

Python

A quickstart into the very basics
Get to know your user environment

1

Thank You

- <https://github.com/gjbex/training-material/tree/master/Python>
- *Whirlwind Tour of Python* by Jake VanderPlas (O'Reilly).
Copyright 2016 O'Reilly Media, Inc., 978-1-491-96465-1.
<https://www.oreilly.com/programming/free/files/a-whirlwind-tour-of-python.pdf>
- University of Virginia, Advanced Research Computing Services, Python Quickstart
<https://arcs.virginia.edu/python-quickstart>
- <http://www.cs.cornell.edu/courses/cs1110/2018sp/>
- https://fabienmaussion.info/scientific_programming/html/00-Introduction.html
- <https://justinbois.github.io/bootcamp/>



2

See also

- <https://www.southampton.ac.uk/~fangohr/teaching/python/book.html>
- <https://www.math.ubc.ca/~pwalls/math-python/>
- <http://troll.cs.ua.edu/ACP-PY/index.html>
- <https://data-flair.training/blogs/python-lambda-expressions/>
- <http://pages.physics.cornell.edu/~myers/teaching/ComputationalMethods/GettingStarted.html>
- <https://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/index.html>
- <https://www2.cs.duke.edu/courses/spring18/compsci101/index.php>
- <https://github.com/parrt/msan501>
- <https://docs.python-guide.org/intro/learning/>



3

Tutorials

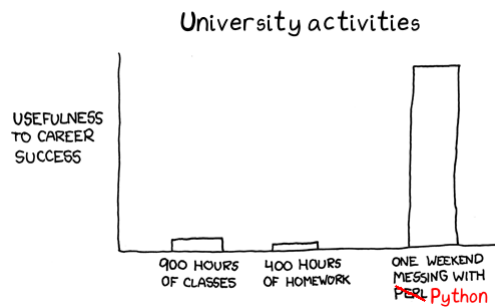
- <https://www.python.org/about/gettingstarted/>
- <https://realpython.com/>
- <https://www.learnpython.org/>
- Cheat sheets
- <https://www.datacamp.com/community/data-science-cheatsheets>



4

Why Programming

5



https://fabienmaussion.info/scientific_programming/img/00_messing_python.png



6

Why programming?

- Programming is an integral part of research
- Programming occurs on different levels:
 - Write small scripts,
 - Write complete projects,
 - Or need a good understanding of what a software packages does
- All programming languages offer to a certain extend the same building blocks
 - Understand the basic building blocks
 - Decompose your problem to fit those blocks

7

Programming?

- It may be (almost) impossible to solve a problem by executing commands at the command prompt.
- What is needed? A **sequence of precise instructions** that, once performed, will complete a **specific task**.
- Computer programs can't do that many things, they can:
 - Assign values to variables (memory locations).
 - Make decisions based on comparisons.
 - Repeat a sequence of instructions over and over.
 - Call subprograms.

8

Programming language

- There are many programming languages, with changing popularity
- Check the Tiobe Index: <https://www.tiobe.com/tiobe-index/>
- Consider:
 - it is suited to the problem at hand?
 - is there an active community?
 - is it any good for the job market?

9

Key concepts in programming

- Check Isaac Computer Science:
https://isaaccomputerscience.org/topics/programming_concepts?examBoard=all&stage=all
- Instructions / Basic Syntax
- Data Types
 - Classification of the type of data being stored or manipulated within a program.
 - Data types are important because they determine the operations that can be performed on the data.
- Variables
 - Named container, held by the computer in a memory location.
 - Has a unique identifier (name) that refers to a value.
- Input / Output

10

10

Key concepts in programming

- Operators
 - Arithmetic
 - Comparison
 - Logical
- Sequence:
statements are written one after another, will be executed one statement at a time in the order that the statements are written in.
- Selection:
execute lines of code only if a certain condition is met.
- Iteration (loop):
repeat a group of statements .
- Subprogram (function):
is a named sequence of statements, can be repeatedly “called” from different places in the program

11

11

And there will be errors...

- Syntax error
 - A mistake against the language rules
 - Program will not run and will return an error message
- Runtime error
 - Usually due to some missing variables, modules,...
- Semantic error
 - A mistake in the reasoning
 - Program is not executing as intended / expected

12

Python: setting the scene

get comfortable within the Python universe



13

What is the playfield?

- *The best way to learn a language is to "get your hands dirty"*



- Get data (simulation, experiment, etc.)



- Manipulate and process data



- Visualize results
 - quickly to understand,
 - high quality figures, for reports or publications



15

What is Python?

- From www.python.org: "Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance."
- Python is a general purpose programming language used for a huge variety of purposes. It's user community is growing rapidly!
(<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>)



16

What is Python?

- a **general purpose interpreted** programming language.
- a language that supports multiple approaches to software design, principally **structured** and **object-oriented** programming.
- provides **automatic memory management** and **garbage collection**
- **dynamically** typed.

Brian Gregor (BU): A Brief Introduction to Using Python for Computational Neuroscience



17

Why Python?

- Python is quick to program in (*explorative programming*)
- Python is popular in research, and has lots of libraries for science
Widely used – extensive capabilities, documentation, and support
- Python interfaces well with faster languages
- Python is free
- Cross-platform (Windows, Mac, Linux)
- Access to advanced math, statistics, and database functions
- *Why write programs for research?*
 - Scripted research can be tested and reproduced
 - Programs are a rigorous way of describing data analysis for other researchers, as well as for computers. By sharing codes, which are much more easy for a non-author to understand than spreadsheets

<http://github-pages.ucl.ac.uk/rsd-engineeringcourse/ch00python/00pythons.html>



18

Popular Python?

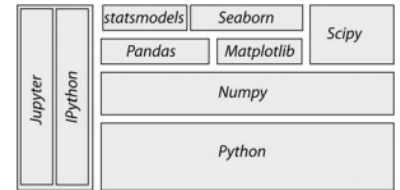
- Popular programming languages?
- <https://www.tiobe.com/tiobe-index/>
- What is Python used for?
- <https://www.jetbrains.com/research/devecosystem-2018/python/>



20

Python ecosystem

- Large and active ecosystem
- Core Python
 - Standard libraries
 - third-party packages:
 - *NumPy* for manipulation of homogeneous array-based data,
 - *Pandas* for manipulation of heterogeneous and labeled data,
 - *SciPy* for common scientific computing tasks,
 - *Matplotlib* for publication-quality visualizations,
 - *IPython* for interactive execution and sharing of code, etc. *Python versions*



21

Python versions

- Current 3.x
 - More clean than 2.x
 - Python 3.x introduced some backwards-incompatible changes to the language, so code written for 2.7 may not work under 3.x and vice versa.
 - Almost all Python libraries supported
- Version 2.7.x
 - Last of the 2.x releases
 - Many Python 3.x features have been retrofitted
 - All libraries support it

Note: in-application scripting may be stuck at Python 2.7!

Python 2 countdown:
<https://pythonclock.org/>

- Taken from GJ Bex

22

22

User Environment

Running Python Code



25

- <https://docs.anaconda.com/anaconda/user-guide/getting-started/>
- <https://realpython.com/run-python-scripts/>
- <https://plot.ly/python/ipython-vs-python/>
- <https://yihui.name/en/2018/09/notebook-war/>
- <https://www.theatlantic.com/science/archive/2018/04/the-scientific-paper-is-obsolete/556676/>
- <https://fangohr.github.io/blog/installation-of-python-spyder-numpy-sympy-scipy-pytest-matplotlib-via-anaconda.html>



26

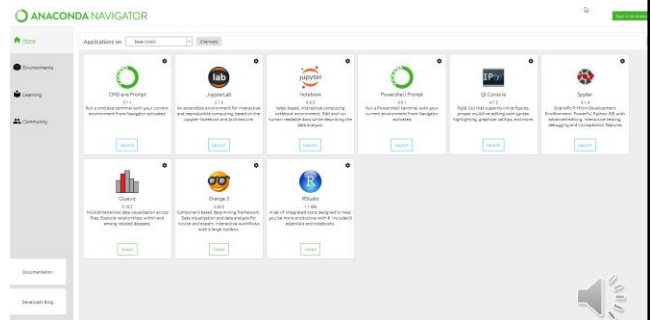
Making it work

- Write the code
 - choose a good editor (or integrated development environment - IDE)
 - featuring color coding, syntax checks, ...
 - code is just a text file
- Convert to machine code
 - make sure that you have the right interpreter (or compiler) available
- Run the code
 - run on the command line
 - run in a script mode (Python)
 - run in IDE or in Jupyter notebooks

27

Where to start?

- 2 elements needed for programming in Python:
 - writing and editing Python code
 - running that code in an interpreter
- Choose a platform - primary ways to run Python code:
 1. Terminal
 1. Python interpreter
 2. IPython interpreter
 3. Running scripts
 2. IDE
 - Spyder
 3. Jupyter notebook.



28

How do I get Python?

- core Python package
 - <https://www.python.org/downloads/>
 - easy to install but probably *not* the way to go.
- Using a distribution simplifies the process of setting up your python environment, includes core Python, necessary data packages, and integrates useful tools (IDE's, notebooks, etc)

Python Distributions:

- **Anaconda distribution**
 - <https://www.anaconda.com/>
 - Download: <https://www.anaconda.com/distribution/>
- WinPython (<https://winpython.github.io/>)
 - Windows specific data science distribution



29

Anaconda installation

- Installation documentation:
<https://docs.anaconda.com/anaconda/install/>
- Downloading and installing Anaconda is simple.
- Download the installer – follow the wizard

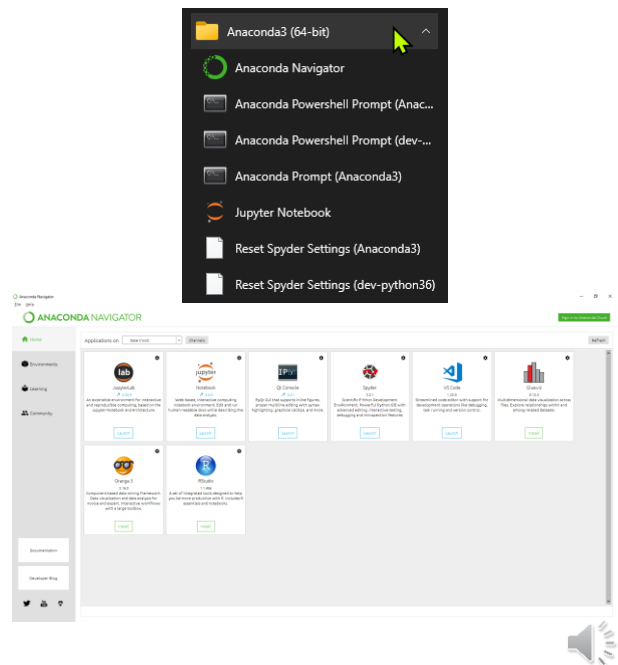


- Anaconda installation – managed PC - Windows
- Install as Admin under C:\Workdir\MyApps\Anaconda3

30

Anaconda Navigator

- <https://docs.anaconda.com/navigator/index.html>
- The Navigator is a main landing page for working with your python environment.
- Launch editors (spyder, jupyter notebook, etc.) to develop python code
- Manage (install packages, etc.) the python environment

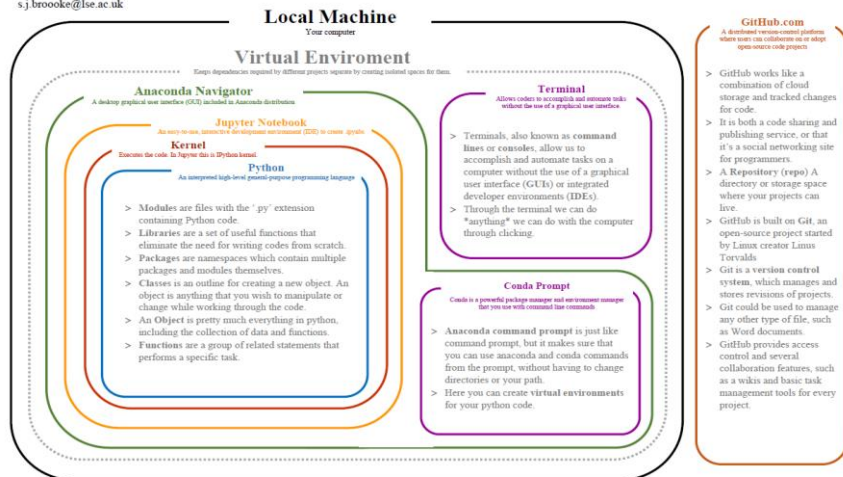


31

How everything fits together

Coding in Python: Understanding How Everything Fits Together

Dr Siân Brooke
s.j.brooke@lse.ac.uk



<https://www.sianbrooke.co.uk/dr-brookes-blog/coding-in-python-understanding-how-everything-fits-together>

32

Check this out

- <https://www.sianbrooke.co.uk/dr-brookes-blog/coding-in-python-managing-packages-with-conda-and-pip>
- <https://www.earthdatascience.org/courses/intro-to-earth-data-science/python-code-fundamentals/use-python-packages/introduction-to-python-conda-environments/>
- Quick intro into
 - Packages, modules
 - Why working with environments: keep dependencies
 - Installing with conda, pip

33

Why Environments?



- A major barrier to reproducibility in the world of computational science is package dependencies.
- When sharing code, we want to be able to setup a computer ecosystem that is exactly the same as when we were developing the code. This is where environments come in.

34

Environment: General Workflow



- **Create the environment**, either from scratch or based off an existing environment.
 - Create environment : `conda create --name myenv python=3.9`
 - where myenv is the name of the environment you would like to create
- **Activate the environment**, either using the command line or using the Anaconda Navigator graphical user interface. Anaconda starts by default in the base environment.
 - Via Command Line
 - Open the Anaconda Prompt.
 - Activate the environment using: `conda activate myenv`
 - where myenv is the name of your environment.
 - Via Anaconda Navigator
 - Open the Anaconda Navigator.
 - Switch to the desired environment by clicking the drop-down menu next to the “Applications on” text.

35

Environment: General Workflow



- Do whatever you would like to do with Python.
 - Develop code and install the required packages for the code as you go, using either :
 - `conda install --name myenv <package>` or
 - `pip install <package>`
 - where <package> is the name of the python package you wish to install.
- Either **close** the programs you were working on and the Anaconda programs or **deactivate** the environment with `conda deactivate`.

36

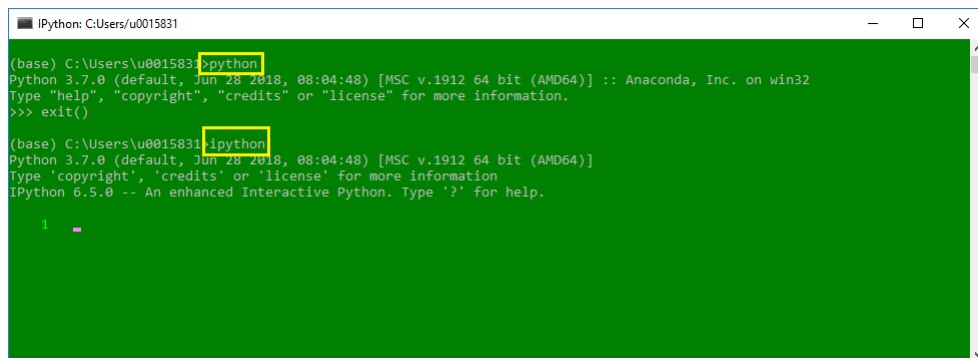
Python interpreter

<https://realpython.com/run-python-scripts/>

39

Python interpreter

- The interpreter is able to run Python code in two different ways:
 - As a piece of code typed into an interactive session
 - As a script or module



```
IPython: C:\Users\u0015831
(base) C:\Users\u0015831>python
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

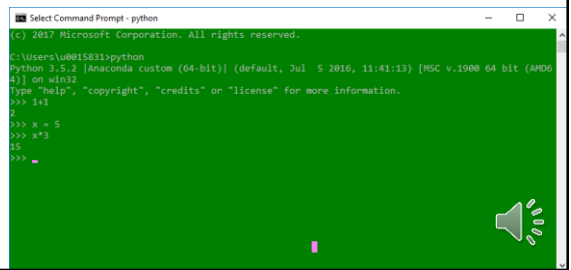
(base) C:\Users\u0015831>ipython
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information
IPython 6.5.0 -- An enhanced Interactive Python. Type '?' for help.

1  _
```

41

Python interpreter

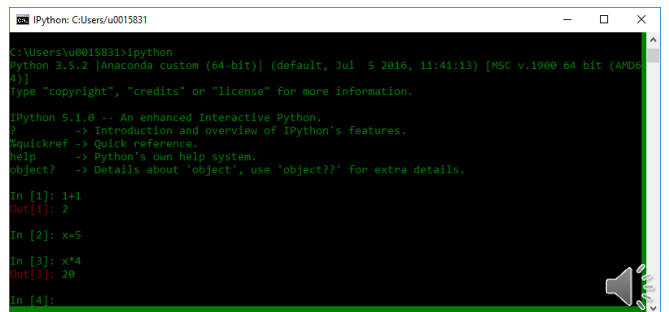
- The most basic way to execute Python code is line by line within the Python interpreter (interactive session).
- The Python interpreter can be started by typing: `python`
 - Terminal on Mac OS X and Unix/Linux systems,
 - (anaconda)Command Prompt application in Windows
 - `>>>` by default
- `help()` starts the helper environment



42

IPython interpreter

- Enhanced Interactive shell
- Enhancements to the basic Python interpreter: `ipython`
- <https://stackoverflow.com/questions/12370457/what-is-the-difference-between-python-and-ipython>



43

IPython interpreter

- IPython is an enhanced version of python that makes interactive python more productive.
 - Tab autocompletion (on class names, functions, methods, variables)
 - More explicit and color-highlighted error messages
 - Better history management
 - Basic UNIX shell integration (run simple shell commands such as cp, ls, rm, cp, etc. directly from the IPython command line)
 - Nice integration with many common GUI modules (PyQt, PyGTK, and tkinter)
 - <https://www.quora.com/What-is-the-difference-between-IPython-and-Python-Why-would-I-use-IPython-instead-of-just-writing-and-running-scripts>



44

Magic commands

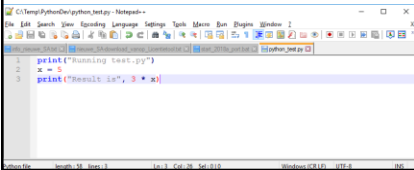


- <https://ipython.readthedocs.io/en/stable/interactive/magics.html>
- An enhancement in IPython known as magic commands
- Know more: `magic` command.
Information of a specific magic function is obtained by `%magicfunction?`
- Designed to solve various common problems in standard data analysis. There are two types of magic commands –
- 2 types
 - Line magics
 - They are similar to command line calls. They start with % character. Rest of the line is its argument passed without parentheses or quotes.
 - Cell magics
 - They have %% character prefix. Unlike line magic functions, they can operate on multiple lines below their call.

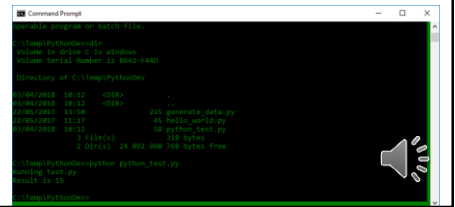
45

Python scripts

- Programs: save code to file, and execute it all at once.
 - Script: A plain text file containing Python code that is intended to be directly executed by the user
 - By convention, Python scripts are saved in files with a `.py` extension.



```
C:\Temp\PythonDev\python_test.py - Notepad++
[File Edit Search View Spelling Language Settings Tools Macro Run Plugins Window Help]
C:\Temp\PythonDev\python_test.py
1 print("Running test.py")
2 x = 5
3 print("Result is", 3 * x)
```



```
Microsoft Windows [Version 6.0.6002.18005]
(c) 2009 Microsoft Corporation. All rights reserved.

C:\Temp\PythonDev>python python_test.py

Running test.py
Result is 15

C:\Temp\PythonDev>
```

46

Run Python script

Linux

- Write script in editor
- Run script using Python interpreter
`python hello_world.py`
- Make script executable
- `chmod u+x hello_world.py`
- Run script directly
`./hello_world.py`

Windows

- Write script in editor
- Run script using Python interpreter
`python hello_world.py`
- Run script directly
`hello_world.py`

47

Python scripts



- Linux

```
#!/usr/bin/env python
```

- determines the script's ability to be executed like a standalone executable without typing `python` in the terminal
- double clicking it in a file manager (when configured properly).



48

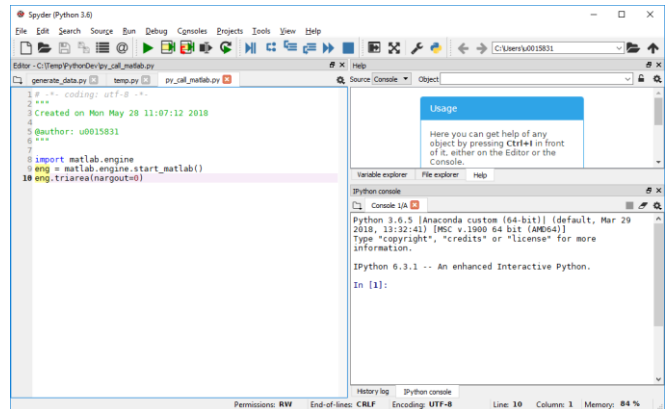
Spyder



50

IDE: Spyder

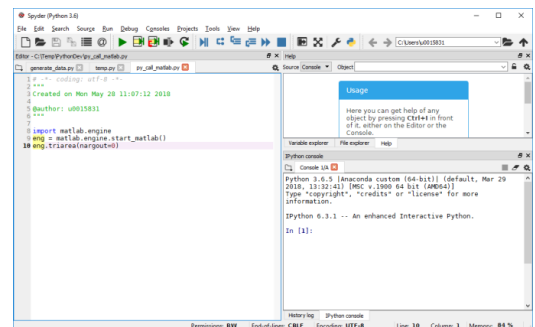
- Integrate different aspects of programming and running code.
- SPYDER: "Scientific Python Development Environment"
<https://www.spyder-ide.org/>
- Several tools in one integrated environment (cfr MATLAB desktop)
 - a code editor
 - IPython interpreter / console
 - variable inspector
 - control icons
- Documentation:
<https://docs.spyder-ide.org/current/index.html>



51

IDE: Spyder

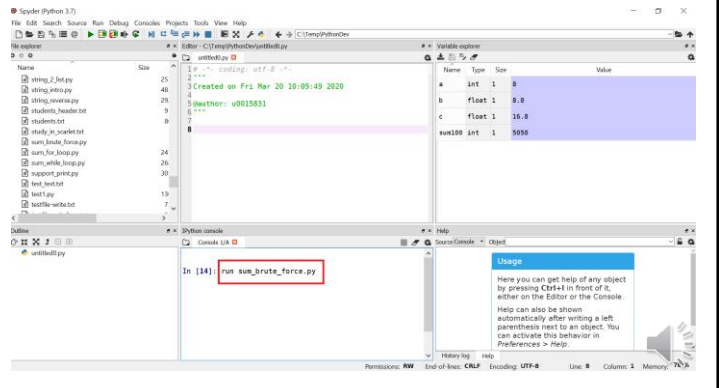
- Spyder for code development.
 - Start from Anaconda Navigator
 - Command window: spyder
- Magic commands apply
 - Clear Console:
 - %cls
 - Clear all variables from Variable Explorer (reset the namespace):
 - %reset
 - With `automagic on`, % prefix not needed



52

Running scripts in Spyder console

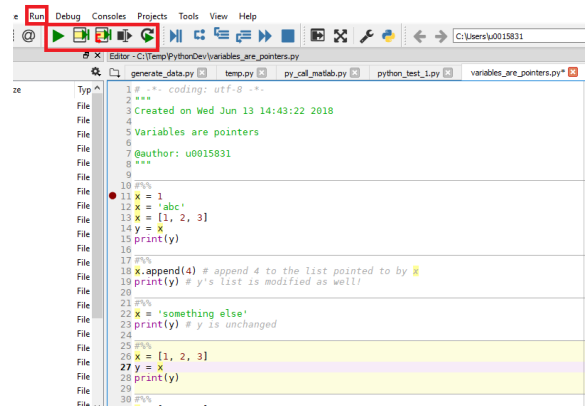
- Run a .py file from the console
 - `run script.py`
- Tab autocompletion works!



53

Running scripts in Spyder

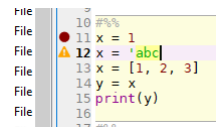
- Run scripts either with the green arrow icons or through the Run menu. *Run/green arrow* runs the entire script.
- *Run selection or current line* will run a highlighted portion of the script.
- Create **cells** by enclosing chunks of code with lines consisting of `# %%`
Run cell/green arrow with a box runs the cell.
- File: `first_prog_1.py`



54

Running scripts in Spyder

- A yellow triangle beside a line indicates a syntax error or potential problem.
- Tab completion for names familiar to it. It can show a list of members of a package for your selection, and when you have chosen a function it can show you a list of its arguments.



55

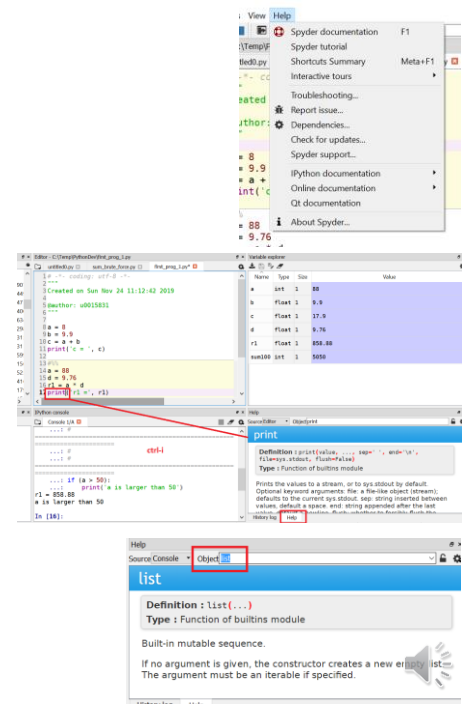
Spyder Help

- Help on Spyder from Help menu
- Help related to Python
 - Select a command and press `ctrl-I`
 - Information opens in help window
 - Enter object in help window
- `help(command)` in console

```
In [18]: help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n',
          file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by
    default.
    Optional keyword arguments:
    sep: string separator inserted between values, default
    '\n': string inserted between values, default '\n', and string
    inserted after the last value, default '\n'.
```



56

Jupyter

getting_started_jupyter.ipynb



58

Jupyter notebook

- A nice idea popularized by Mathematica is a “notebook” interface, where you can run and re-run commands
- In the notebook, you can easily mix code with comments, and mix code with the results of that code; including graphics, ...
- <https://realpython.com/jupyter-notebook-introduction/>
- <https://docs.anaconda.com/ae-notebooks/4.2.2/user-guide/basic-tasks/apps/jupyter/>
- <https://towardsdatascience.com/5-reasons-why-jupyter-notebooks-suck-4dc201e27086>



59

Jupyter notebook

- Excellent for
 - Explorative programming
 - Data exploration
 - Communication, especially across domains
- Problems
 - What was (re-)executed, what not?
 - Version control?
- https://github.com/gjbex/training-material/blob/master/Python/python_intro.pptx



60

Jupyter notebook vs Jupyterlab

- Basically, Jupyterlab is the new generation user interface for executing and editing notebook documents, similar to the Jupyter notebook.
- Jupyterlab is more advanced and offers more features,
- It gives a more IDE-like experience.
- Beginner: start with Jupyter notebook

61

Jupyter: how to start

- Anaconda Navigator:

- Start menu
- Launch jupyter

- Anaconda prompt



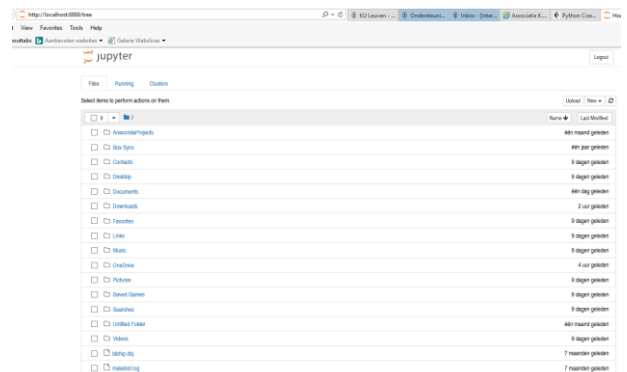
- open terminal and navigate to the **directory** where you would like to save your notebook
- `jupyter notebook`



62

Jupyter

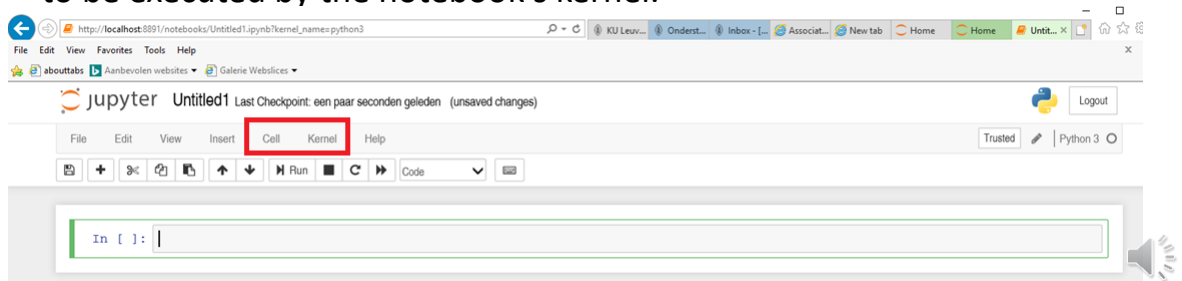
- Notebook Dashboard, specifically designed for managing your Jupyter Notebooks.
- Use it as the launchpad for exploring, editing and creating your notebooks.
- Start the dashboard on any system via the command prompt (or terminal on Unix systems):
`jupyter notebook`
The current working directory will be the start-up directory.



63

Jupyter notebook

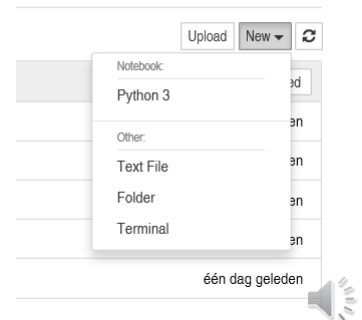
- Jupyter is essentially an advanced word processor.
- A kernel is a "computational engine" that executes the code contained in a notebook document.
- A cell is a container for text to be displayed in the notebook or code to be executed by the notebook's kernel.



64

Jupyter notebook

- Browse to the folder in which you would like to create your first notebook,
- Click the "New" drop-down button in the top-right and
- Select "Python 3" (or the version of your choice).



65

Jupyter: basics of editing

- Jupyter notebook: sequence of cells
 - Code
 - Label “In []” in front of the code
 - a * will appear when executing
 - replaced by a number that always increases by one with each cell execution. This allows for keeping track of the order in which the cells in the notebook have been executed.
 - Markdown
- Important shortcut: ctrl+Enter (execute cell)
- Color code
 - Blue bar on the left: active cell in command mode
 - Click in cell, changes in edit mode – Green bar
- Jupyter will periodically autosave the notebook

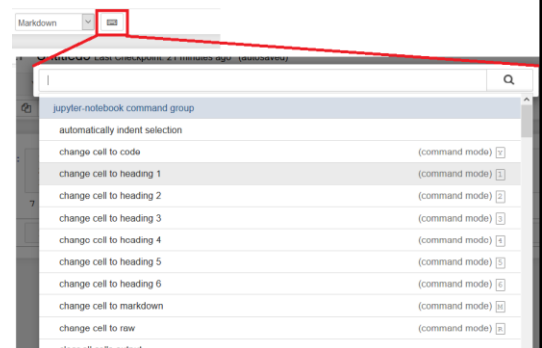


66

Jupyter: basics of editing



- try to know the basic shortcuts
- **Command mode** shortcuts:
 - Basic navigation: enter, **shift-enter**, up/k, down/j
 - Saving the notebook: s
 - Change Cell types: y, m, 1-6, t
 - m to change the current cell to Markdown,
 - y to change it back to code
 - Cell creation: a, b
 - a to insert a new cell above the current cell,
 - b to insert a new cell below
 - Cell editing: x, c, v, d, z
 - c copy selected cells
 - x cut selected cells
 - v paste copied cells
 - d + d (press the key twice) to delete the current cell
 - z undo cell deletion



67

Jupyter: some tips



- Run a notebook on the command line with `ipython`
- Jupyter notebook tips
<https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/>
- <https://www.dataquest.io/blog/jupyter-notebook-tutorial/>
- <https://jupyter4edu.github.io/jupyter-edu-book/>
- <https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/>
- Change the default startup directory
 - <https://stackoverflow.com/questions/35254852/how-to-change-the-jupyter-start-up-folder>
- Change the default browser
 - <https://support.anaconda.com/customer/en/portal/articles/2925919-change-default-browser-in-jupyter-notebook>

