

# **Course: Intro to Python & R for Data Analysis**

## **Lecture: Introduction**

Professor: Mary Kaltenberg  
Fall 2020

contact: [mkaltenberg@pace.edu](mailto:mkaltenberg@pace.edu)

About me: [www.mkaltenberg.com](http://www.mkaltenberg.com)

Readme (aka syllabus):

## About the Course

- Learn basics of coding in Python and R
- Practical application of statistics and econometrics using Python and R
  - This course complements stats and metrics and is not a substitute. If you haven't taken these courses in the past, you may get lost.
- How to collect data online, clean it and use it.
- How to visualize data (and tell a story with graphs).

WARNING: I am not a computer scientist, which means the way I teach the course is not geared to understanding all of the nuances in python language. Rather, I will teach you practical applications of python and R for economics - my goal is to teach you what I wish I learned in class.

My goal as a professor and in this course is to ensure that we have a community of engaged learners. That includes me!

- If I make a mistake - tell me!
- If you have a constructive suggestion - tell me!
- If at any point you do not feel included OR you feel excluded - tell me!
- You can email me or you can report your suggestions and concerns anonymously (it goes directly and only to me) at anytime to this form: [here](#)

## Grading

- 5 Assignments (40%)
- 1 Presentation (10%)
- Final Project (40%)
- Participation (10%, includes class discussion, online forum and cook book contribution).

Late assignments will not be accepted (No exceptions), but given that life circumstances may come up, I will drop the lowest grade of your problem sets from the average of your problem set portion of your grade. Use this policy wisely. In other words, you have 5 assignments, and only 4 will be included in the assignment portion of your grade.

This is a project based course. The goal is that you will accumulate skills so that you learn how to code so that you can:

- scrape data from the web
- clean that data
- analyze that data

You'll do all of those things and report your final project to the class in a final presentation.

## Why Python AND R?

These 2 languages are the languages of data science, and quickly becoming the language of applied economics.

These 2 languages are complementary; IMO if you know python, you will quickly know R.

I use Python and R for different things, so there is some path dependency.

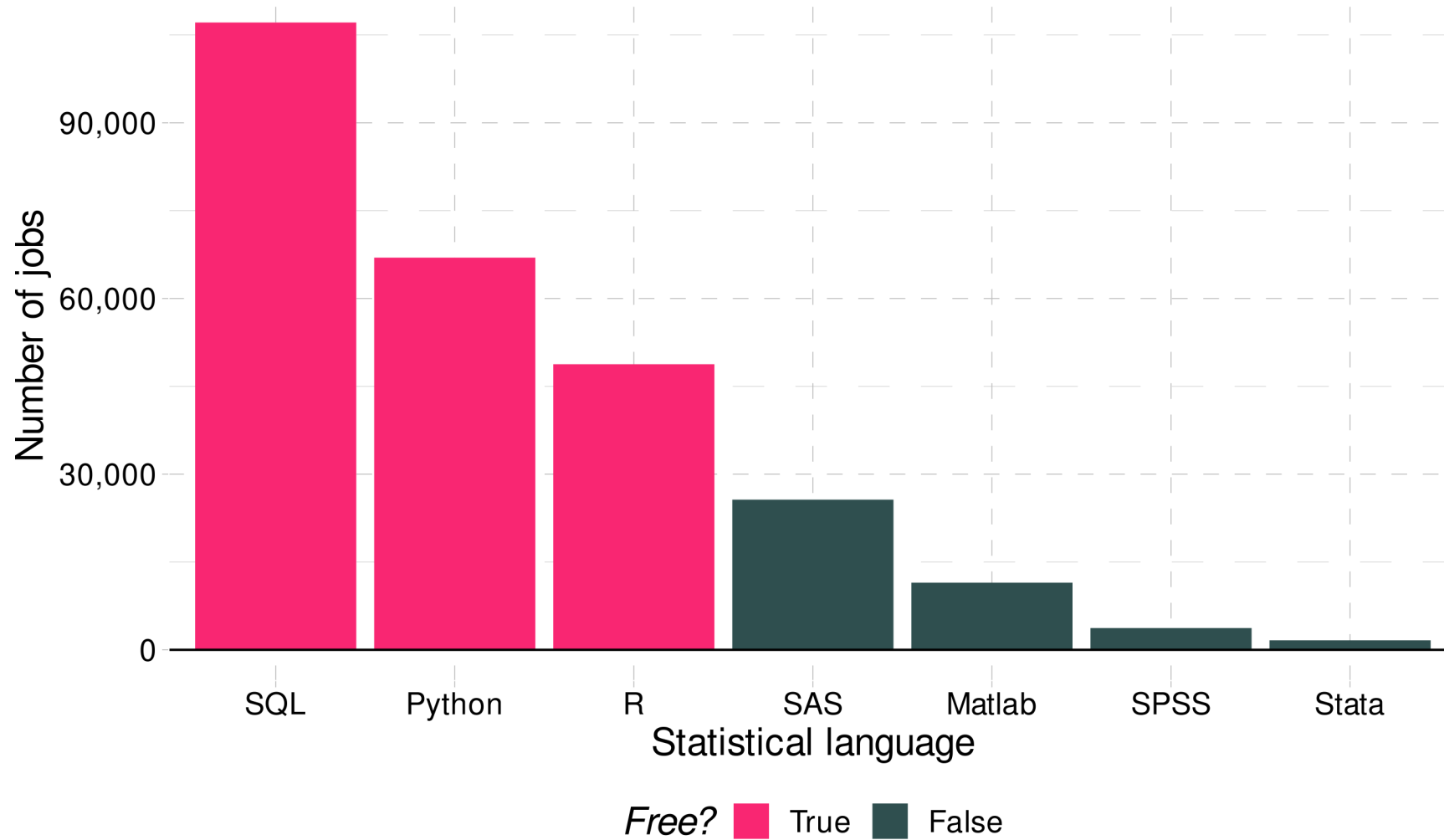
- Python for data wrangling, webscraping and visualization (and Machine Learning if you're into that)

- R for data analysis and visualization

I think it's possible to learn the basics of both, but this is the first time I'm teaching them together. Please be patient with me! If it doesn't work, we'll focus on just Python.

# Comparing statistical languages

Number of job postings on Indeed.com, 2019/01/06



Graph by [Grant McDermott](#)

**Python** is a versatile language and does much more than analysis. You can automate tasks, build software programs, apply artificial intelligence and machine learning techniques, program - and a ton of other things. It's a great code to learn because it's in high demand in the labor market for a broad range of industries.

**R** is a bit more limited, but is excellent for data analysis for a variety of fields. A broad range of industries use it, as well. Everything I teach today can be done in R, as well. Python's weakness at the moment is regressions, while R has a breadth of packages - and often the latest econometrics technique is already up on R.

The first half of the class will focus on basic coding principles in Python - we'll move into data wrangling, web scraping and visualization. The latter half of the class will be in R and focus on visualization and regression analysis.

## Objectives for Lecture 1

What you should have *attempted* to do before today's class is to:

- READ the syllabus
- Download anaconda (where python, jupyter notebook and R is installed)

What we will do today together:

- Define what are your objectives for the course
- Trouble shoot software installation
- Start a new Jupyter notebook
- Integrate R in jupyter notebook
- Learn about cells and markdown
- Learn how to read documentation (and find other helpful sources to troubleshoot)

## Installation

### "shell" or "terminal" Installation

You'll need a "shell" to install packages and it's an easy way to open jupyter notebook.

**ALSO** it's necessary if you work with servers.

- **Mac and Linux** users have terminal already installed.
- **Windows** users must download and install an additional program to get a shell or powershell (sometimes it's also called a Bash). This [video](#) is particularly useful. Installation depends on which Windows OS you currently have - so, you may have to google around for appropriate installation guides..

Other OS specific installations to do now as it will save you time in the future:

- **Windows:** Install [Rtools](#).
- **Mac:** Install [Homebrew](#).
- **Linux:** None (you should be good to go).



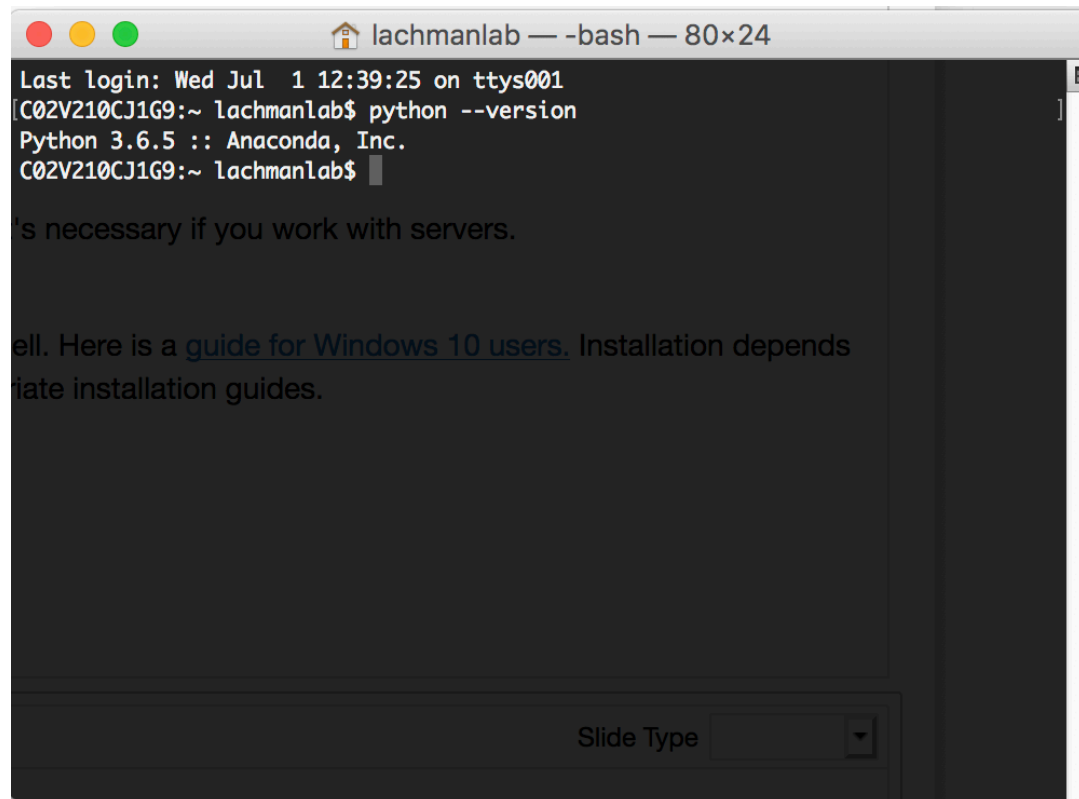
## Checking your Python Version

We'll need to check a few things and update python if necessary:

First, check the python version you have currently installed. Open up your terminal/shell and type:

`python --version`

It'll look like this:

A terminal window titled 'lachmanlab — -bash — 80x24' with standard macOS window controls. The terminal shows the command 'python --version' being executed, resulting in the output 'Python 3.6.5 :: Anaconda, Inc.'. Below the terminal output, there is additional text on the slide: 's necessary if you work with servers.' and 'ell. Here is a [guide for Windows 10 users](#). Installation depends iate installation guides.' At the bottom right of the slide, there is a 'Slide Type' dropdown menu.

```
lachmanlab — -bash — 80x24
Last login: Wed Jul 1 12:39:25 on ttys001
[C02V210CJ1G9:~ lachmanlab$ python --version
Python 3.6.5 :: Anaconda, Inc.
C02V210CJ1G9:~ lachmanlab$
```

s necessary if you work with servers.

ell. Here is a [guide for Windows 10 users](#). Installation depends iate installation guides.

Slide Type

You'll want to make sure your version is *at least* version 3 (Python 3.X.X)

This should be the case, if you installed the correct version of anaconda. If not, you'll need to install python version 3 - it might be easier to uninstall anaconda and reinstall the correct version of Python 3.8 [here](#)

Read the system requirements and troubleshoot guide if you're having issues.

## Pip and Packages

To install **packages** for python, you'll use **pip**

Packages are basically pre-programmed directories of code - both python and R uses packages.

For now, you can image that packages are kind of like an application on a phone. Sometimes you'll have to update your application for them to work - and they are often updated to meet other dependent package updates.

Pip is like Google Play or the App Store on your phone. Pip manages packages where you can install, update, and uninstall packages.

There is a huge community beyond the developers of Python that contribute packages - and we have a lot to thank them for! It's part of the open source community.

Let's make sure pip is up to date by inputting this code in the terminal/shell:

```
python -m pip install --upgrade pip
```

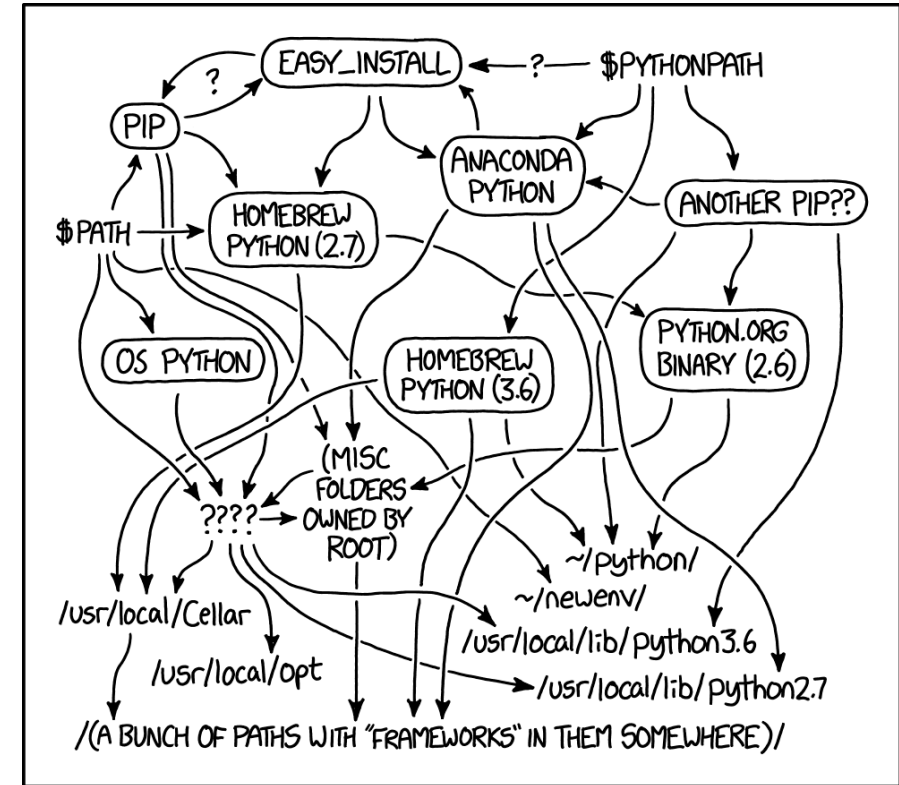
## CRAN and Packages

R also has a package manager called CRAN. We'll talk more about it when get to the R section of the course. For now, you need to install it so we can ensure the R environment is working in your notebook.

If you haven't already, be sure to download R [here](#)

I'll be honest with you, package management of python can get complicated. It's rare to have this issue, but I have spent a couple hours to resolve an issue in the past.

So, it's important that you know how to install and uninstall packages if an error occurs. Usually, an install and reinstall will solve the problem.



## Installing R environment for Jupyter Notebooks

What's nice about jupyter notebook is that you can run notebooks in different languages. Since this class will teach you both Python and R, we will use relevant "notebooks" of code when using a particular language. I'll come back to this later today when we talk about jupyter notebooks.

For now, let's just install everything we need so we don't have to deal with it later.

We'll want to get R running in Jupyter notebook.

The following code will be run in your terminal/shell.

First, let's update our notebooks to make sure they are in order:

```
conda update jupyter notebook
```

Conda is Anaconda, which you downloaded earlier.

Install IRKernel package. It's a package that will allow you to connect R to jupyter notebook:

```
conda install -c r r-irkernel
```

When asked: *Proceed ([y]/n)?*

Respond with y

# Jupyter Notebooks

Jupyter notebooks are a great way to organize and document code.

To Launch a Jupyter Notebook:

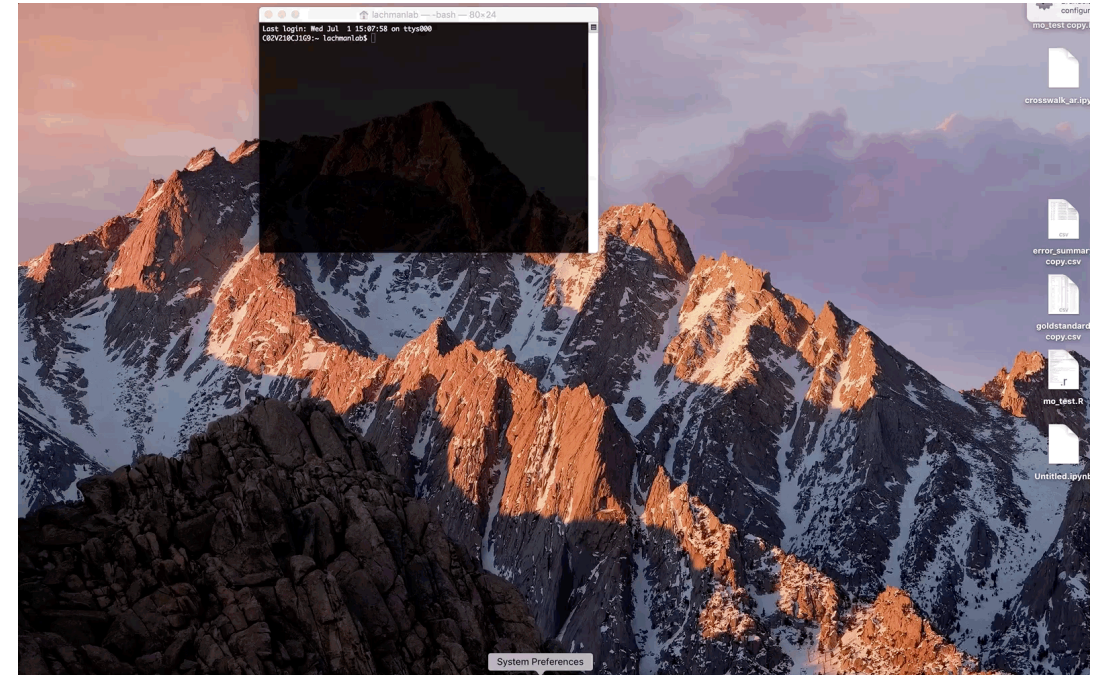
**\*\* Mac\*\*:** go to your terminal and type:  
jupyter notebook

**Windows:** go to your control prompt (search for cmd) and type: jupyter notebook

This will launch a localhost, where you should see a directory of your files/folders.

Directories are the backbone of how your computer organizes files. You'll need to get comfortable with this structure partly because it will help you organize your own files neatly for data projects.

This is what happens when you launch a notebook ->



If you are unable to install these programs for a variety of reasons, you can use jupyter-like notebooks from Colab (run by google).

- They provide free computing power, too!
- They can be stored on your google drive and shared with others.
- Go [here to start a new project for python](#)
- Go [here to start a new project for R](#)
- Go [here is another option to use install-free R through Kaggle Rstan](#)

However, I won't be able to help troubleshoot - I don't know all of their limitations, but it is a free and install-free way to use notebooks.

Jupyter notebook runs locally from whatever is your preferred web browser. I personally use Mozilla Firefox, but jupyter notebook is compatible with any web browser.

Note: This is particularly useful if you work with servers - you can locally work on your notebook environment, but run your code on a sever.

After jupyter notebook launches, it takes you to your current directory. If you launch a new notbeook, it will be stored in that folder, but you can navigate and create folders as you wish from the director tree.

On the top right you click "New" and that should give you options to click python3 or R. Notice that if you installed the R environment properly, the R option should show up.

If you click on the title, you can change the name of the workbook.

Note on directory file structure: Whenever you have a new data project, I suggest that you create a new folder where you can store all of your files. We'll talk more about this when we get to the data section.



## Using Notebooks

**cells** is where you put input (code) and it will print out results below the cell (example below). They are container for code or text.

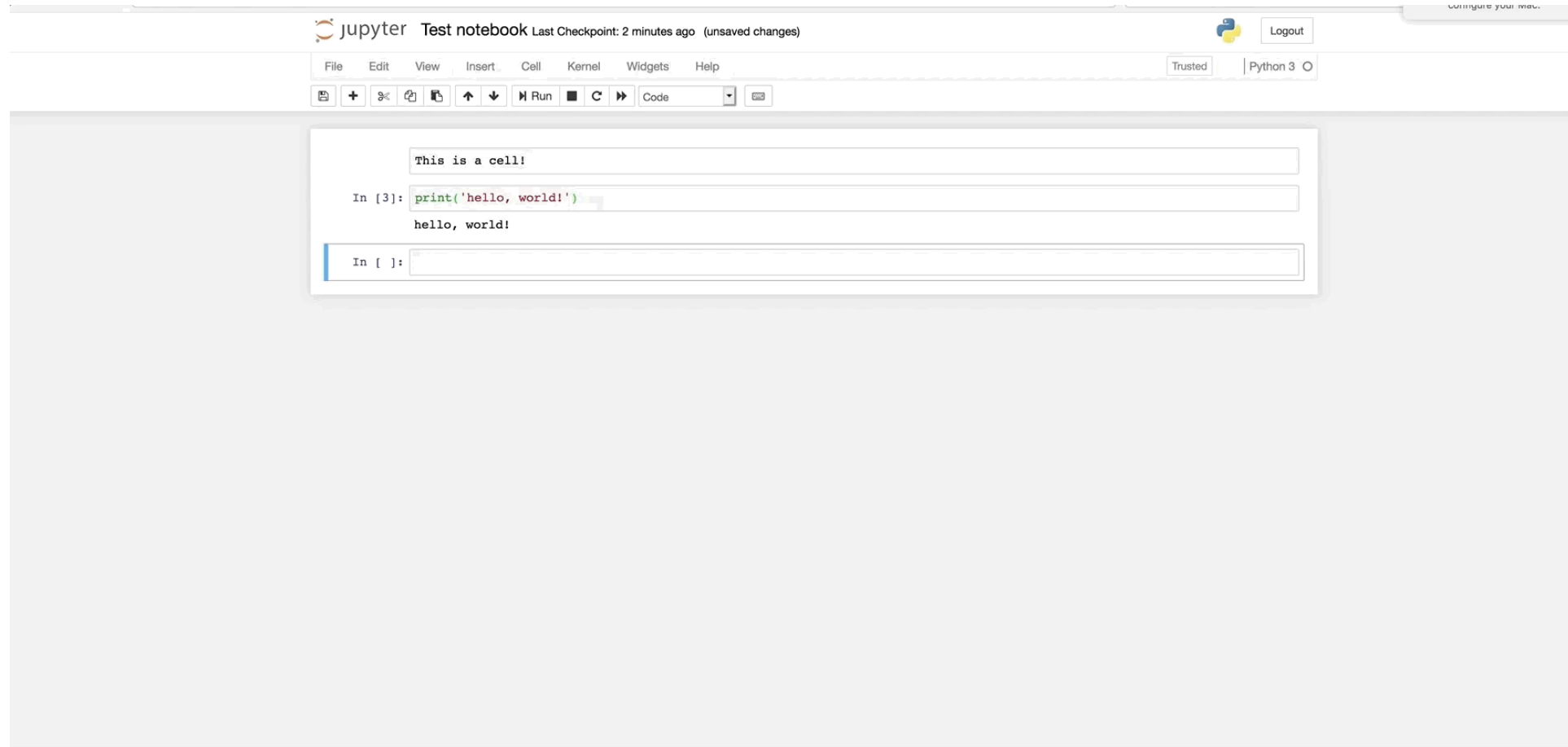
You can change the language a **cell** uses - which is super useful to code and document at the same time.

"Markdown" makes writing on the web easy - it's an easy to format language that can embed text, html, latex easily. Here is a quick cheat sheet for Markdown to make your documentation easier: [here](#)

"Code" is the code of which your environment is set up. When you opened a new notebook and could click Python3, then the code of choice is Python3.

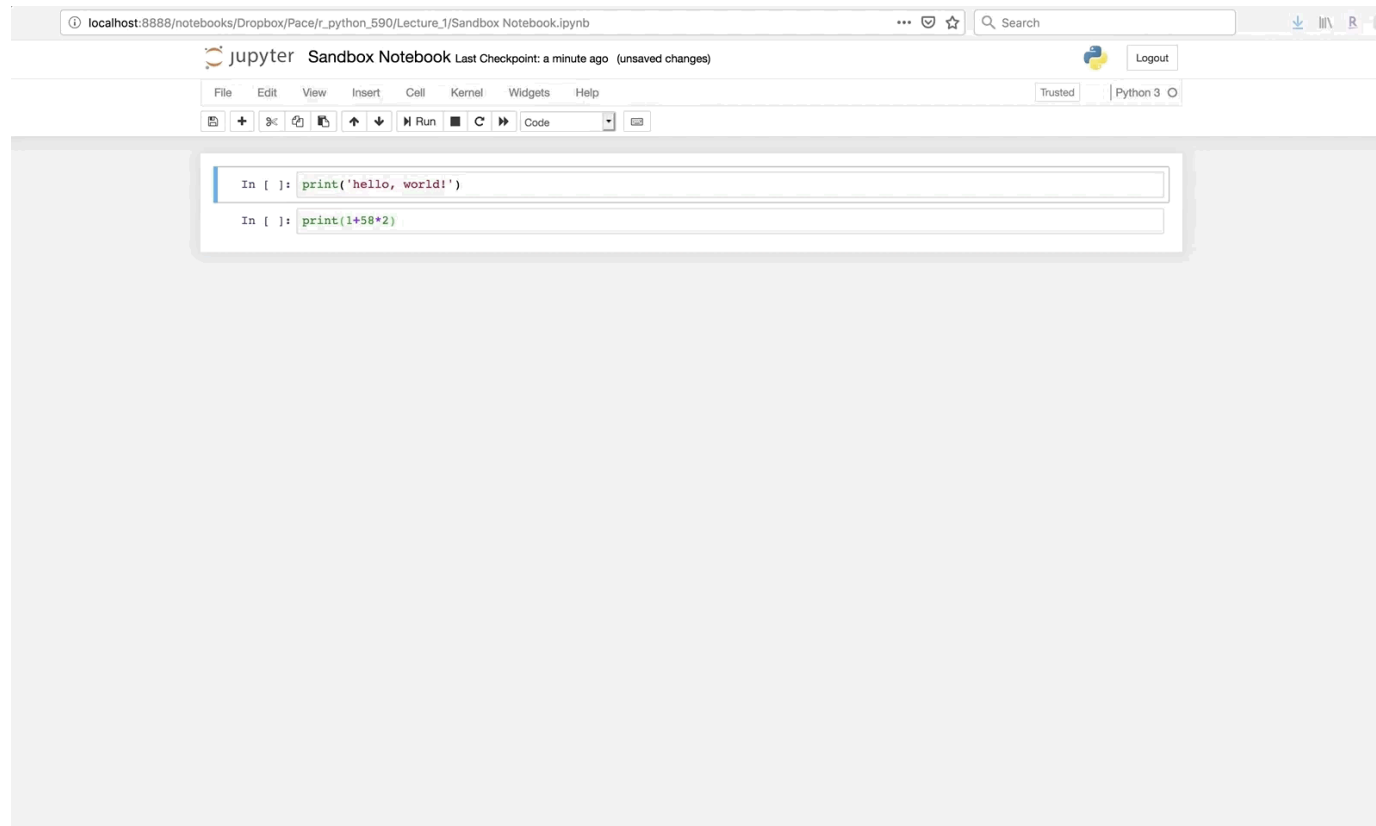
Here is how you can change cell environments. Each cell can be either markdown or code (never both).

# Opening a Notebook



**kernels** are the "computational engine" which executes code blocks of the notebook. Sometimes when there is an error or super long run-time, you may need to "restart the kernel"

Here's how you can do that:



## How to read Documentation

The great thing about open source is that everything is WELL Documented.

Which is a nice reminder: **DOCUMENT EVERYTHING YOU DO**

A few reasons:

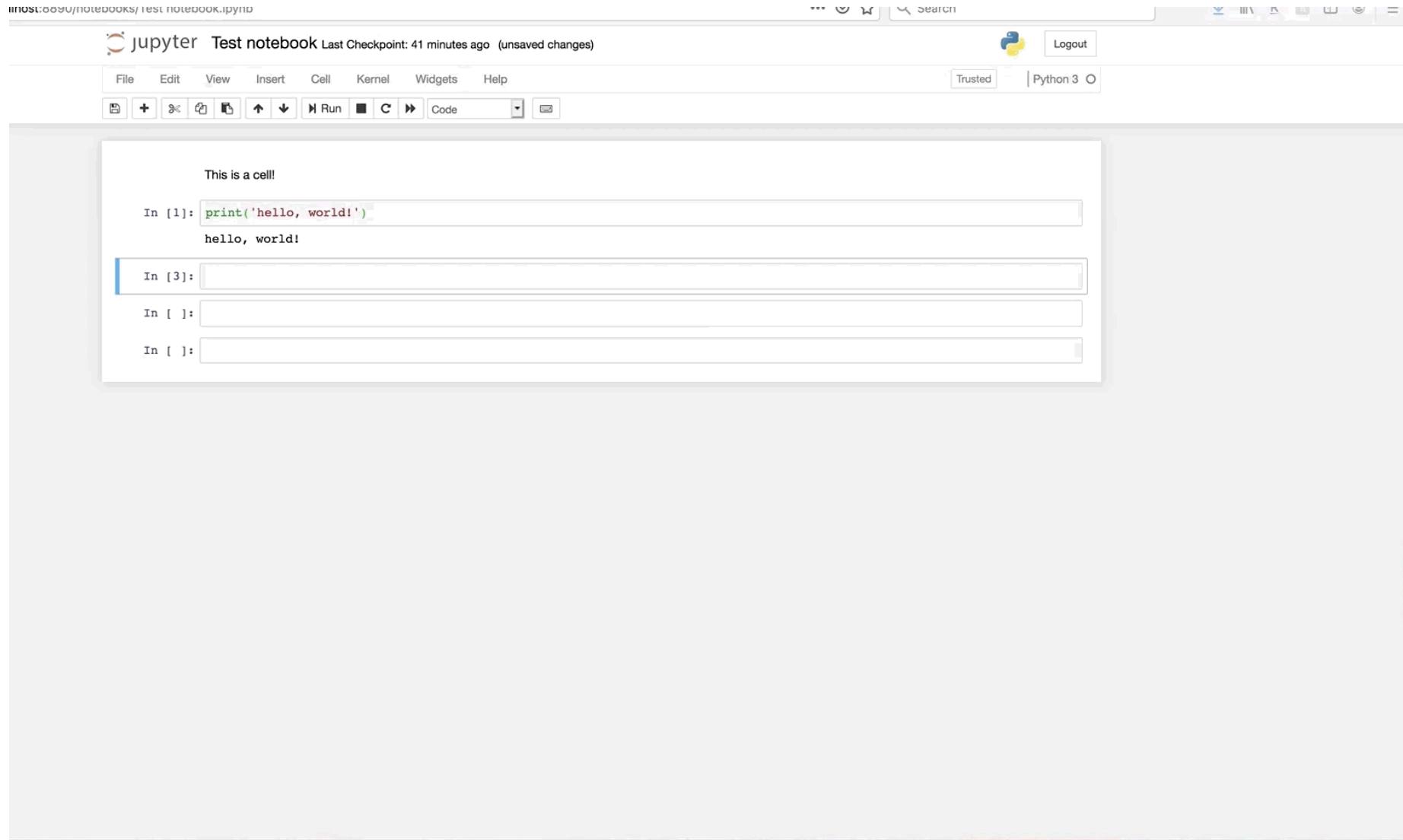
- helps you remember where you left off
- helps other people know how you did you work
- helps replicate your work for yourself and everyone else

Jupyter notebook has a built in feature that you can call the help feature of any code. All you have to do is add a question mark at the end of the command.

`print?`

The documentation will pop up at the bottom. This is particularly useful when you forget key arguments that are needed for the command.

# Integrated Help Function



Python also has detailed documentation on their [website](#). In general, they have great resources for beginners [here](#).

Each package also has their own documentation.

Let's go take a look at an example on the print function [here](#)

### Print Is A Function

The `print` statement has been replaced with a `print()` function, with keyword arguments to replace most of the special syntax of the old `print` statement ([PEP 3105](#)). Examples:

```
Old: print "The answer is", 2*2
New: print("The answer is", 2*2)

Old: print x,           # Trailing comma suppresses newline
New: print(x, end=" ")  # Appends a space instead of a newline

Old: print              # Prints a newline
New: print()            # You must call the function!

Old: print >>sys.stderr, "fatal error"
New: print("fatal error", file=sys.stderr)

Old: print (x, y)        # prints repr((x, y))
New: print(x, y)         # Not the same as print(x, y)!
```

You can also customize the separator between items, e.g.:

```
print("There are <", 2**32, "> possibilities!", sep="")
```

which produces:

```
There are <4294967296> possibilities!
```

Note:

- The `print()` function doesn't support the "softspace" feature of the old `print` statement. For example, in Python 2.x, `print "A\n", "B"` would write `"A\nB\n"`; but in Python 3.0, `print("A\n", "B")` writes `"A\n B\n"`.
- Initially, you'll be finding yourself typing the old `print x` a lot in interactive mode. Time to retrain your fingers to type `print(x)` instead!
- When using the 2to3 source-to-source conversion tool, all `print` statements are automatically converted to `print()` function calls, so this is mostly a non-issue for larger projects.

The documentation includes:

- details of changes that were made between versions
- what the function does
- examples of implementation.

Let's take a look at the package **numpy** and their documentation for calculating the standard deviation. [Click here](#)

When you see codeblocks, anything with ">>>" are lines that indicate that they were inputted. Lines that do not include that are outputs.

### Examples

---

```
>>> a = np.array([[1, 2], [3, 4]])
>>> np.std(a)
1.1180339887498949 # may vary
>>> np.std(a, axis=0)
array([1.,  1.])
>>> np.std(a, axis=1)
array([0.5,  0.5])
```

>>>

Hashtags "comment out" text. When you run a line of code you can block out text that you may want to include as a note.

I often use #for documentation and type out a sentence about what I am doing in the cell block to remind myself.

For example:

```
print('hello, world!') #This is how you print strings
```

```
print('hello, world!') #This is how you print strings
```

```
# you can "comment out" entire lines with keystrokes: command  
+forward slash for mac (shift + forward slash for windows)
```



You may not know how to execute something that you want to do or get a puzzling error. Most likely, someone out there had the same issue or error. A great resource to find help on how to resolve a problem that you might have is to search on [stackoverflow.com](https://stackoverflow.com). You'll want to know how to read code blocks on this site (and similar ones).

A *BIG* part of coding is practicing. You should sign up for [datacamp.com](https://datacamp.com). It's free, but not all of the content is free. However, the content that will help you jump start coding is free. There are videos and most importantly, coding examples and practices that you can work on - I *HIGHLY* recommend doing the practices

# Let's Start Coding!