

# Python for Analytics

Course Introduction  
Syllabus, Software, Tutorials, etc.

# Topics

- Introductions
- Why this course?
- Developers vs Analysts
- Course Syllabus, etc.
- Software Tools

# Welcome

- Dr. Christopher L. Huntley
  - PhD in Systems Engineering (UVa, 1995)
  - At Fairfield U since 1997, before that (mostly) in industry
  - Worked in over a dozen programming languages since 1977
- Questions for you:
  - Who are you? (Name, nickname, and hometown)
  - Background? (Degrees and professional experience)
  - Something ***distinctive*** about yourself that we can't tell by looking at you?

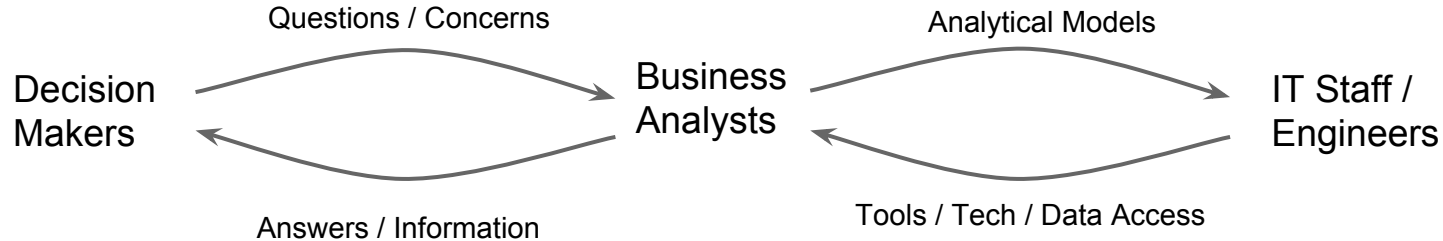
# Skipping Ahead ...

Please follow the [instructions](#) to download Anaconda ... It will likely take a long while to download and install. We'll fill the time with some basic concepts and a pep talk.

# The Big Picture

Python and Business Analytics

# The Role of the Business Analyst



# Computers: Fast but Dumb!

- Can perform millions (even billions or trillions or quadrillions) of calculations per second
  - Compute weather predictions
  - Search terabytes of Google indexes
  - Simulate a nuclear explosion
- However, they continue to struggle with things we do every day
  - Process and interpret visual input
  - Understand the meaning of human language/behavior
  - Determine mood by watching a person's face

# Why Learn to Program?

- Computers only do what we tell them to do
- Limited by our ability to communicate unambiguously
  - What we want the computer to do
  - How the computer is expected to do it
- Two implications for business analysts:
  - More likely to get what we want from computers if we ask the ***right questions in the right way***
  - ***Languages and tools continuously evolve*** as we ask for different things, so we should ***always be learning something new***

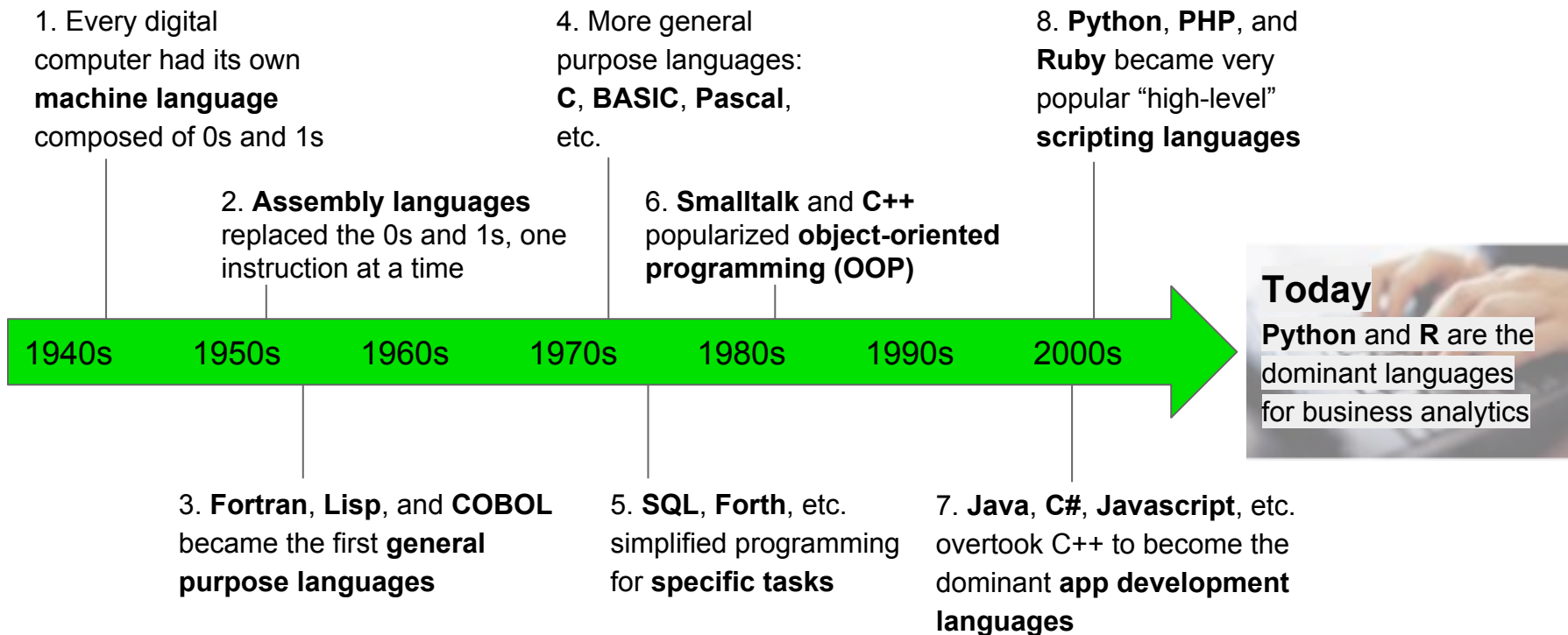


# So Why Python?

- Easy to learn
  - Simple syntax, without a lot of clutter
  - [Most things have] “One obvious way to do it”
- Encourages logical thinking
  - Block structure, data structures, etc.
  - “Coding at the speed of thought”
- Amazing industry support
  - Lots and lots of free/open source tools and libraries
  - Easy to “glue together” whatever you need

# A Brief History Lesson

Note: Until #8 all programmers were engineers or scientists



# Key Observation

Until only a few years ago, professional-level data analytics required mastery of a whole stack of programming languages and systems.

**Now, however, one can do pretty much anything with just Python, which has the best tooling around for analytics!**

Excel is fine, but a Jupyter Notebook with live data is much better.

# Course Expectations

What does success look like?

# Why are you here again?

The goal of this course is not to turn you into a systems engineer, but **to provide fundamental programming skills needed to design and build data-driven analytical models.**



# From the Course Catalog

“In this course, we **introduce** Python as a language and tool for **collecting, preprocessing, and visualizing** data for business analytics ... **[with an emphasis on ...]** business modeling, optimization, and statistical analysis.”

# Course Objectives

Objectives for each assignment will always be cover both **theory** and **practice**

**Theory:** You should understand ...

- Fundamentals of logical reasoning as used in contemporary programming languages
- Basic and advanced data structures needed organize, store, and integrate data
- Syntax, control structures, data types, etc. in the Python language
- Importance of Python as a tool for data science

# Course Objectives (Continued)

**Practice:** You should be able to ...

- Write Python scripts of moderate length and complexity
- Design and use data structures using built-in and third-party data types
- Integrate and use Python libraries like Numpy, Pandas, etc. to collect, process, and display data
- Master professional tools (Anaconda, Jupyter Notebooks, GitHub, etc.) used in the field



# About the Workload ...

Things progress pretty quickly in this course (about 2x an UG class) and **each new thing builds on the last.**

- **Coming to class is necessary (and much appreciated) but not sufficient to pass.**
  - Lectures are intended to cover the highlights, not substitute for readings and hands-on work. Even ungraded work has a purpose!
- **Put in the time and don't let yourself get behind.**
  - Expect to do lots of work at home, preferably at least an hour **every day**. It takes time and repetition for things to sink in.

# A Slight Digression about Values

**Be sure you know why you are here.**

Graduate School is a social contract, with a professional code of conduct that values ...

- Integrity over Grades
- Courage over Ego
- Community over Competition

**If you can't abide by our values then please take another course.**

# ... and Professionalism

**What does it mean to be a professional?**

- Get paid. Duh.

**So why do they pay you?**

- To get the job done right!

**What does it take to do things right?**

- Knowledge + Skills + Code of conduct

**Being a professional is as much how you approach things as it is getting the right answers.**

# Course Plans and Policies

Assignments, Grading, etc.

# Assignments

- Programming Tutorials (ungraded but required)
  - Cover specific theory and practice needed for the graded assignments. ***Progress is tracked online.***
- Quizzes (50% of course grade)
  - 5 Quizzes, with lowest grade dropped from Quiz Avg
- Team Project (40% of grade)
  - 2-3 students per team
  - Assigned just after midway through the course
- Professionalism (10% of grade)
  - Participation and timely completion of assigned work

# Grading System: Curve *Everything*

Every graded assignment will be **scored** and then **normalized** using the following formula:

$$QP = 3.5 + \frac{1}{2} (x - \mu) / \sigma,$$

← Note: that is half the Z-score plus 3.5, which is an A-

where

- $x$  is the student's raw score for the assignment
- $\mu$  and  $\sigma$  are the class average and standard deviation for the assignment

**Letter grades are then 3.67+ → A, 3.34-3.66 → A-, ...**

# Academic Honesty

- Cheating will be dealt with swiftly in accordance with Fairfield University policy
  - Unless given **explicit permission** to collaborate, do not share your work with others
  - *Avoid even the appearance of cheating!*
- Each graded assignment will be accompanied by the following (signed) pledge:
  - *On my honor as a Fairfield University student, I have neither given nor received any unauthorized aid on this assignment/quiz/project.*

# Course Logistics

Website, Syllabus, Software



# Class Docs / Website

All lectures, programming assignments, etc. are available here:

[christopherhuntley.github.io/is505-docs](https://christopherhuntley.github.io/is505-docs)

The class syllabus is linked from the home page:

[christopherhuntley.github.io/is505-docs/Syllabus.html](https://christopherhuntley.github.io/is505-docs/Syllabus.html)

# Sign Up for DataCamp

- Data Camp is an online school for data analytics in Python, and R. We have a “class group” for IS505 where your progress can be tracked.
- Invitation emails will be sent to your `@student.fairfield.edu` address.
- If no email is found, then use the following link:  
DATA CAMP LINK HERE

# GitHub / GitHub Classroom

All class documents, assignments, and projects will be managed online using GitHub.

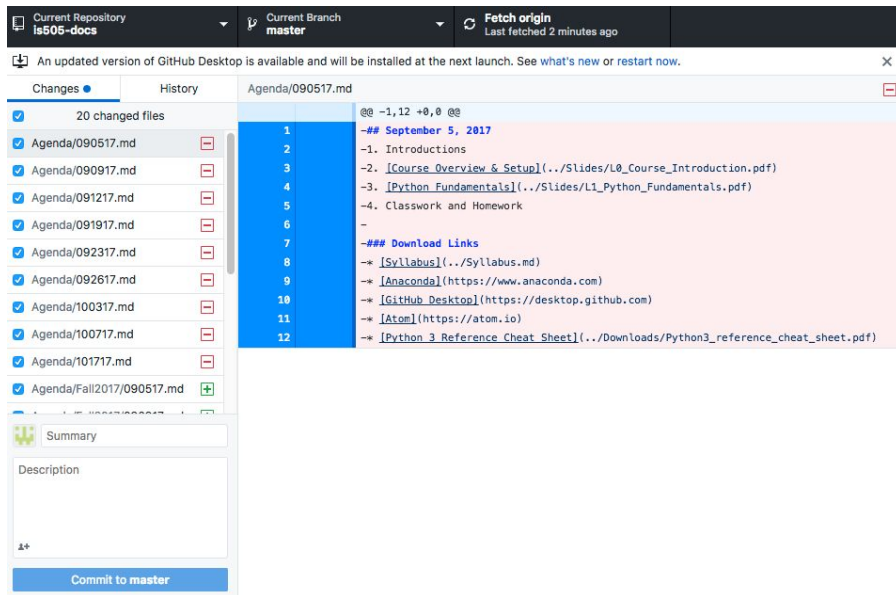
- Syllabus, lectures, etc. are in the is505-docs repo:
  - <https://github.com/christopherhuntley/is505-docs>
- GitHub Classroom will be used to post and submit programming assignments
  - Invitations for each assignment will be sent by email
- We will more about GitHub as we go along, starting with installation in class tonight

# Sign Up for GitHub

- Go to GitHub.com
- Sign up for a new account using your **@student.fairfield.edu** account.
- Send an email from your student email to [chuntley@fairfield.edu](mailto:chuntley@fairfield.edu) with your GitHub account username. The email subject is “GitHub account”.
- Keep an eye out for assignments from GitHub Classroom.

# Install GitHub Desktop

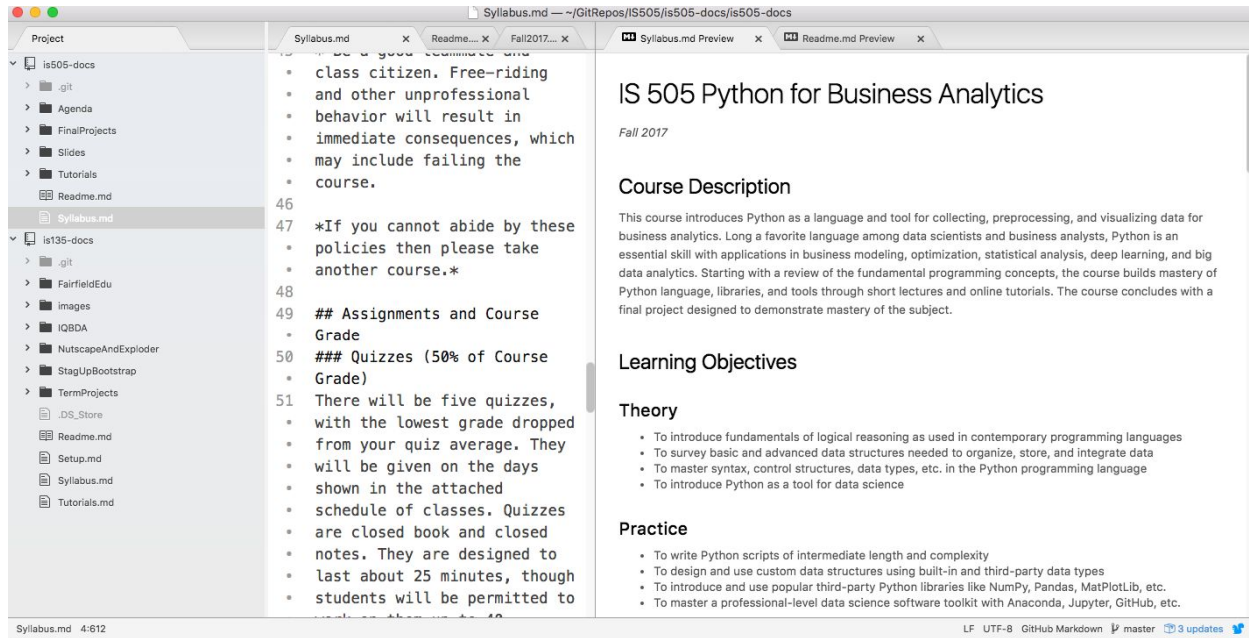
Download the **official version** from [desktop.github.com](https://desktop.github.com).  
Then install as usual.



# Install Atom (Recommended)

A code editor  
that works  
great with  
GitHub.

Install from  
[atom.io](https://atom.io)

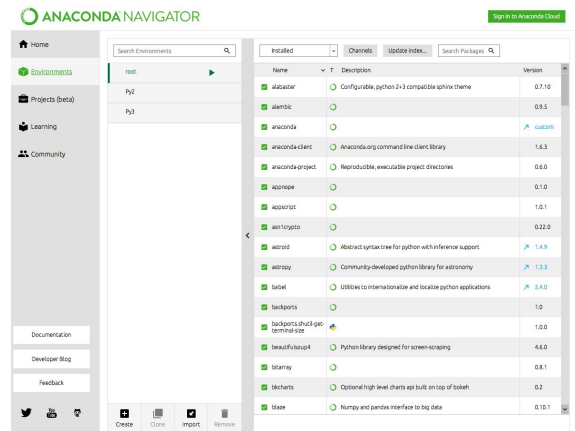


# Create a Folder for your work

1. Create a new folder called `IS505` in your `documents` folder or desktop.
2. All your Git repositories and other work should reside in this new `IS505` folder.
3. **Take note of where you created the folder. You will need it later.** No, being able to search for it doesn't count. Know where it is on your hard disk.

Anaconda is a desktop Python environment that bundles lots of tools and packages:

- Conda: command line tools





# Install Anaconda

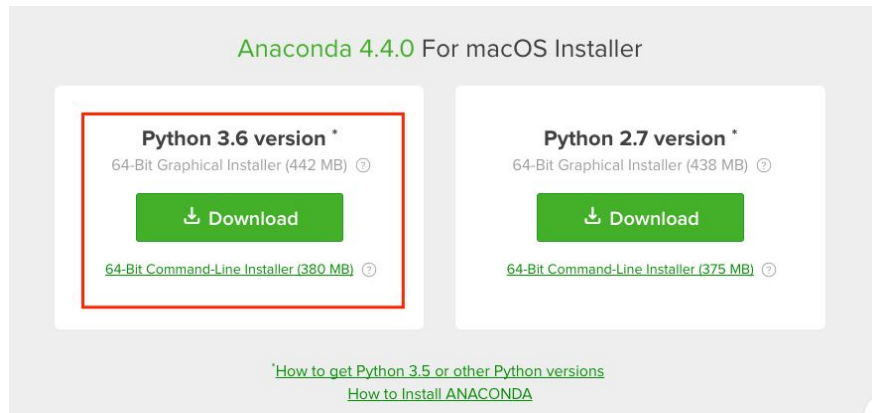
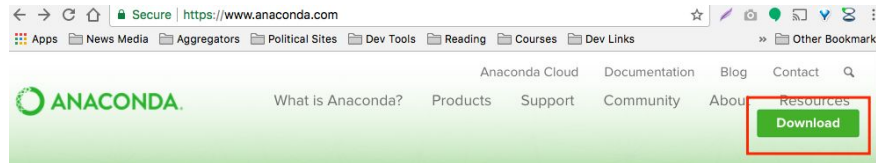
Go to [anaconda.com](https://anaconda.com) and click the download button.

Choose the Python 3.\* version for your OS.

Windows users: Use the 64-bit version unless you know you need 32-bit version.

The download may take a while. Be patient.

Install as usual.

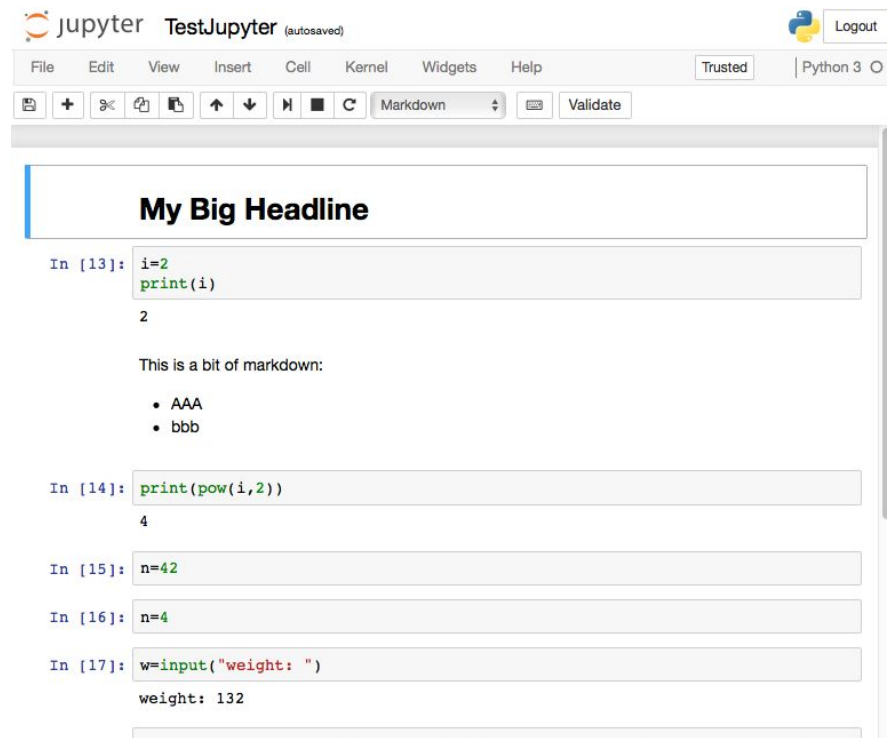


# Experiments with JupyterLab

Your first Notebook

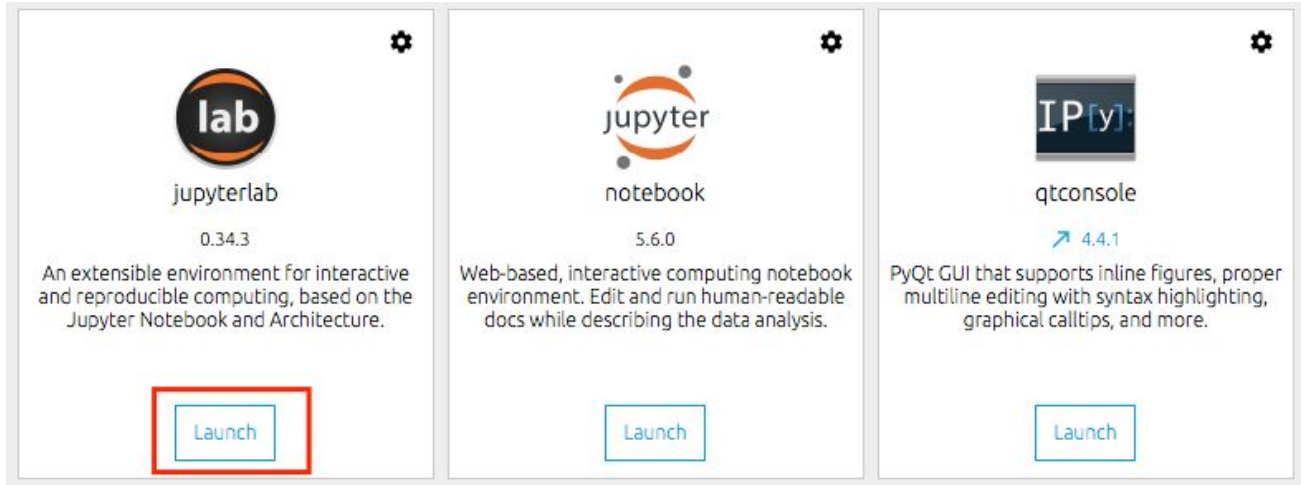
# What's a notebook do exactly?

- Mix formatted text and Python code to create interactive reports that can run in your web browser.
- GitHub can *render* your notebook pages automatically without any extra software.



# Create Your First Notebook (1)

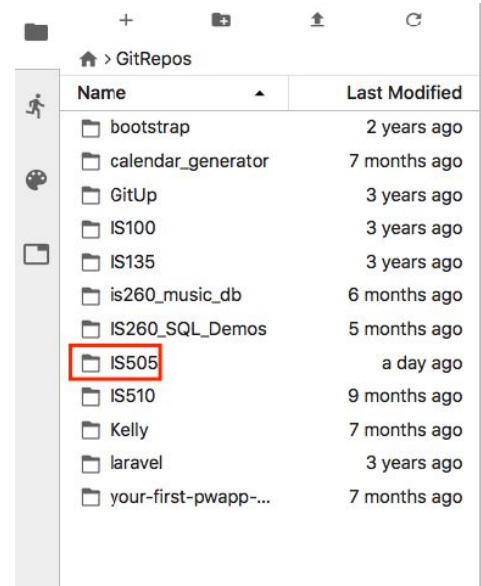
Open Anaconda-Navigator and launch JupyterLab from the home screen.



# Create Your First Notebook (2)

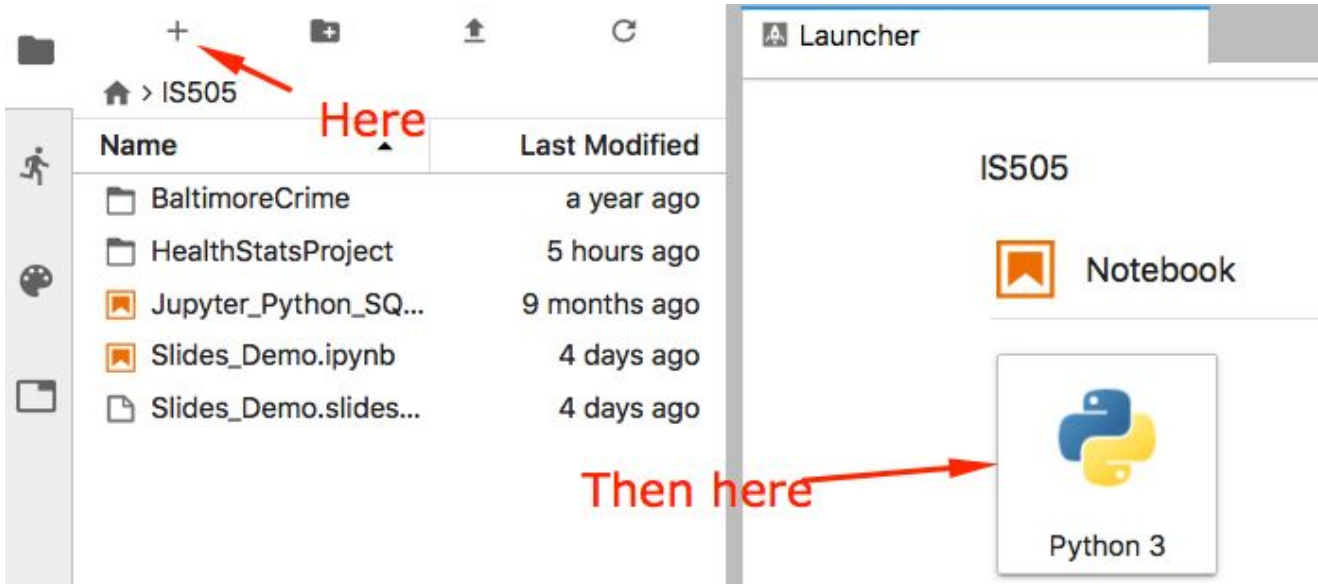
Navigate to and then open your **IS505** folder.

You may have to click on the folder icons rather than the folder names to open them. It's a weird JupyterLab bug that should get fixed soon-ish.

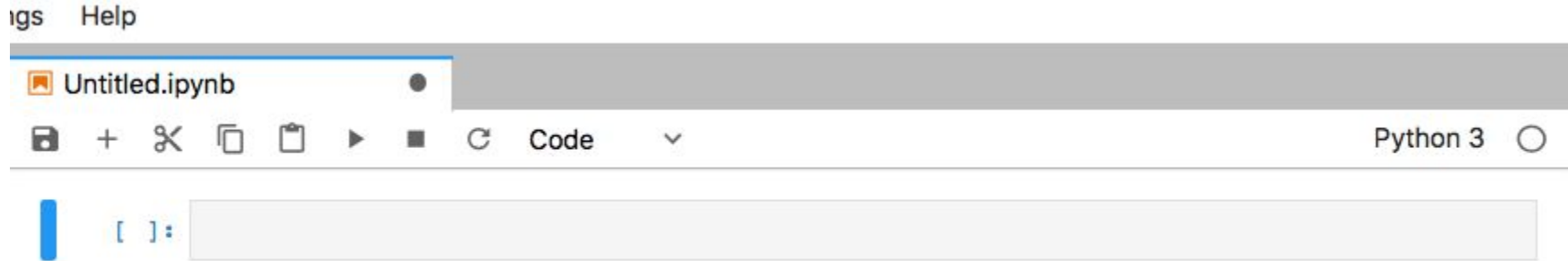


# Create Your First Notebook (3)

Create a blank Python 3 Notebook.



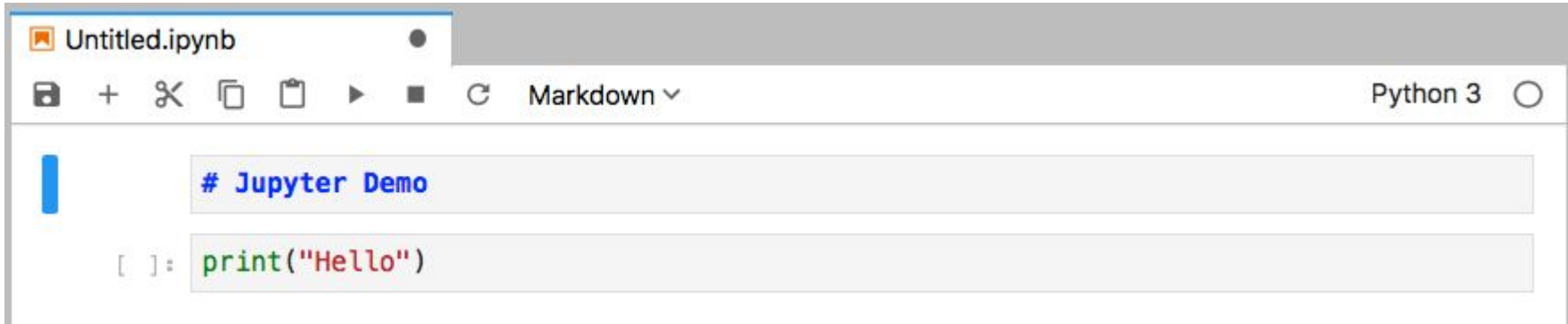
# Create Your First Notebook (4)



You will see a blank page with a box for entering text:

- Each *cell* (box) of the notebook is a snippet of formatted text or Python code.
- The toolbar at the top of the tab provides commands to add, move, format, etc. cells in the notebook.

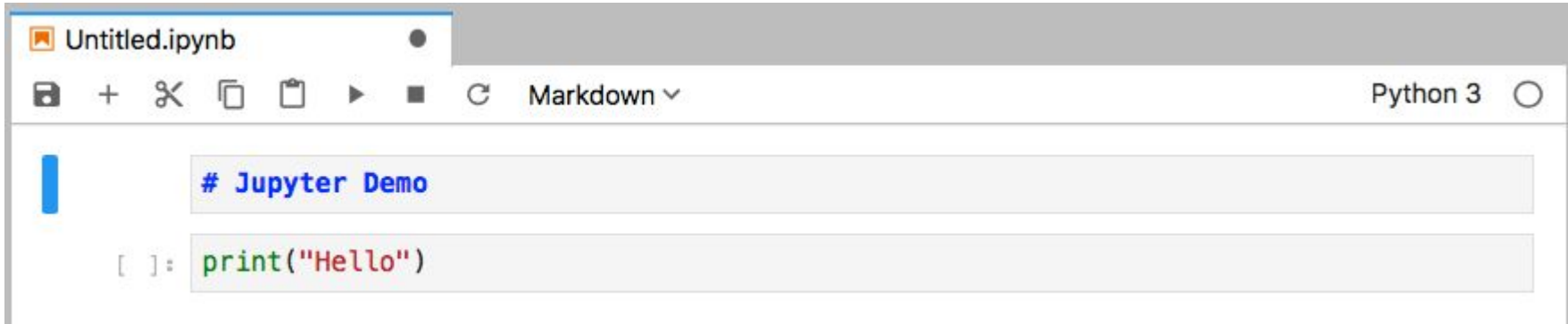
# Create Your First Notebook (5)



1. Enter `print("Hello")` into the first cell.
2. Press the `+` button to add a new cell.
3. Select `Markdown` from the cell type selector.
4. Enter `# Jupyter Demo` in the second line.
5. Drag the second line to the top of the notebook.



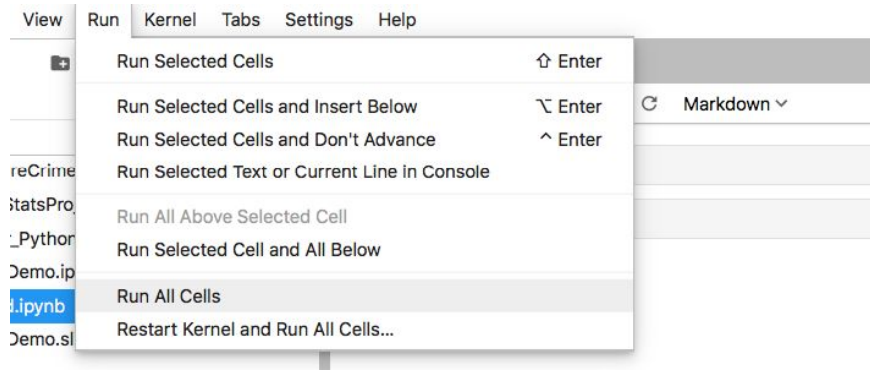
# Create Your First Notebook (6)



- The `# Jupyter Demo` snippet is written in a text formatting language called [Markdown](#), which is commonly used for wikis, blogs, and manuals.
- The `print("Hello")` snippet is Python 3 code that we want to run inside our notebook.

# Create Your First Notebook (7)

From the Run menu, Run All Cells in the notebook.



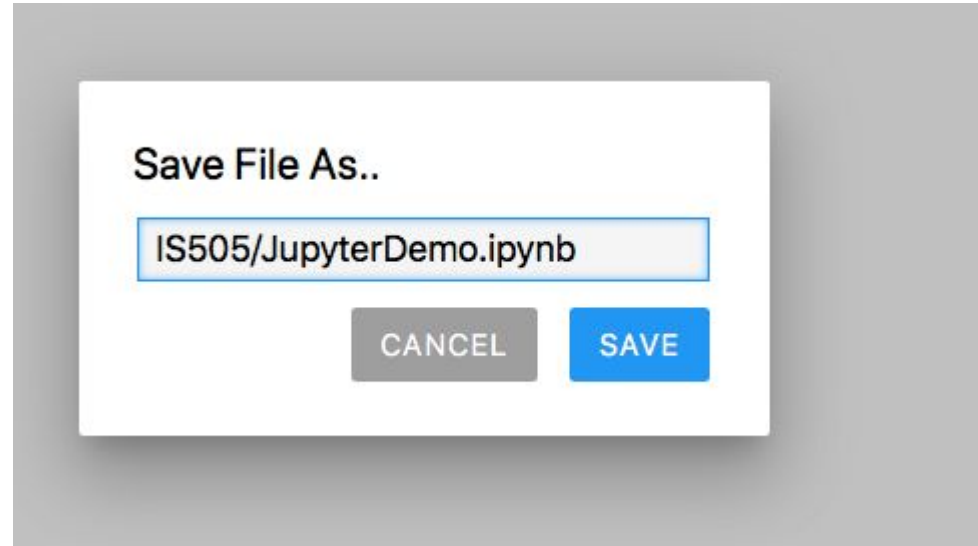
The notebook will **render** with formatted text and code output.



# Create Your First Notebook (8)

From the File menu,  
save the notebook as  
`JupyterDemo.ipynb`  
(with no spaces).

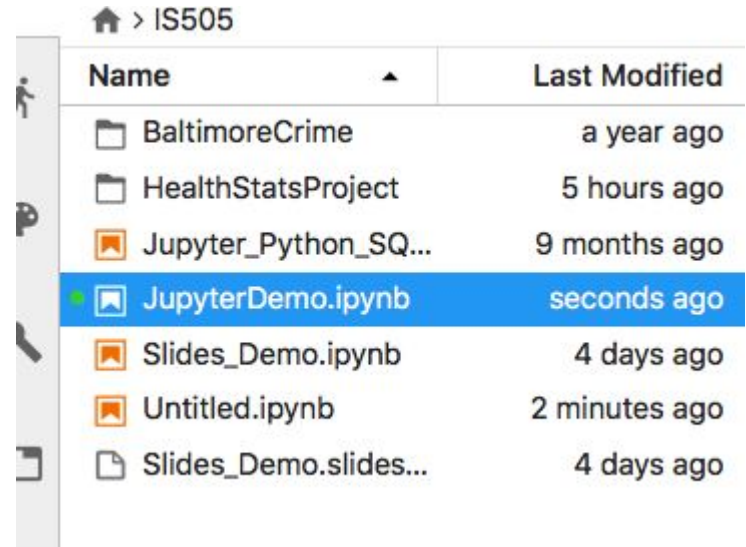
Then close the  
notebook.



# Create Your First Notebook (9)

You can reopen the notebook from the file explorer.

You can also hide the file explorer (to give more space to notebooks) by clicking the folder icon on the far left. Clicking again brings it back.



Name	Last Modified
BaltimoreCrime	a year ago
HealthStatsProject	5 hours ago
Jupyter_Python_SQ...	9 months ago
JupyterDemo.ipynb	seconds ago
Slides_Demo.ipynb	4 days ago
Untitled.ipynb	2 minutes ago
Slides_Demo.slides...	4 days ago

# Python for Analytics

Course Introduction  
Syllabus, Software, Tutorials, etc.