COMP30810
Intro to Text Analytics

Dr. Binh Thanh Le
thanhbinh.le@ucd.ie
Insight Centre for Data Analytics
School of Computer Science
University College Dublin

Installation Instructions

- Anaconda
- Python 3.6
- Jupyter Notebook

Install Python3.6 with Anaconda

- **Anaconda:** Free Python distribution, includes popular Python packages for science, engineering, data analysis
- Overview of main Python DA packages:
  - https://www.anaconda.com/distribution/
- Download Python 3.6 distribution:
  - https://www.anaconda.com/download
- Installing Anaconda for **Python3.6** for your Operating System (or basic Miniconda):
  - https://docs.anaconda.com/anaconda/install/
- Go to the shell/command line, check version of Python installed (you should see python3.6):

```
python --version
```

Python Virtual Environment

- **Conda**: 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 Python3.6)
- To create a new **Python virtual environment** named **comp30810** and to install some packages run in the shell:

```
conda create --name comp30810py36 python=3.6 numpy matplotlib scipy pandas scikit-learn nltk
```

Activate Virtual Environment

- Activate the newly created virtual environment:

  [source] activate comp30810py36

- Install other required packages:

  conda install requests

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

  pip install nltk

- To deactivate (i.e., get out of) 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
- Activate the virtual environment:

  [source] activate comp30810py36
- Run needed packages, e.g.:

  jupyter notebook
- If not installed, install needed package, e.g.,:

  conda install jupyter
- If you are done with your work, de-activate the virtual environment:

  [source] deactivate
- To check list of already installed packages:

  pip list

  conda list

Pro TIP: Exporting a Virtual Environment

- If you work with Python and need to run your code/project later on, or on a new machine, export your virtual environment using:

```
pip freeze --all > pip-freeze-venv_name-date.txt
```

- Example:

```
pip freeze --all > pip-freeze-venv_comp30810py36-100918.txt
```

- This exports all Python packages and their exact version installed at that time (when your code was working) in your virtual environment
- To setup the virtual environment on a new machine do:

```
pip install -r pip-freeze-venv_comp30810py36-100918.txt
```

- You can call your file whatever you want, e.g., requirements.txt, required packages.txt, etc.

How to run Python Code

For file ".py"

- From the shell (aka command line) type: python
- To stop the Python interpreter type: quit()
- From the shell, run a full Python file/script.

  python <your_script_file_name.py>

- Using IDE, e.g. Pycharm, Spider, … to open
- Run using IDE. Note that python should be installed in IDE platform

For file ".ipynb"

best

- From a browser: to use the web-based interactive Jupyter Notebook in the browser, in the shell run:

  jupyter notebook

- Then click and open that file with notebook

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, and homeworks.**

Installing Jupyter Notebook

- Make sure to be in the virtual environment for the module:

  [source] activate comp30810py36

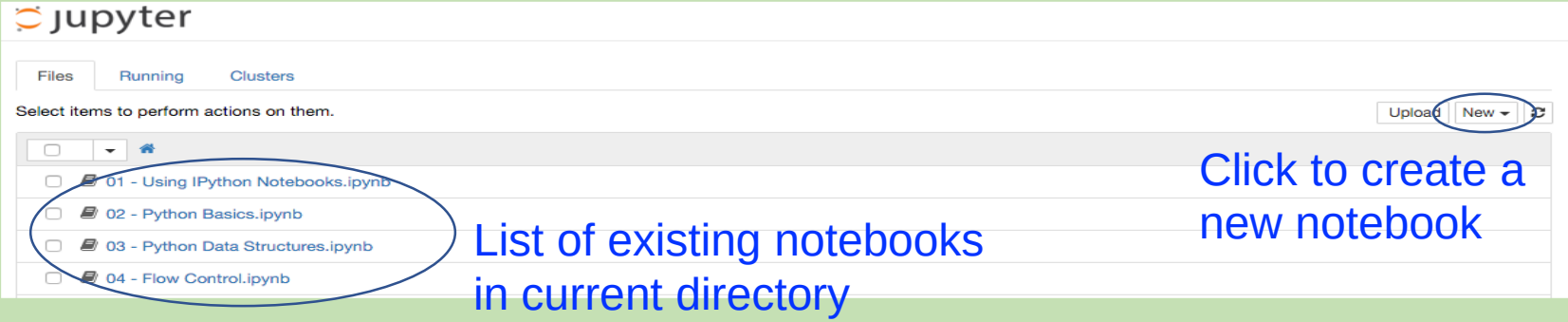
- If not installed already, install the Python package for Jupyter Notebook:

  conda install jupyter

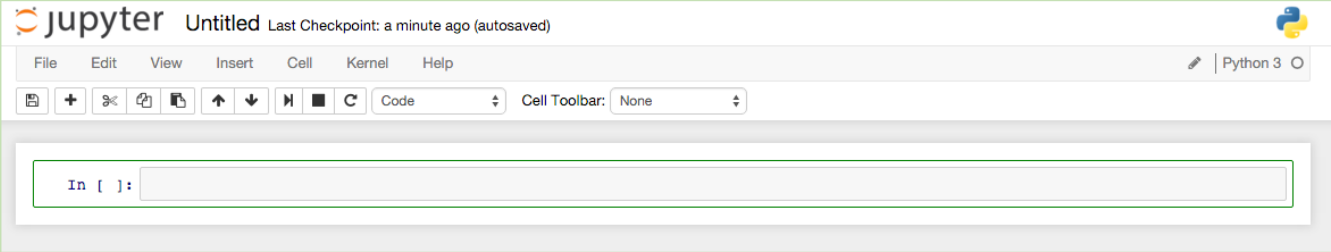
- Start a Jupyter Notebook: jupyter notebook
- Opens a web browser at:
  http://localhost:8888/tree
- Notebook example: Python crash course
  http://nbviewer.ipython.org/github/ipython-books/minibook-2ndcode/blob/master/chapter1/14-python.ipynb

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.

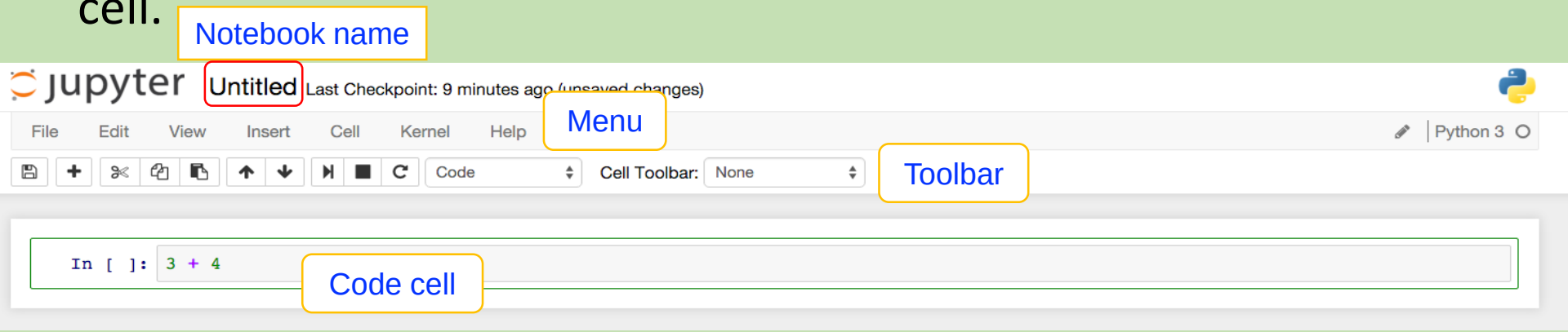


- To start writing code, create **New → Python 3 Notebook**



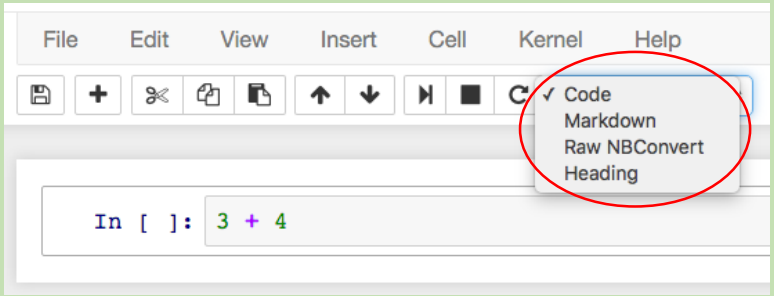
Notebook Interface

- 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

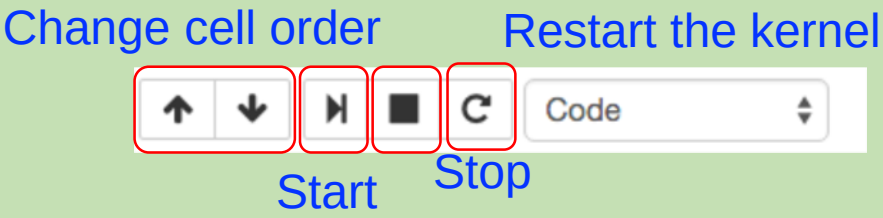
Basics for Jupyter Notebook

- Two types of cells: **code** and **markdown**
  - **Code:** Writing and running Python code
  - **Markdown:** Text for describing the problem and the code

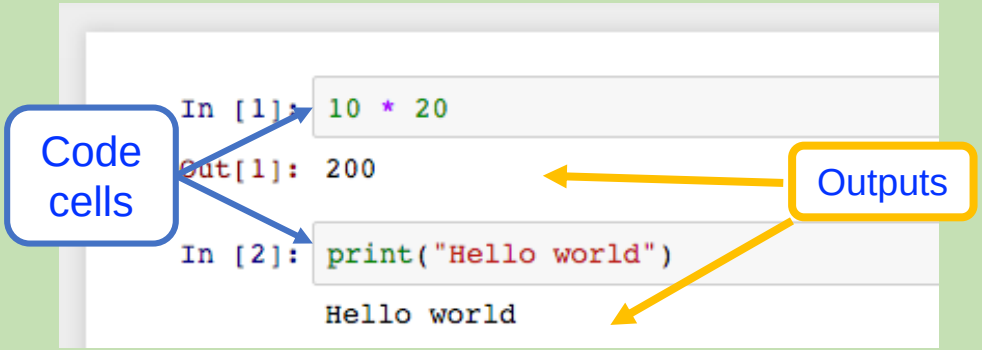
- Jupyter documentation:
  - http://jupyter.readthedocs.io/en/latest/
  - http://nbviewer.ipython.org/github/ipython/ipython/blob/3.x/examples/Notebook/Index.ipynb
- Jupyter tips and tricks:
  - https://www.dataquest.io/blog/jupyter-notebook-tips-tricksshortcuts/

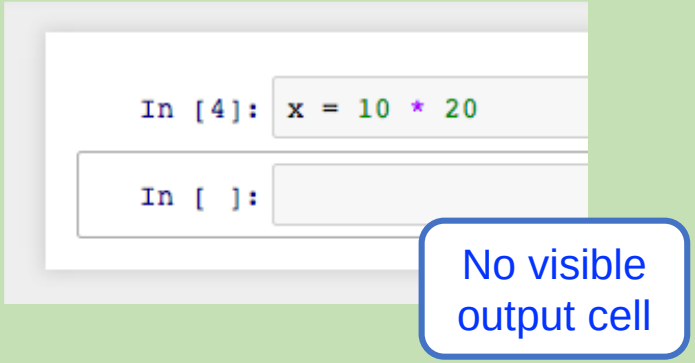
Code Cells

- 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.





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.

```
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>
```

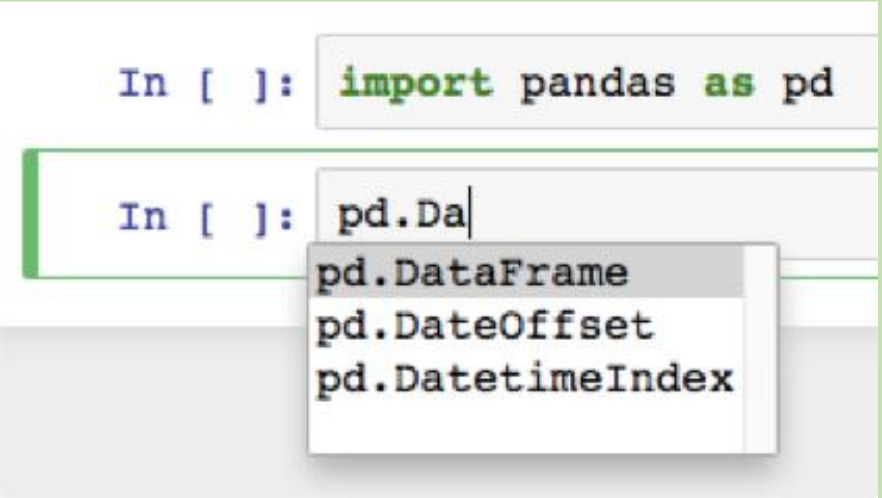
Example HTML use

```
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 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)



Backing Up

- Get used to backup your work every day (using a backup hard-drive or Github)

- 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! It is worth the extra effort, and it forces you to structure your work better.

References

- **Python 3**

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

- **Jupyter Notebook**

Official documentation:
http://jupyter.readthedocs.io/en/latest/

- **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