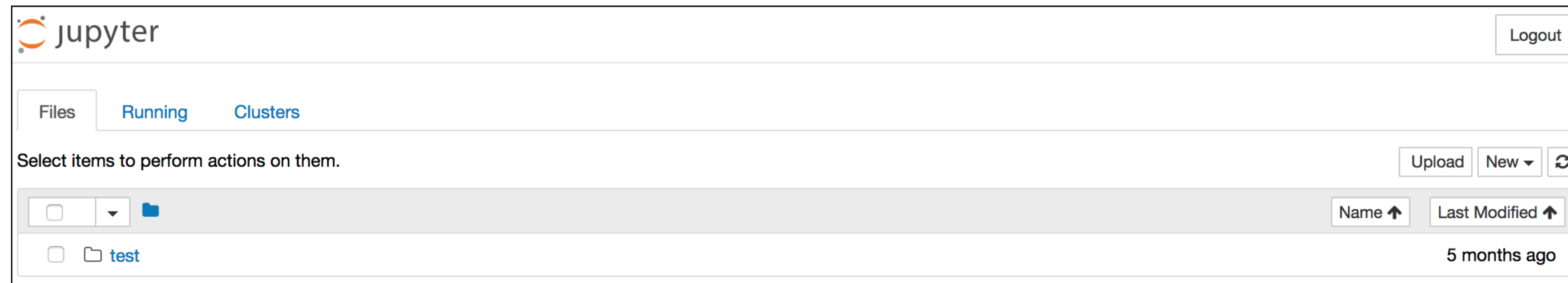
- computational engine
- executes the code in a notebook doc
- focus here: ipython kernel
- opening a notebook doc automatically launches the kernel
- executing the (content of a) notebook makes the kernel perform the computation
- makes use of CPU + RAM; RAM is released only after shutting the kernel down

Working with a notebook

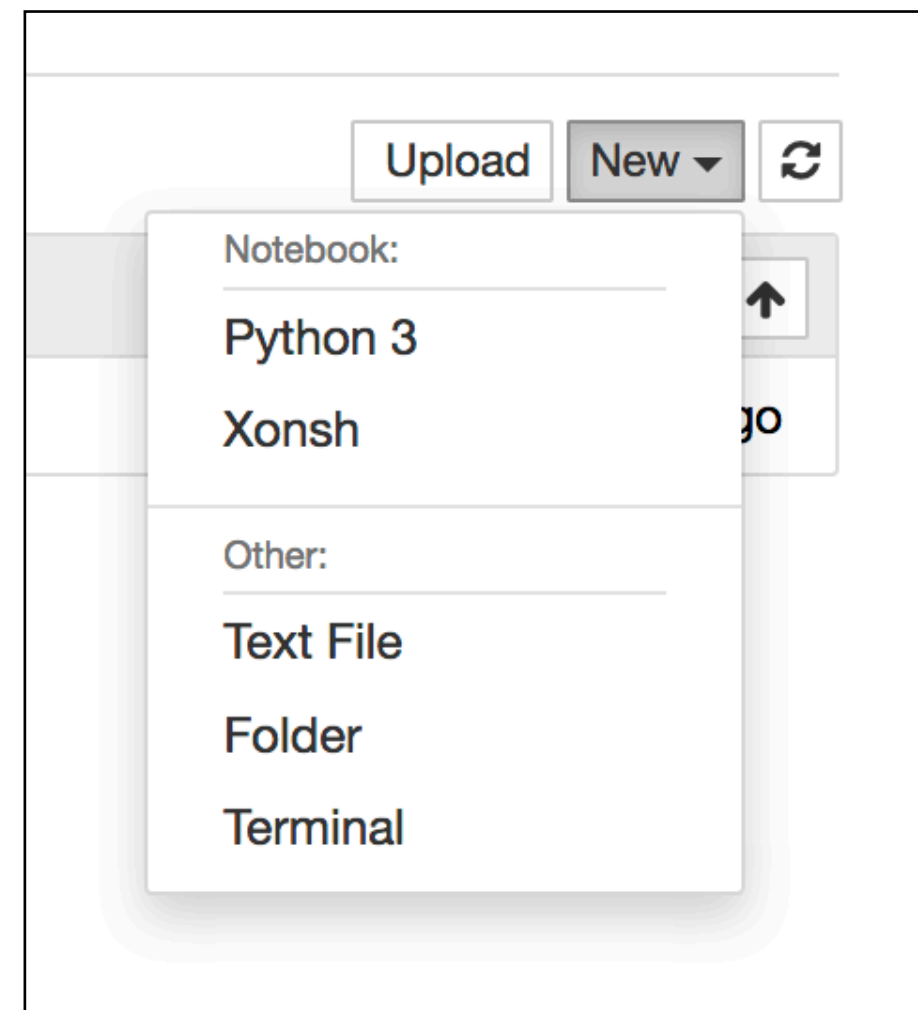
- type `jupyter notebook` into the console



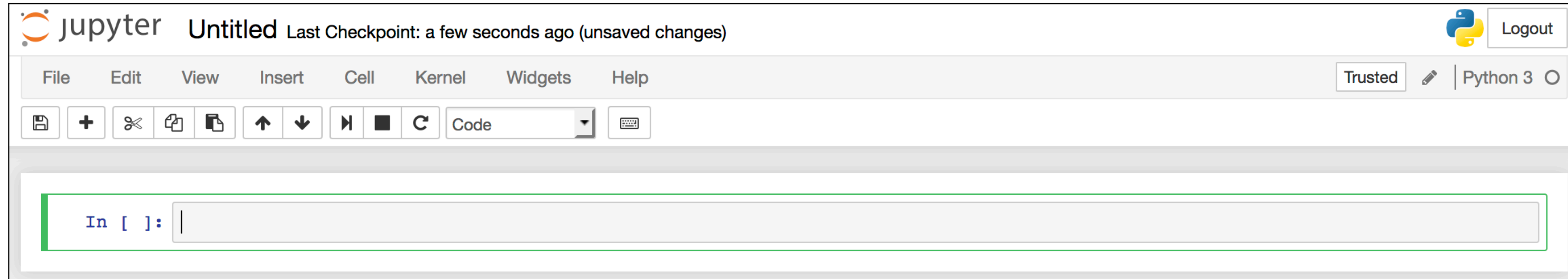
```
test — -bash — 80x24
nogo:test thomas$ jupyter notebook
```



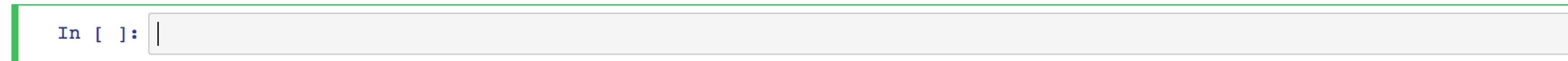
- create a new notebook:



Working with a notebook



- edit mode (green highlighting):




- command mode (blue highlighting):



- command ,c' copies a cell
- command ,v' pastes a copied cell

Working with a notebook

- typically one cell is edited
- run the cell either by pressing 
- or via the menu bar with ,cell:run‘

```
In [2]: list = [12, 34, 56, 78]  
list
```

```
Out[2]: [12, 34, 56, 78]
```

Help -> Keyboard Shortcuts

1. Basic navigation: enter, shift-enter, up/k, down/j
2. Saving the notebook: s
3. Change Cell types: y, m, 1-6, t
4. Cell creation: a, b
5. Cell editing: x, c, v, d, z
6. Kernel operations: i, 0 (press twice)

Quick reference guide

```
In [11]: %quickref
```

```
IPython -- An enhanced Interactive Python - Quick Reference Card
=====

obj?, obj??      : Get help, or more help for object (also works as
                  ?obj, ??obj).
?foo.*abc*       : List names in 'foo' containing 'abc' in them.
%magic           : Information about IPython's 'magic' % functions.

Magic functions are prefixed by % or %, and typically take their arguments
without parentheses, quotes or even commas for convenience.  Line magics take a
single % and cell magics are prefixed with two %.

Example magic function calls:

%alias d ls -F    : 'd' is now an alias for 'ls -F'
alias d ls -F     : Works if 'alias' not a python name
alist = %alias    : Get list of aliases to 'alist'
cd /usr/share     : Obvious. cd -<tab> to choose from visited dirs.
%cd??            : See help AND source for magic %cd
%timeit x=10      : time the 'x=10' statement with high precision.
%%timeit x=2**100
x**100           : time 'x**100' with a setup of 'x=2**100'; setup code is not
```

magic functions

```
In [1]: %lsmagic
```

```
Out[1]: Available line magics:
%alias %alias_magic %autocall %automagic %autosave %bookmark %cat %cd %clear %colors %config %connect_info
%cp %debug %dhist %dirs %doctest_mode %ed %edit %env %gui %hist %history %killbgscripts %ldir %less %lf
%lk %ll %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %lx %macro %ma
gic %man %matplotlib %mkdir %more %mv %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2
%popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref
%recall %rehashx %reload_ext %rep %rerun %reset %reset_selective %rm %rmdir %run %save %sc %set_env %sto
re %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode

Available cell magics:
%%! %%HTML %%SVG %%bash %%capture %%debug %%file %%html %%javascript %%js %%latex %%markdown %%perl %%pr
un %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writef
ile

Automagic is ON, % prefix IS NOT needed for line magics.
```

```
In [12]: ?%timeit
```

```
Docstring:
Time execution of a Python statement or expression

Usage, in line mode:
    %timeit [-n<N> -r<R> [-t|-c] -q -p<P> -o] statement
or in cell mode:
    %%timeit [-n<N> -r<R> [-t|-c] -q -p<P> -o] setup_code
    code
    code...

Time execution of a Python statement or expression using the timeit
module. This function can be used both as a line and cell magic:

- In line mode you can time a single-line statement (though multiple
  ones can be chained with using semicolons).

- In cell mode, the statement in the first line is used as setup code
  (executed but not timed) and the body of the cell is timed. The cell
  body has access to any variables created in the setup code.

Options:
-n<N>: execute the given statement <N> times in a loop. If this value
is not given, a fitting value is chosen.
```

course information

- during the course of the term, more jupyter notebooks will be added
- additionally, there will be tasks to solve with the help of interactive jupyter notebooks and slides
- this course aims at preparing you to be able to deal with data
- this means:
 - know the most important features of Python
 - be able to setup a programming environment for data analysis
 - start the process of data preprocessing, data cleaning and data wrangling
 - be able to do data exploration
 - apply data visualization to common problems
 - be prepared for upcoming courses during the data science master studies
- at the end of the term:
 - final exam ,Introduction to Data Science‘
 - 90 minutes, 90 points | 45 points needed to pass