COMP47350: Data Analytics (Conv)

Dr. Georgiana Ifrim
georgiana.ifrim@ucd.ie

Insight Centre for Data Analytics
School of Computer Science
University College Dublin

2016/17

Installation Instructions

- Python 3.5 and useful packages
- Jupyter Notebook

Python3.5 vs 2.7 (useful if using legacy Python code)

- Key differences
  - Using the `__future__` module
  - The print function
  - Integer division
  - Unicode
  - xrange
  - Raising exceptions
  - Handling exceptions
  - The next() function and .next() method
  - For-loop variables and the global namespace leak
  - Comparing unorderable types
  - Parsing user inputs via input()
  - Returning iterable objects instead of lists
- For more details:

http://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html

https://docs.python.org/3/tutorial/index.html

Install Python3.5: Anaconda

- **Anaconda:** Free Python distribution, includes popular Python packages for science, engineering, data analysis https://www.continuum.io/downloads

- Installing Anaconda for **Python3.5** for your Operating System (or basic Miniconda): http://conda.pydata.org/docs/install/quick.html

- Go to the shell/command line, check version of Python installed (Python3.5):

  python --version

Install Python3.5: Anaconda

- **Conda**: cross-platform package and environment manager
- A **virtual environment** is a folder on your computer where you can install all the required packages
- You can keep different projects in different virtual environments (e.g., some projects may require Pyhon2.7, some Python3.5)
- To create new **Python virtual environment** comp47350 and install packages run in shell:

conda create --name comp47350 numpy matplotlib scipy pandas scikit-learn

Install Python3.5: Anaconda

- Activate the newly created virtual environment:

source activate comp47350

- Install other required packages:

conda install nltk

If package not available with **conda**, install with **pip**:

pip install twitter

To deactivate this virtual environment:

source deactivate

Virtual Environment

- If you get an error that a Python package or function is not known, but you remember having installed it, most likely you forgot to activate the required virtual environment
- You need to repeat these steps:

source activate comp47350

- Run or install needed packages, e.g.,

jupyter notebook

conda install json

- If you are done with your work run:

source deactivate

How to run Python Code

- From the shell: Type python in the shell to start the Python interpreter. To stop: quit()
- From the shell: Run full script.

  python <your_script_file_name.py>
- From the browser: Use web-based interactive notebook: Jupyter Notebook
- From an IDE: Point-and-Click tools for software development (e.g., Spyder, PyCharm)

Interactive Python: Jupyter Notebook

- Jupyter Notebook (aka Ipython Notebook): Web application to create and share documents that contain code, equations, visualizations and text https://try.jupyter.org

- **Great for reproducible data analysis:** self-contained record of a computation
- Notebooks can be exported as HTML or PDF (nbconvert) and shared online (nbviewer)
- Notebooks are rendered by Github (i.e., can be visualized with a browser)
- We will use Jupyter Notebooks for most of our labs, homework and project

Installing Jupyter Notebook

First make sure to be in the virtual environment for the module:

source activate comp47350

conda install jupyter

Start a Jupyter Notebook (opens a web browser at http://localhost:8888/tree):

jupyter notebook

You can now see the files in the folder where you lunched the notebook. If you want to use a different folder for Jupyter Notebook, run in shell:

pwd

cd <change_to_your_desired_path_here>

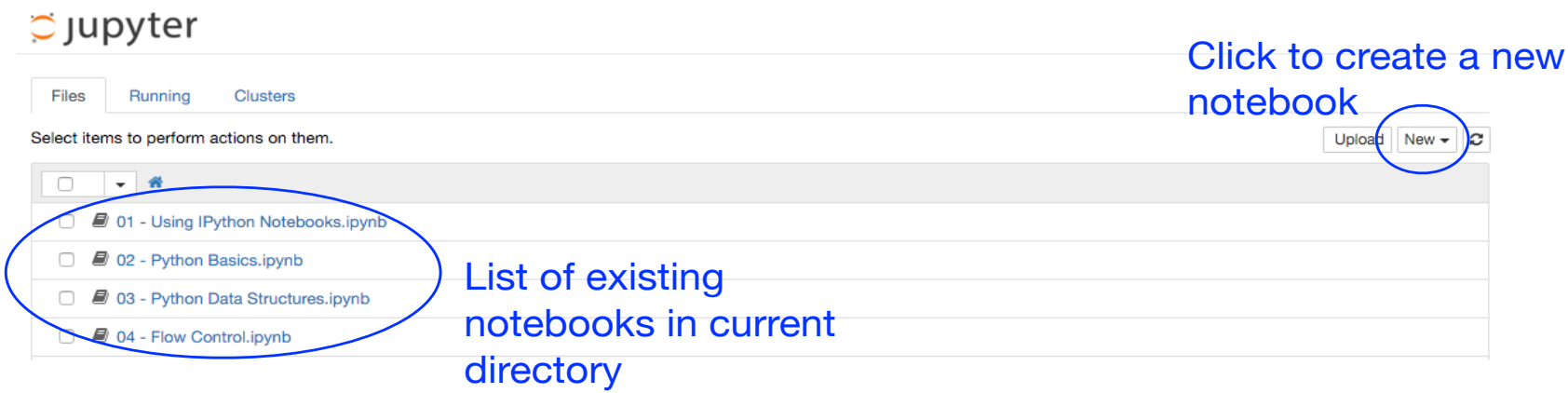
jupyter notebook

Notebook example: Python crash course

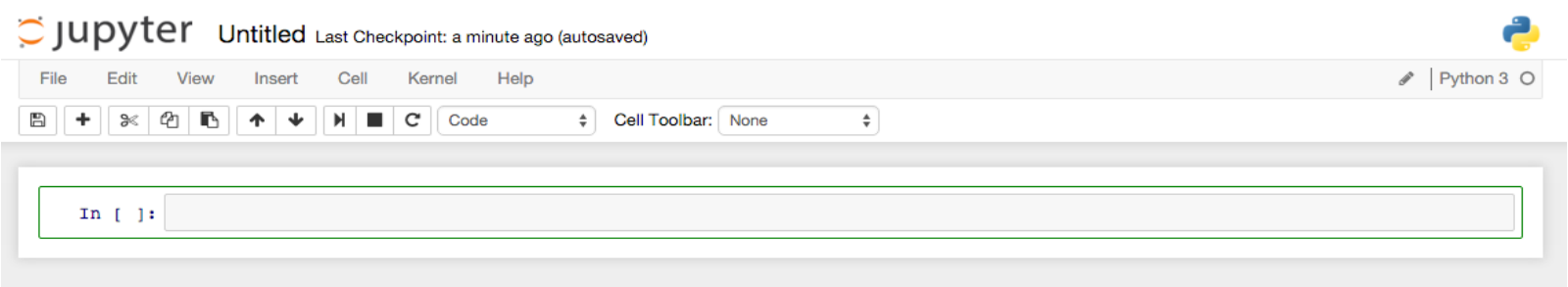
http://nbviewer.ipython.org/github/ipython-books/minibook-2nd-code/blob/master/chapter1/14-python.ipynb

Notebook Dashboard

- The IPython dashboard provides a mini filesystem interface for creating and accessing notebooks.
- Note: The dashboard shows notebooks in the directory where you launched the notebook server.

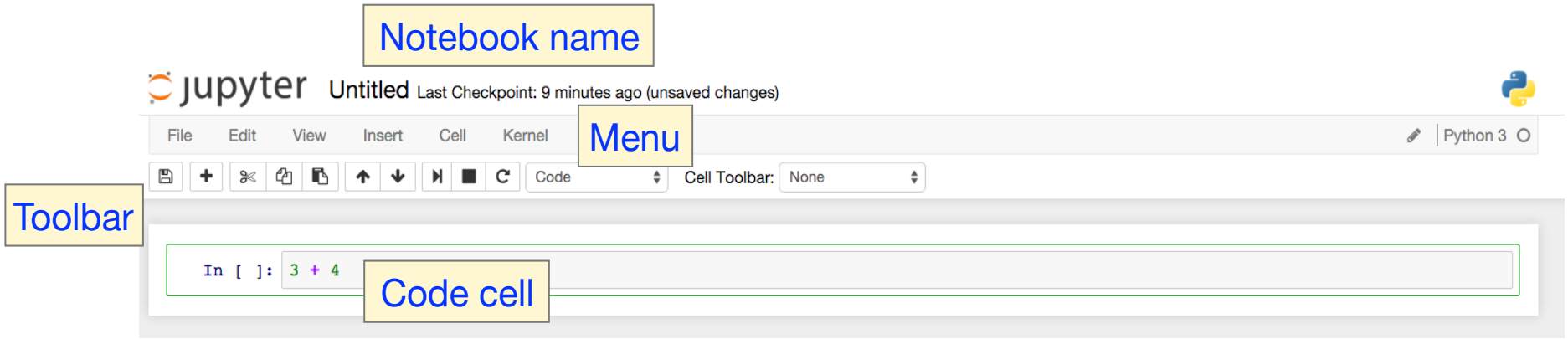


- To start writing code, create **New → Python 3 Notebook**



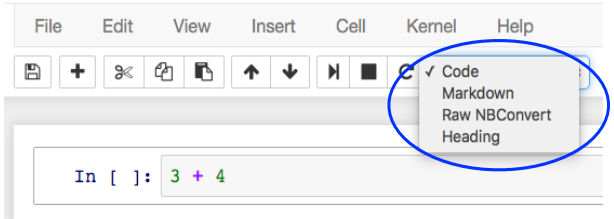
Notebook Interface

- When you create a new notebook, you will be presented with the notebook name, a menu bar, a toolbar and an empty code cell.



- IPython notebooks have two fundamental types of cells:

1. **Markdown cells:** Contain text content for explaining a notebook.
2. **Code cells:** Allow you to type and run Python code.



Every new cell starts off being a code cell. But this can be changed by using the drop-down on the toolbar

Basics for Jupyter Notebook

- Two types of cells: **code** and **markdown**
- **Code:** Writing and running Python code
- **Markdown:** Text for describing code
- Magic commands:
  - Functions that work with code: **%run, %edit, %save**
  - Functions which affect the shell: **%color, %autoindent, %automagic**
  - Other useful functions: **%timeit, %%writefile, %load**

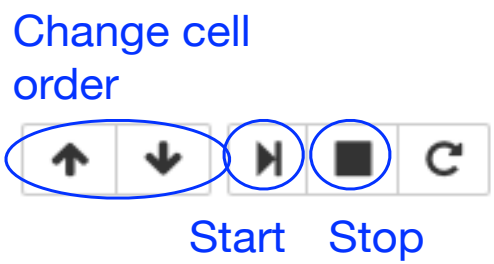
http://ipython.readthedocs.org/en/stable/interactive/index.html

http://nbviewer.ipython.org/github/ipython/ipython/blob/3.x/examples/Notebook/Index.ipynb

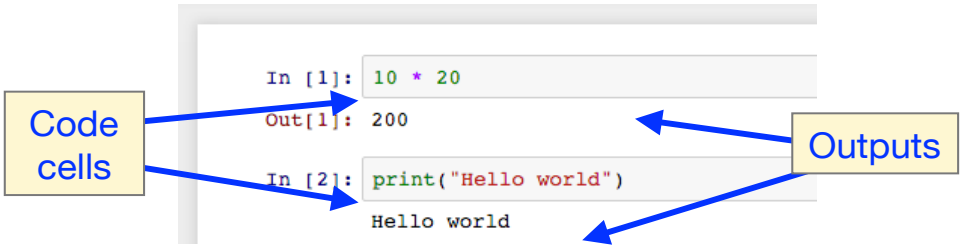
https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/

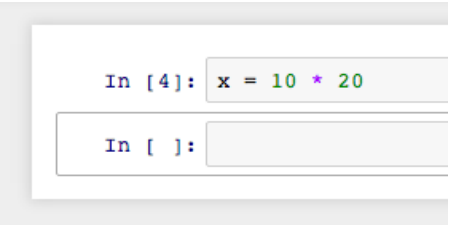
Code Cells

- In a code cell, you can enter one or more lines of Python code. Run the code by hitting Shift-Enter or by pressing the **Play** button in the toolbar.



- You can modify and re-run code cells multiple times in any order.
- When a code cell is executed, the code it contains is sent to the **kernel** associated with the notebook - i.e. the Python instance running in the background.
- The results returned from this computation are displayed as the cell's output. Note that some code will not have an output.





No visible output cell

- Restarting the kernel associated with a notebook clears all previous history (e.g. variable values).

Markdown Cells

- It can be helpful to provide explanatory text in notebooks.
- **Markdown** is a lightweight type of markup language with plain text formatting syntax which can be rendered as HTML.
- IPython supports a set of common Markdown commands. HTML tags and LaTeX formulae can also be included.
- When a Markdown cell is executed, the Markdown code is converted into the corresponding formatted rich text.

```
This is normal text.
```

This is normal text.

```
*This is italics*.
```

This is italics.

```
And **this is bold**.
```

And **this is bold**.

```
# Heading 1
## Heading 2
### Heading 3
```

Heading 1
Heading 2
Heading 3

```
Example <font color='red'>HTML use</font>
```

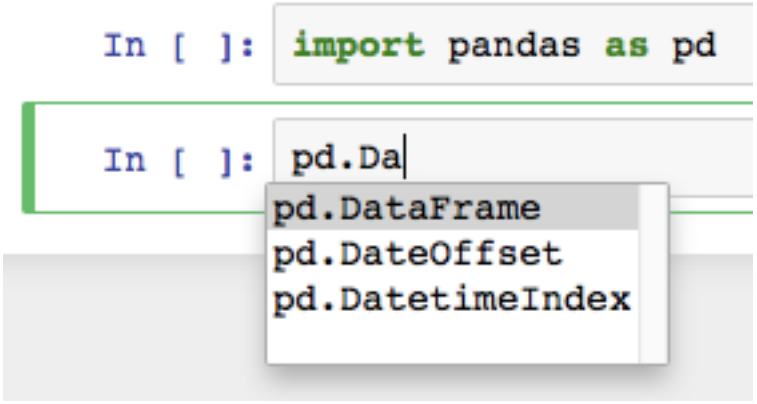
Example HTML use

```
Formula: $x=\frac{y}{z}$
```

Formula: $x = \frac{y}{z}$

Using Notebooks

- Can download any Notebook from the Web and open it with your local Jupyter Notebook installation
- Useful for re-using code and learning from other notebooks
- Useful for showing all the steps of a data analysis, tutorial style
- **Very handy tip: use Tab for auto-completion** of your Python commands in Jupyter Notebook, e.g., type **pd.D** and press Tab (it will show you a popup box with auto-completion options)



Python IDE: Spyder or PyCharm

For bigger projects a good IDE is important. We will only use Jupyter Python in this module, but its good to know about IDEs: PyCharm or Spyder (like Rstudio, if you worked with R before).

Spyder is open-source:

conda install spyder

Run Spyder from Terminal:

spyder

PyCharm has free and commercial versions:

https://www.jetbrains.com/pycharm-edu/

Backing Up

- Get used to backing up your work every day
- Jupyter Notebook has versioning (saves intermediate work) but you may still loose the homework right before submission (e.g., if accidentally deleting the .ipynb file)
- Use Github and push your work to Git every day!

References

Python 3
Official documentation: https://docs.python.org/3/

Jupyter Notebook
Official documentation:
http://ipython.readthedocs.org/en/stable/overview.html

Markdown
Guide to Markdown
https://help.github.com/articles/markdown-basics
Original Markdown syntax specification
http://daringfireball.net/projects/markdown/syntax/

Conda
Official documentation: http://conda.pydata.org/docs/index.html