Intro to Data Analysis in Python

PyLadies Vancouver Workshop

July 7, 2018

Instructor: Jennifer Walker

Agenda

- Navigating the Python world as a data geek
- Jupyter Lab orientation + quick recap of Python basics
- Working with spreadsheet data
  1. Reading and summarizing CSV files
  2. Basic calculations and graphs
  3. Text data and messy / missing data
  4. Sorting, aggregation, and subsets
- Data visualization: a brief tour of the Python landscape
- Next steps, ideas, and inspiration

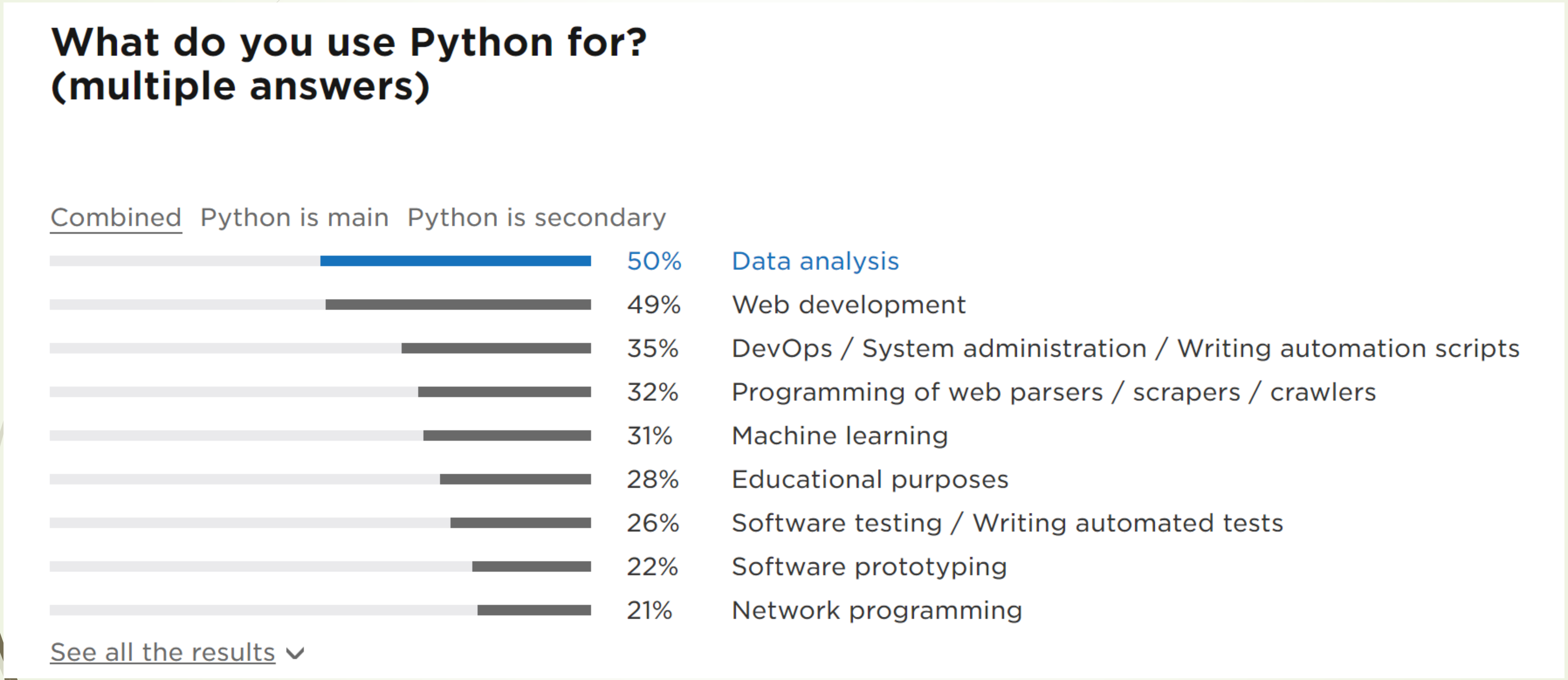
Navigating the Python world

… as a data geek

- Python is used in a huge variety of applications

- It has recently become a powerhouse for data analysis

Python Developers Survey 2017



What do you use Python for? (multiple answers)

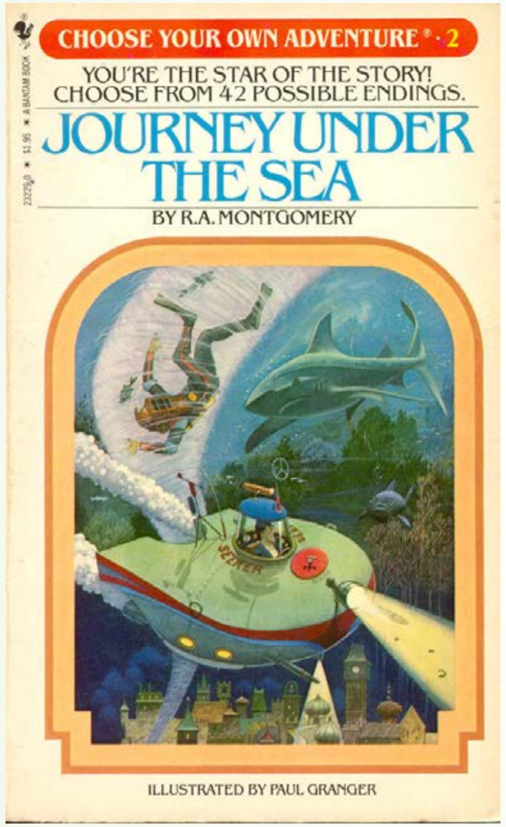
Combined | Python is main | Python is secondary

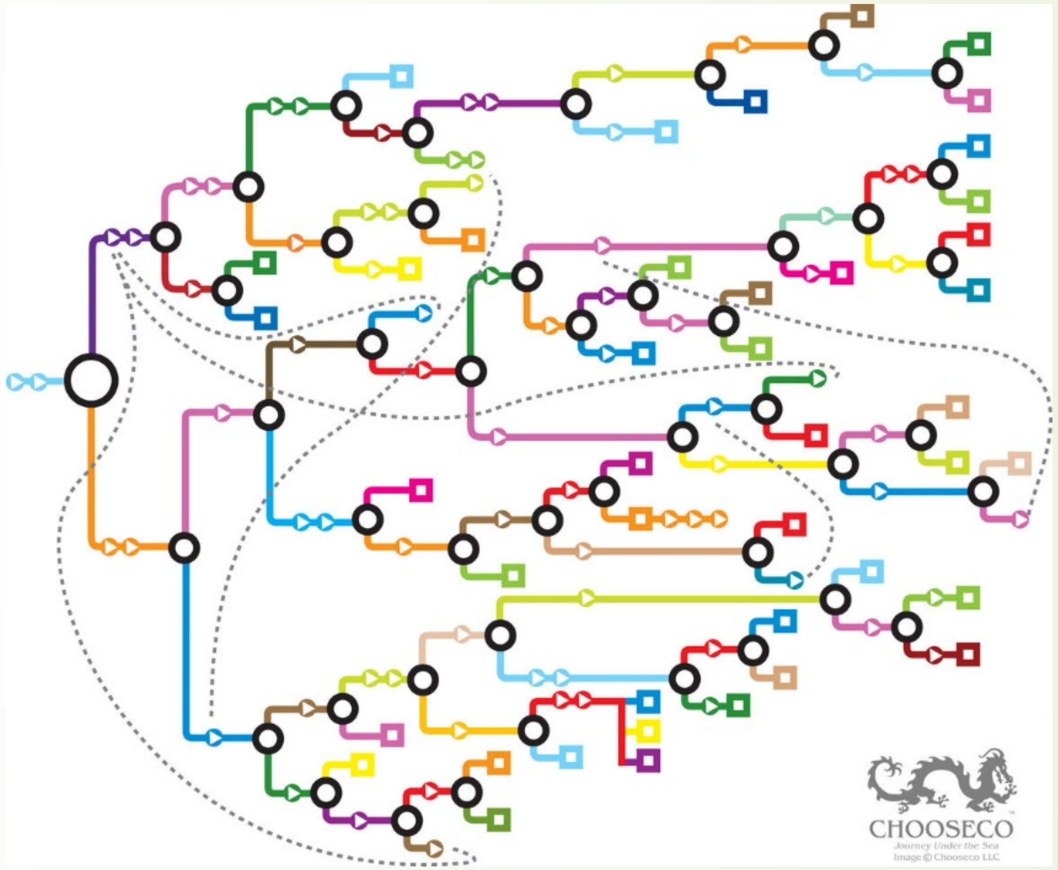
50%	Data analysis
49%	Web development
35%	DevOps / System administration / Writing automation scripts
32%	Programming of web parsers / scrapers / crawlers
31%	Machine learning
28%	Educational purposes
26%	Software testing / Writing automated tests
22%	Software prototyping
21%	Network programming

See all the results

https://www.jetbrains.com/research/python-developers-survey-2017

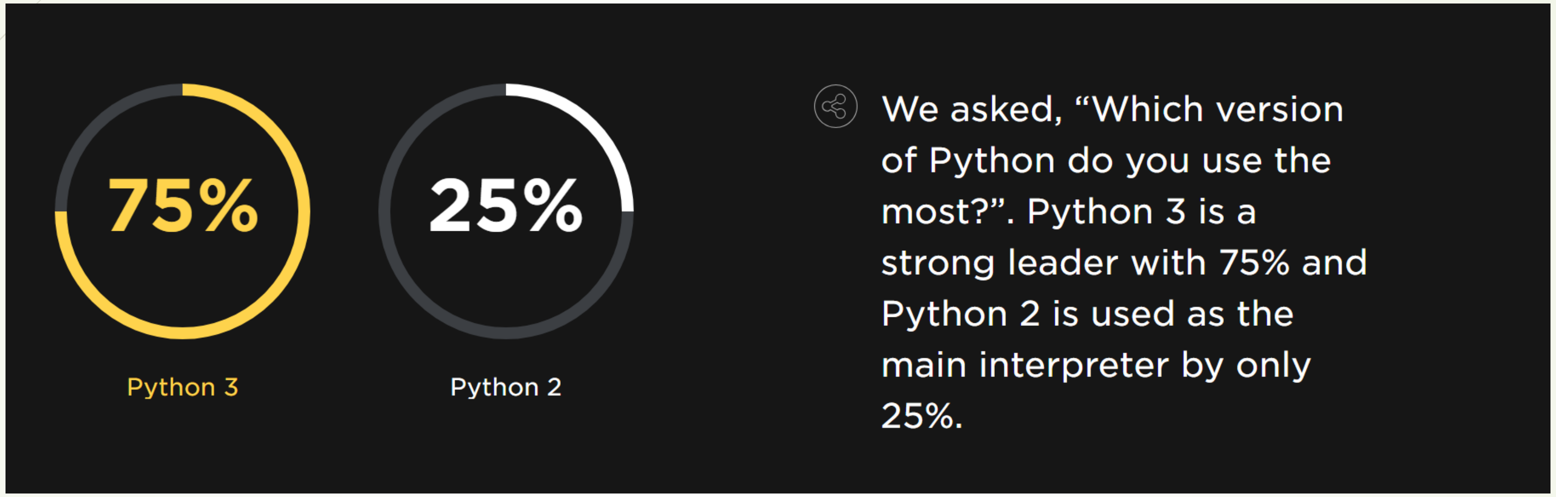
Choose Your Own Adventure


CHOOSE YOUR OWN ADVENTURE® · 2
YOU'RE THE STAR OF THE STORY!
CHOOSE FROM 42 POSSIBLE ENDINGS.
JOURNEY UNDER THE SEA
BY R.A. MONTGOMERY
ILLUSTRATED BY PAUL GRANGER

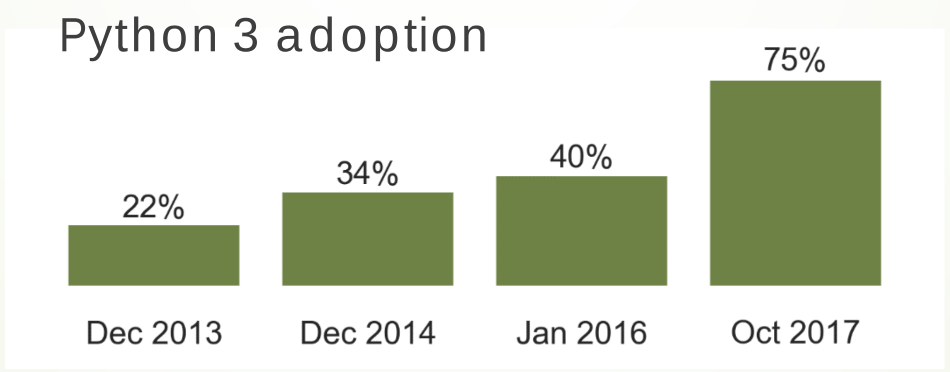


https://www.atlasobscura.com/articles/cyoa-choose-your-own-adventure-maps

Python 3 vs. Python 2



We asked, “Which version of Python do you use the most?”. Python 3 is a strong leader with 75% and Python 2 is used as the main interpreter by only 25%.



https://www.jetbrains.com/research/python-developers-survey-2017

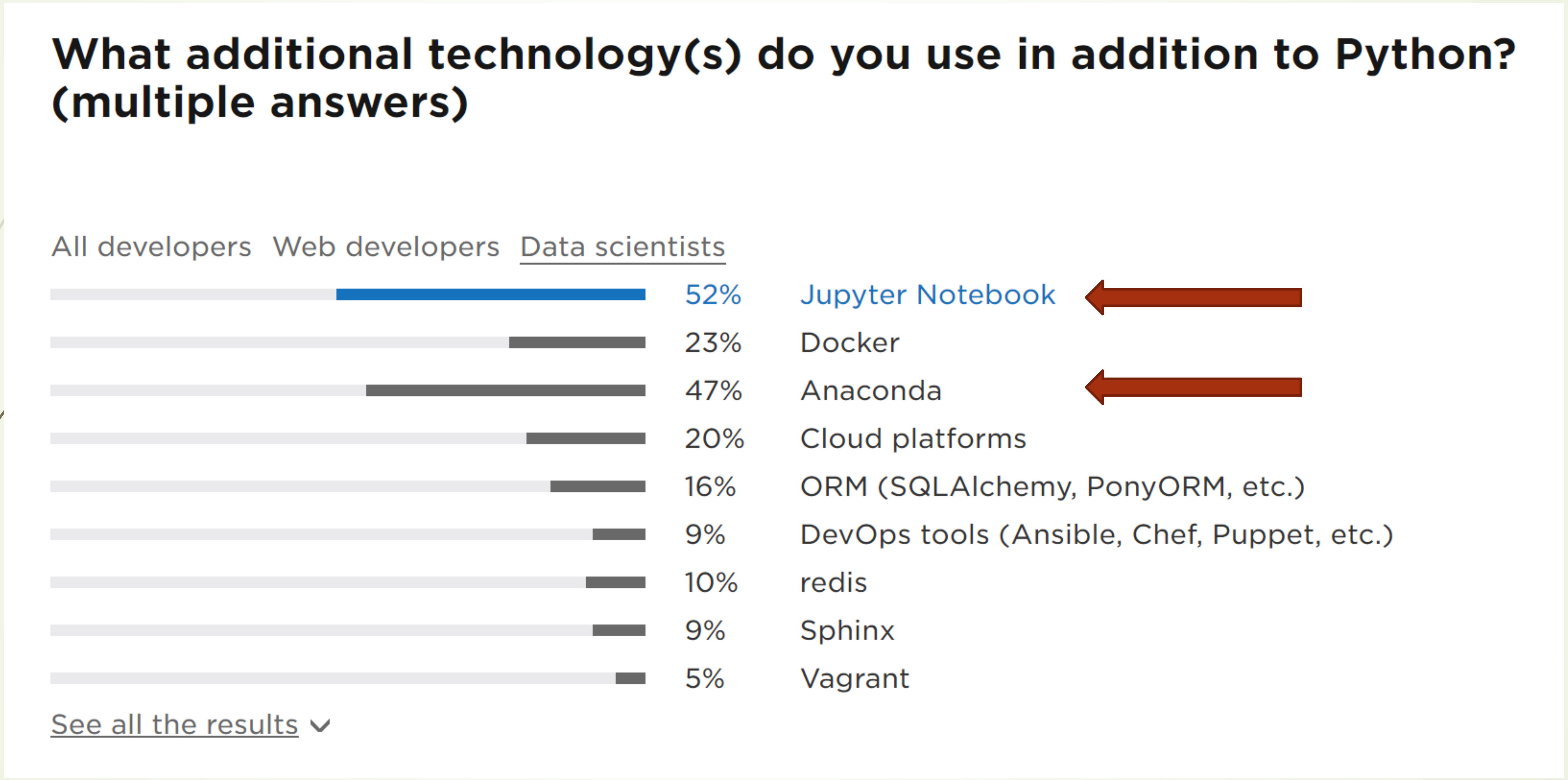
Jupyter Lab

- Development environment for working with data
  - Human-centered, interactive coding
- We'll be using Jupyter **notebooks:**
  - Code, graphs, formatted text, equations, etc. in a single document
    - Ideal for exploratory data analysis and for sharing your work with others who are interested in the entire workflow (step by step presentation of your code and results)
  - Uses an IPython kernel to run Python code (IPython = Interactive Python)
    - Many handy features for a much better interactive experience compared to the standard Python console
  - Also supports R, Julia, Perl, and over 100 other languages (and counting!)

Example Jupyter Notebook

https://www.kaggle.com/arthurtok/generation-unemployed-interactive-plotly-visuals

Anaconda and Jupyter



What additional technology(s) do you use in addition to Python? (multiple answers)

All developers Web developers Data scientists

52%	Jupyter Notebook
23%	Docker
47%	Anaconda
20%	Cloud platforms
16%	ORM (SQLAlchemy, PonyORM, etc.)
9%	DevOps tools (Ansible, Chef, Puppet, etc.)
10%	redis
9%	Sphinx
5%	Vagrant

See all the results

https://www.jetbrains.com/research/python-developers-survey-2017

Python vs. Other Software

python™

Free, open source

Ecosystem of tools developed separately by different teams

Sometimes need to fiddle with settings or implement a few extra steps to get things working

Dynamic, rapidly evolving

- New releases of a tool sometimes introduce drastic changes and incompatibilities with previous versions and other tools

Excel MATLAB

$$$, proprietary, closed source

Self-contained software developed by a single company

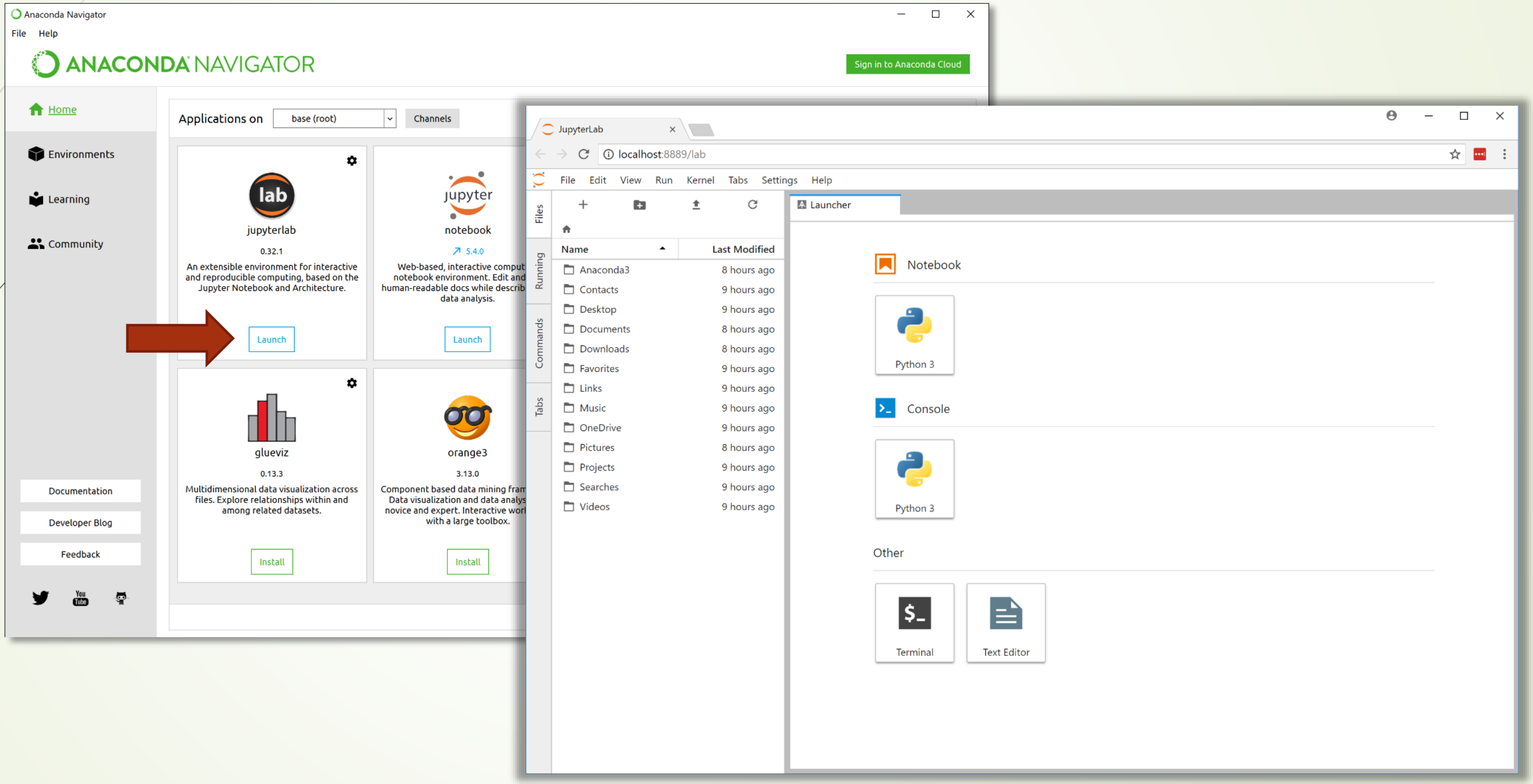
Works “out of the box” on any computer

Gradual and less frequent changes

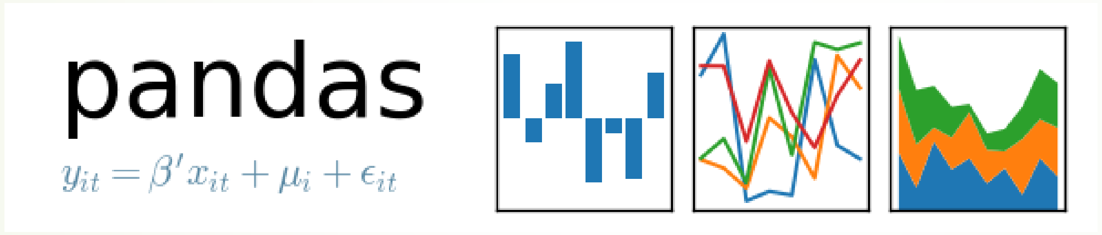
No need to panic!

- *Google search*
- *Tons of online resources*
- *Huge community of Python users helping each other out*
- *You’re not alone!*

Let’s Get Coding!



Data Visualization



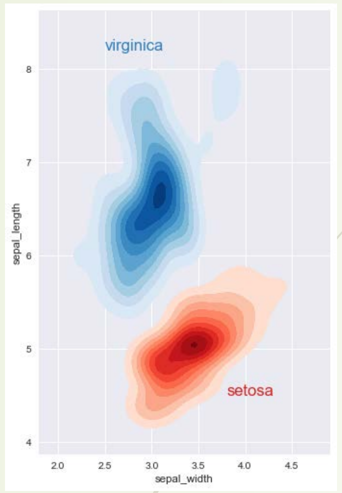
matplotlib

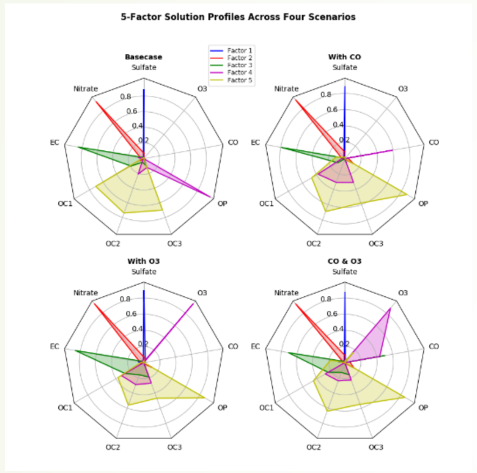
seaborn

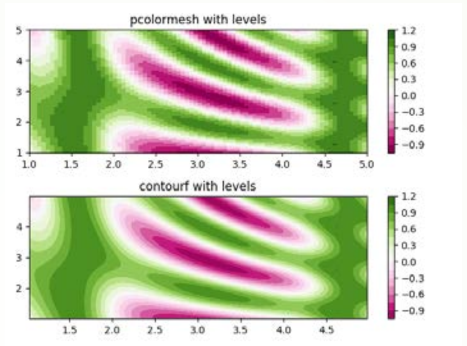
Bokeh

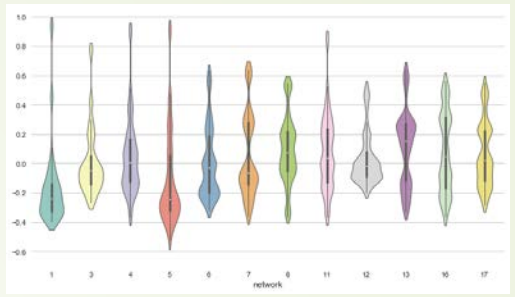
plotly

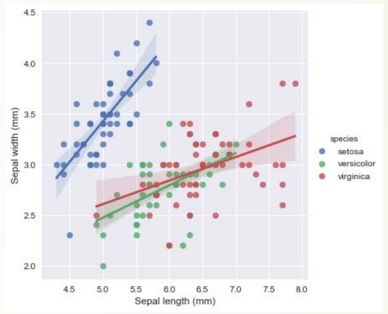
matplotlib & seaborn

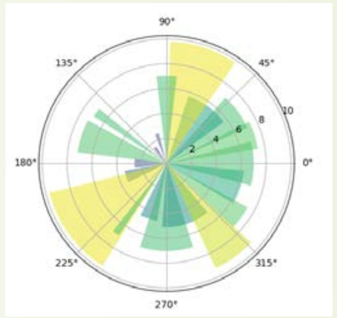


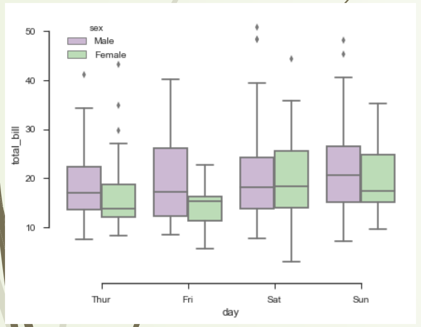


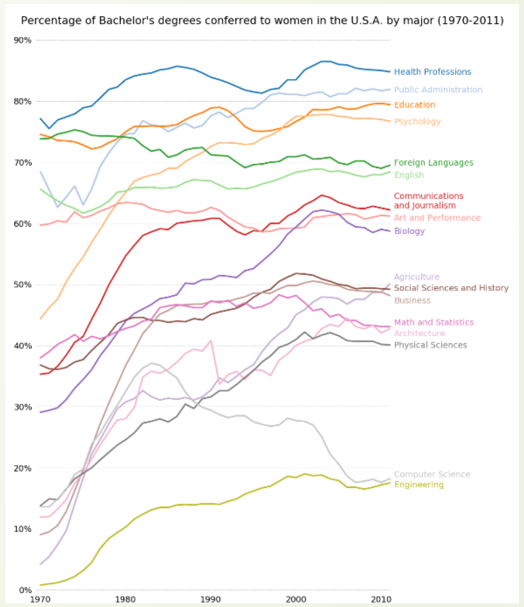


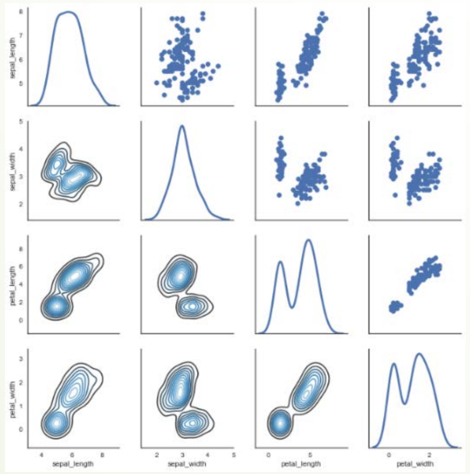


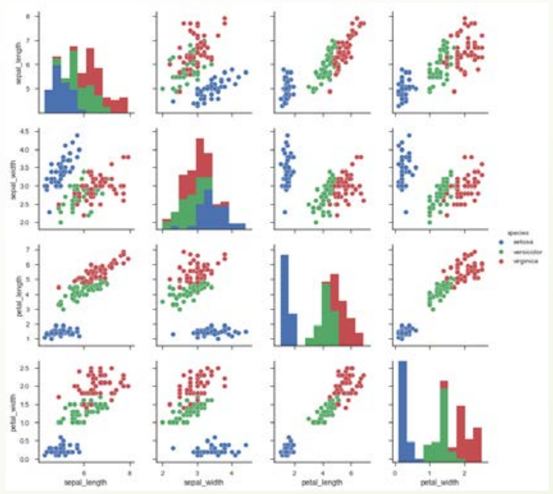




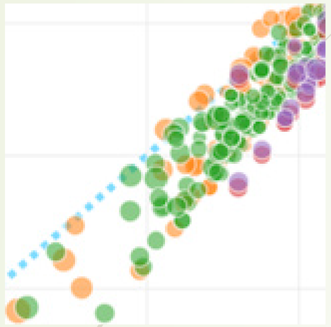
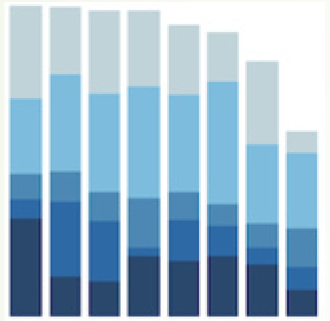
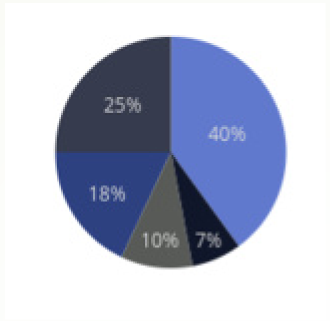
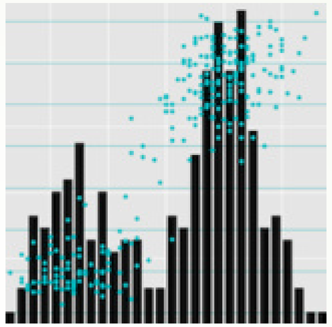
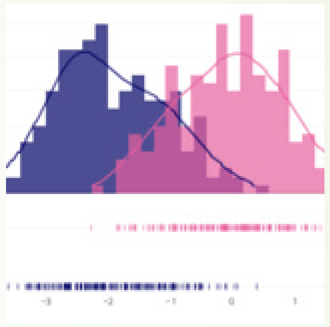
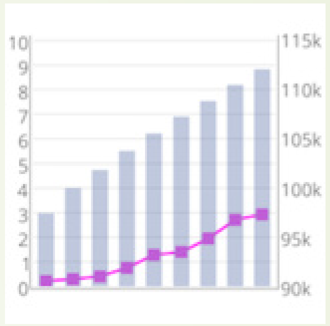
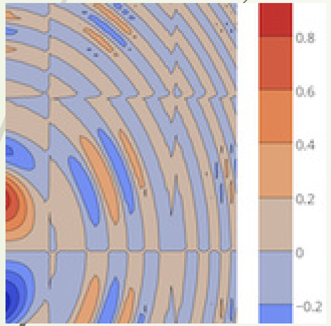
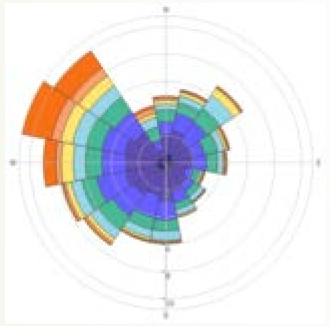
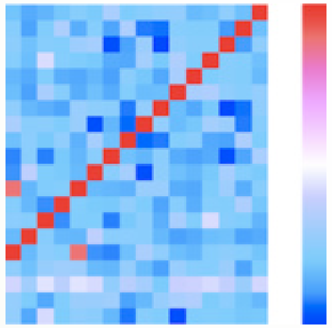






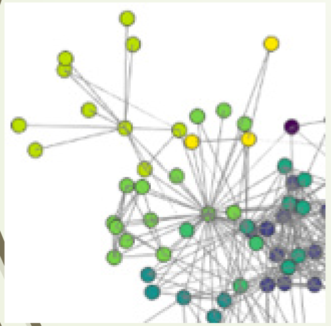
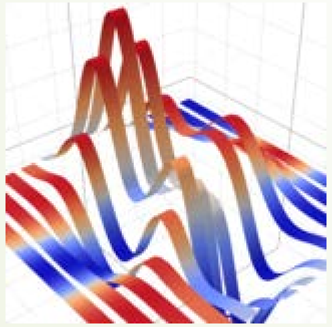
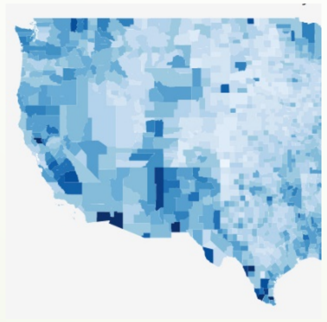


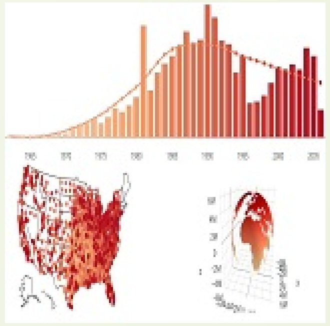
plotly

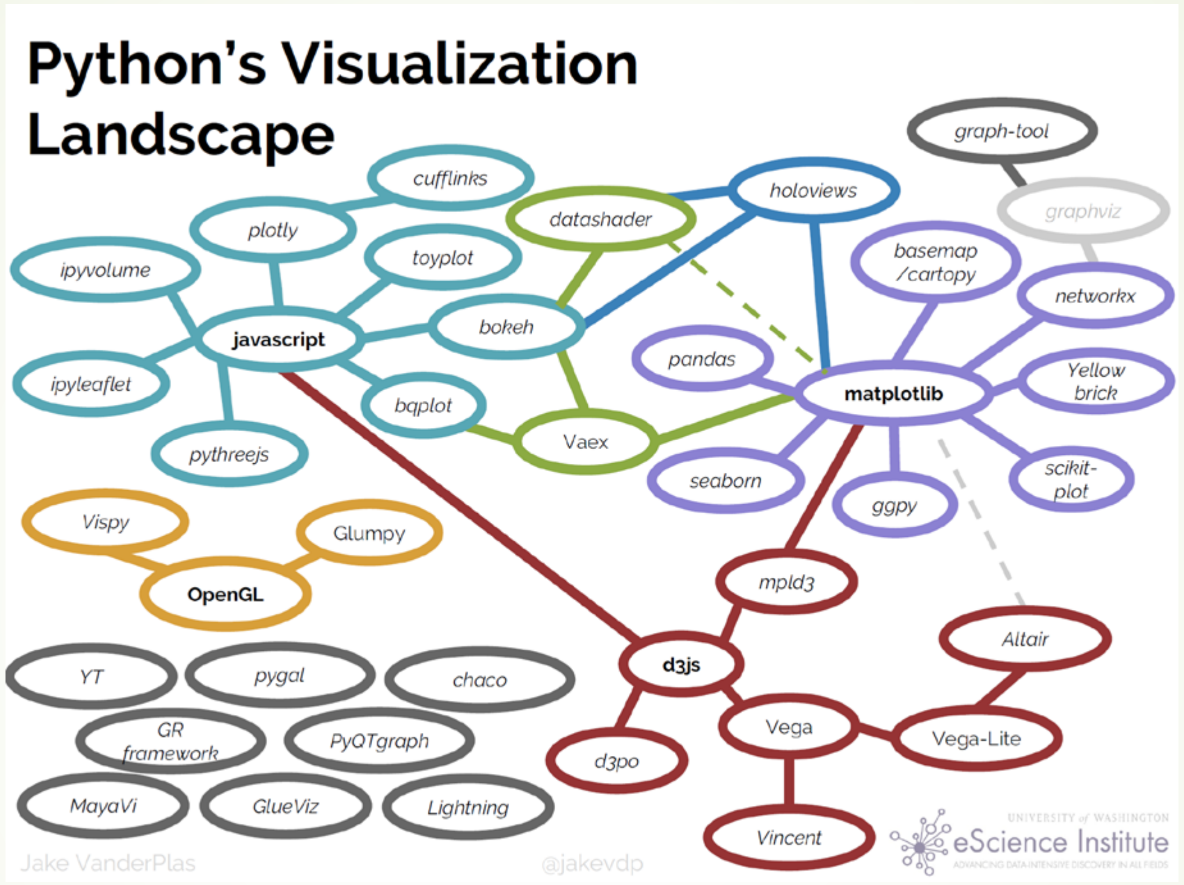



25%
40%
18%
10%
7%



10
9
8
7
6
5
4
3
2
1
0
115k
110k
105k
100k
95k
90k

0.8
0.6
0.4
0.2
0
-0.2



Aug 11
Aug 25
Sep 8
Sep 22
Oct 6

Funnel Chart





- We've only looked at a tiny fraction of the visualization tools available in Python
- To learn about some of the other amazing visualization libraries and how they fit into the bigger picture, check out this fantastic talk by Jake VanderPlas from PyCon 2017: https://www.youtube.com/watch?v=FytuB8nFHPQ

Visualization Examples & Resources

- pandas: https://pandas.pydata.org/pandas-docs/stable/visualization.html
- matplotlib: https://matplotlib.org/gallery/index.html
- seaborn: https://seaborn.pydata.org/examples/index.html
- Interactive plots:
  - plotly: https://plot.ly/python/
  - bokeh: https://bokeh.pydata.org/en/latest/docs/gallery.html
- Maps:
  - plotly: https://plot.ly/python/#maps
  - cartopy: https://scitools.org.uk/cartopy/docs/v0.15/gallery.html
  - folium: http://folium.readthedocs.io/en/latest/

Where to go from here?

- Online resources and courses:
  - Data Carpentry: http://www.datacarpentry.org/python-ecology-lesson/
  - Data Camp: https://www.datacamp.com/courses/intro-to-python-for-data-science
  - Dataquest: https://www.dataquest.io/
    - Excellent blog with great tutorials and useful articles: https://www.dataquest.io/blog/
  - Kaggle: https://www.kaggle.com/learn/overview
    - Many more example Jupyter notebooks and tutorials: https://www.kaggle.com/kernels
    - Tons of datasets to play with: https://www.kaggle.com/datasets
  - plus Coursera, Udemy, and many others
- Book: Python for Data Analysis, by Wes McKinney. All data and code from the book is at https://github.com/wesm/pydata-book

Ideas & Inspiration

- PyData 101 – Jake VanderPlas
  - https://www.youtube.com/watch?v=DifMYH3iuFw
- Reproducible Data Analysis in Jupyter – Jake VanderPlas
  - http://jakevdp.github.io/blog/2017/03/03/reproducible-data-analysis-in-jupyter/
- Project Jupyter: From Interactive Python to Open Science - Fernando Perez
  - https://www.youtube.com/watch?v=xuNj5paMuow
- The Next Generation of Data Products – Hilary Mason
  - https://www.youtube.com/watch?v=OuRlNNSDtlM

Thank You!

Jupyter Lab Orientation

Summary Notes

Getting Started with Jupyter Lab

- Opening Jupyter Lab
  - From Anaconda Navigator: click the “Launch” button under the “lab” icon
  - From the command line (Anaconda Prompt on Windows, Terminal on Mac), run the command: `jupyter lab`
- Jupyter Lab documentation: http://jupyterlab.readthedocs.io/en/stable/

Getting Started with Notebooks

- Navigate the “Files” sidebar in Jupyter Lab to change the working directory to your `pydata-intro-workshop` directory for this workshop
- Click the “Files” button to hide / show the “Files” sidebar
- Use the Launcher to create a new Python notebook in the current directory
  - Notice a new file “Untitled.ipynb” appears in the “Files” sidebar
  - The .ipynb extension stands for “IPython notebook” (the original name of the notebook, before it was changed to “Jupyter notebook”)
- Rename your new notebook from “Untitled.ipynb” to “Workshop.ipynb”
  - If the name hasn’t updated in the “Files” sidebar, click the Refresh button

Getting Started with Notebooks

- Notebook auto-saves periodically (like a Google document)
  - Can also manually save at any time with Ctrl-S (Cmd-S on Mac) or by clicking the Save icon on the top toolbar
  - Next time you open up Jupyter Lab, you can return to your “Workshop.ipynb” notebook by double-clicking it in the “Files” sidebar
- A notebook is a collection of cells, where you can run small chunks of code one at a time
  - Our new notebook starts off with a single, empty cell
- Enter some code into the cell (e.g. 2 + 2) and press Shift-Enter
  - The code runs, displays the output, and advances to the next cell (creating a new cell below if none exists)

Getting Started with Notebooks

- Create some more cells and run some more arithmetic operations
- Notice to the left of the first cell, it says “In[1]” and “Out[1]”, and so on for the second, third, etc. cells
  - The number inside “In[ ]” and “Out[ ]” increases by 1 each time you run a cell
  - Try running your last cell again and notice how the number changes
  - Go back to an earlier cell and run it again and notice how the number changes
  - You don’t generally need to worry about these numbers, but they can give you a heads up if you’ve been running your cells out of order

Notebook Structure

- Notebooks can be nonlinear
  - Unlike a script, where the code in a file is executed a single time, from top to bottom, the cells in a notebook can be executed out of order and multiple times each
  - You can also move cells around a notebook by dragging and dropping them
  - You can insert a new cell below a selected cell by clicking the + icon in the top menu
  - Executing code cells out of order can cause unexpected or confusing behaviour
    - This is something to be mindful of when writing notebooks
    - Following some best practices can help you avoid problems

Notebook Kernel

- From the top menu, select Kernel ➔ Restart Kernel and Clear All Outputs, and see what happens in your notebook
- Then from the top menu, select Run ➔ Run All Cells, and see what happens
- The above two steps can be consolidated into a single step: Kernel ➔ Restart Kernel and Run All Cells
  - As you're writing your notebook, periodically running the above command can help ensure that bugs haven't crept in due to running cells out of order
- Check out the items in the Kernel menu and the Run menu for more handy stuff, for example running a selection of several cells, or running all cells above or below a selected cell

Notebook Kernel

- You can see what kernel sessions are currently running by selecting the “Running” sidebar
  - When you close a notebook’s tab in Jupyter Lab, its kernel keeps running in the background
  - To shut it down, you can select “Shutdown” next to the notebook in the “Running” sidebar, or you can quit Jupyter Lab (see later slide)

Formatted Text: Markdown Cells

- Select an empty cell and from the top menu bar, click where it says "Code", and from the dropdown menu select "Markdown"
  - This changes the cell from a Code cell to a Markdown cell, where you can use the Markdown language to quickly format text, insert images, show equations, and more:
    - https://www.markdownguide.org/getting-started
    - https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Formatted Text: Markdown Cells

- In your Markdown cell, type some lines of text such as the example below, then press Shift-Enter to render the cell into formatted text

```
# Python is Awesome

## And so is Jupyter

Here is some regular text. Here are some *words in italic* and
some **words in bold**. The area of a circle is $A = \pi r^2$.

- Bullet item 1
  - Sub-bullet 1a
- Bullet item 2
- Bullet item 3
```

- To edit a Markdown cell, you need to double click inside it (for a code cell, you only need to single click inside it)
    - Add or remove some lines, and make any other edits you like, and then press Shift-Enter again to display the changes

Splitting and Merging Cells

- A cell can be split into two separate cells
  - Double click inside your Markdown cell to enter edit mode, and put the cursor at the start of one of the lines
  - From the top menu, select Edit ➔ Split Cell
  - Voila! Two separate cells. You can select them and move them around to change the order too.
  - You can also do this with Code cells, when you have multiple lines of code in a cell
- Cells can be merged into a single cell
  - Select two or more Markdown cells, and then from the top menu select Edit ➔ Merge Selected Cells
  - Code cells can be merged the same way

Keyboard Shortcuts

- There are many handy keyboard shortcuts
  - With a cell selected (but not in edit mode)
    - Press Enter to get into edit mode
    - Type A to insert a new cell above or B to insert a new cell below
    - Type M to convert to a Markdown cell or Y to convert to a Code cell
    - Type DD to delete the cell
  - For list of keyboard shortcuts for other commands, check out the “Commands” sidebar

Tabbed and Tiled Views

- You can have multiple notebooks and other files open in Jupyter Lab, and arrange them in different ways
  - From the “Files” sidebar, double-click “Lessons.ipynb” to open it
  - The notebook now shows up as another tab
  - You can keep it in this view, or click on the tab to select it and then start dragging it around the screen to create a tiled view:
    - A blue shaded area appears that changes as you move the cursor around
    - To have a side by side view, drag the cursor until the blue shaded area shows a vertical rectangle on the right. Now the two notebooks are side by side
    - To revert to a single (non-tiled) view, click the notebook’s tab again and drag around until the blue shaded area fills the entire window (excluding the sidebar, if open)

Quitting Jupyter Lab

- To quit Jupyter Lab:
  - Close the Jupyter Lab tab in your browser, and
  - Shutdown the Jupyter Lab server
    - If you launched Jupyter Lab from Anaconda Navigator, close the Anaconda Navigator window (or select File ➔ Quit from the top menu in Anaconda Navigator)
    - If you launched Jupyter Lab from a console (Anaconda Prompt or Terminal), press Ctrl-C (or Cmd-C) at the command line to shutdown the server while keeping the console running, or simply close the console window