

Python Foundations

Nichole Bennett

Course Information

About Me

Used to research climate change impacts on butterflies (ecology, evolution, behavior, some genomics)

Then, I taught adults data science with General Assembly and taught kids coding through various programs.

Now I research science communication.

In my spare time, I do improvisational acting.



About You

Find a partner, find out the following about them:

- Name
- Where they are from
- What they do for work, for fun
- Why they are here
- One fun fact

Then, we'll share.

Course Overview (subject to change)

Day 1: writing/running Python code, use Python for basic data analysis tasks (cleaning, reformatting, exploration, analysis), understand what Python is useful for

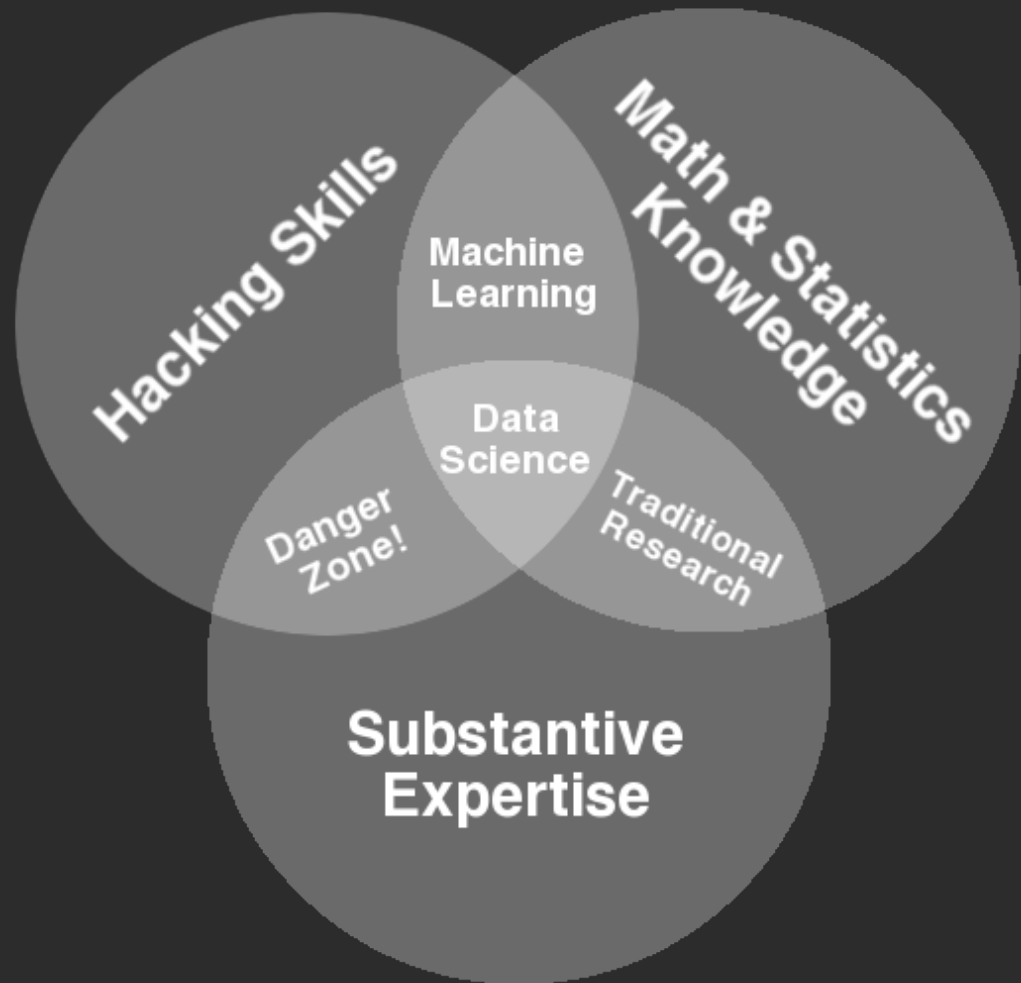
Day 2: collect data from CSVs, explore large datasets, clean and munge data to prepare for analysis, apply machine learning algorithms to gain insight from data, visualize results

Anything missing on here that you were hoping for?

Prerequisites

This should be pretty beginner-friendly.

Feel free to speak out if you don't understand something--I'll adjust.



Install Help Time!

Mac/Linux Users: Go to your Terminal and type `conda list`

PC Users: Go to your Anaconda Prompt and type `conda list`



Questions About Course In General?

Any Requests?

No matter where you are now, you'll be fine.

*You have already proven that you are a
self-starter and self-learner,
and that will take you far.*

What is Python?

What is Python?

Created by Guido Van Rossum in 1991

Easy language to learn

Lots of contributors to modules

Emphasizes human readability



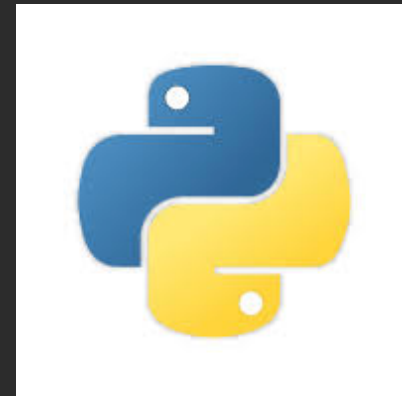
What is an interpreted language?

Python is an interpreted language

Code doesn't have to be translated into machine-language before execution (VS compiled code that is executed by the computer's CPU)

This means it's not as fast as compiled code

But also means it's more portable



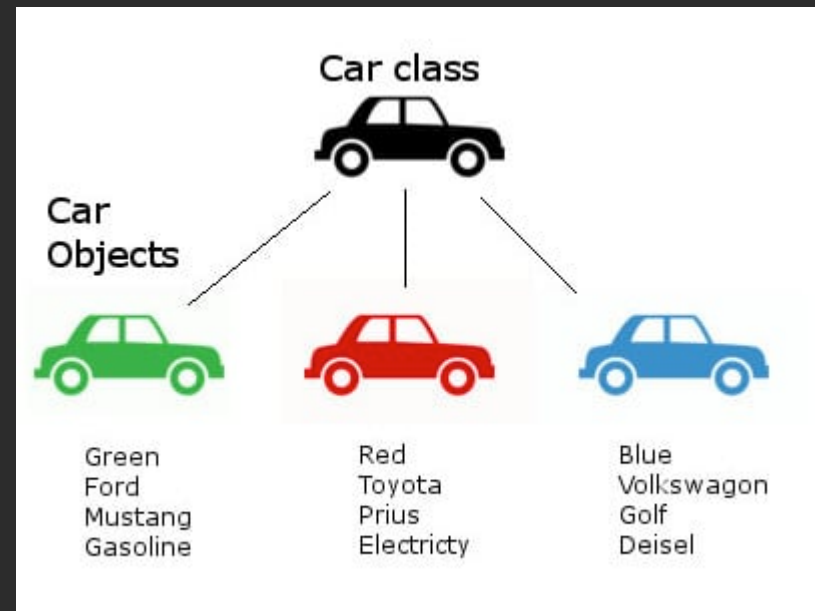
What is an object-oriented language?

Python is an Object-Oriented language

Instead of just being a series of actions, Python lets us form "objects" that have their own data (attributes) and their own actions (methods) to modify their data.

A benefit of OO is that it is easier to reuse the code in other programs.

We'll be using lots of objects. (We may not even realize it).



What is a “high-level” language?

Python is a high-level language

```
1 | print("hello world")
```

Python

```
1 | #include<iostream>
2 | using namespace std;
3 |
4 | int main()
5 | {
6 |     cout << "Hello World!";
7 |     return 0;
8 | }
```

C++

What is a dynamically-typed language?

Python is a dynamically-typed language

This means you don't have to specify what type the variables are. The computer figures it out for you



Why use Python?

Why use Python?

Quick and easy to use

Big community with lots of "batteries included" libraries (especially for machine learning)

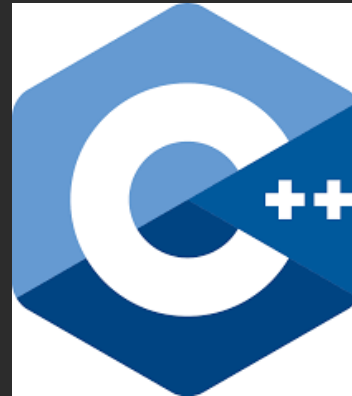
Most libraries written by people with a computer science background, so easy to switch from library to library



Why not use Python?

Why not use Python?

If you have something that needs speed



If you are doing lots of stats



Python Interactive Shells

Write code that is executed immediately by the Python interpreter.

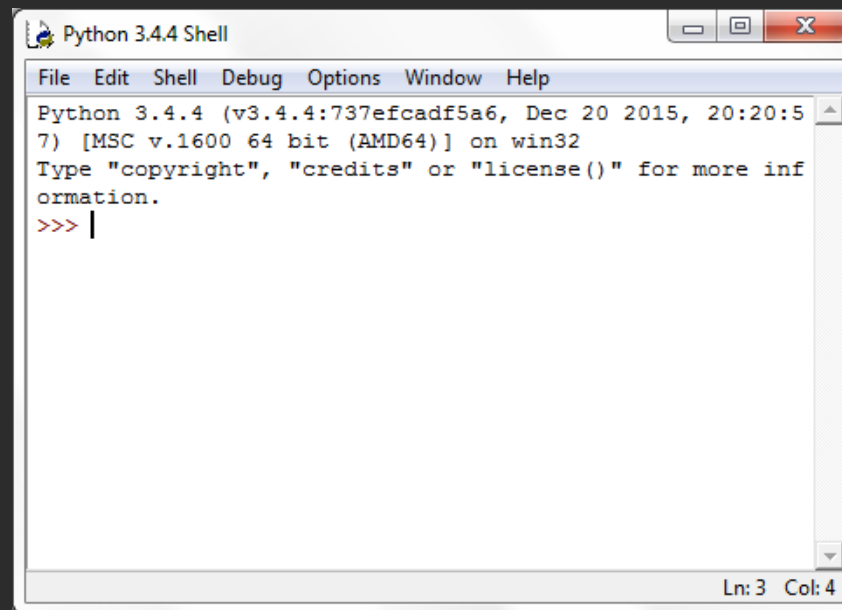
We are able to "interact" with the results of the commands we pass. We can do this using a:

- Python shell
- iPython shell
- Jupyter notebook

Python Shell

A python shell is similar to a Command Line Terminal and it can be launched by typing (into Terminal or Git Bash)

python

A screenshot of a Windows-style application window titled "Python 3.4.4 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following text: "Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600 64 bit (AMD64)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", and a red prompt ">>>" followed by a vertical cursor. The status bar at the bottom right shows "Ln: 3 Col: 4".

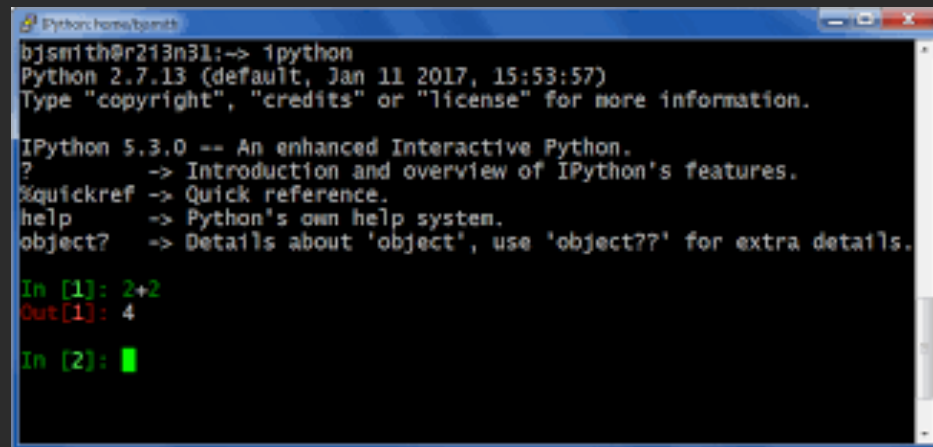
```
Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
Ln: 3 Col: 4
```

iPython Shell

iPython provides syntax coloring and shortcuts to interact with other code.

You can launch it by typing (into Terminal or Git Bash)

`ipython`

A screenshot of a terminal window titled "Python home/basmith". The terminal shows the command "ipython" being executed. The output includes the Python version "Python 2.7.13 (default, Jan 11 2017, 15:53:57)", a prompt to type "copyright", "credits" or "license", and the IPython 5.3.0 startup message. It lists several shortcuts: "?" for introduction, "%quickref" for quick reference, "help" for Python's help system, and "object?" for details about 'object'. The prompt "In [1]:" is shown with the input "2+2", followed by the output "Out[1]: 4". The prompt "In [2]:" is shown with a cursor, indicating the next input.

```
Python home/basmith
bjsmith@r213n31:~$ ipython
Python 2.7.13 (default, Jan 11 2017, 15:53:57)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]: 2+2
Out[1]: 4

In [2]:
```

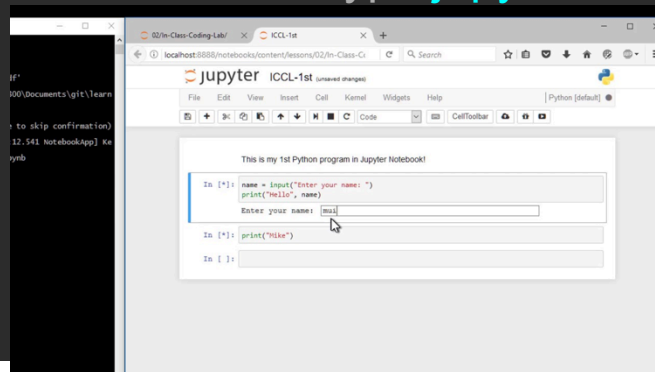
Jupyter Notebook

A Jupyter notebook is a web interface. With it we can use formatting and markdown with our code. This is what we'll be primarily using for this course.

Mac/Linux: In the terminal type `jupyter notebook`

Windows: Click the "Start" button and type "cmd" type `jupyter notebook`

Or open Git Bash and type `jupyter notebook`



Jupyter Notebooks

born out of the IPython project--grew to encompass more languages

The name Jupyter is an indirect acronym of the three core languages it was designed for: J^Ulia, P^YThon, and R

Scripting

Sometimes we just want to execute a program and get results, not interact with our Python code.

In those cases, we use a Python script.

To do so, we can use a text editor of our choice and save the code in a file with extension “.py”.

Text Editors

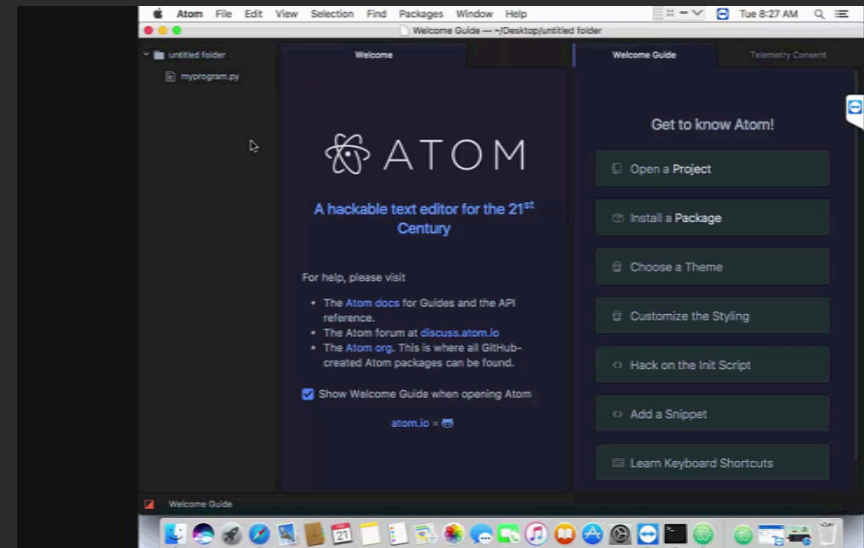
Some common plain text editors include:

Atom

NotePad (Win), NotePad++ (Win)

TextEdit (Mac), TextWrangler (Mac).

Nano (Linux, Mac)



IDEs

Integrated development environments (IDEs) provide tools for writing and testing your software (Brings tools together for you)

PyCharm

Eclipse with PyDev

Atom

Spyder (included in Anaconda)



Introduction to Jupyter Notebooks

Opening a Jupyter Notebook from Git Bash (Windows) or the Terminal (Mac)

```
Nichole's-MacBook-Air:~ nicholebennett$ jupyter notebook
```

Notebook Dashboard



Files

Running

Clusters

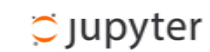
To import a notebook, drag the file onto the listing below or [click here](#).

New ▾



<input type="checkbox"/>	/ examples
<input type="checkbox"/>	..
<input type="checkbox"/>	Builtin Extensions
<input type="checkbox"/>	Customization
<input type="checkbox"/>	Embedding
<input type="checkbox"/>	IPython Kernel
<input type="checkbox"/>	Interactive Widgets
<input type="checkbox"/>	Notebook
<input type="checkbox"/>	Parallel Computing
<input type="checkbox"/>	images
<input type="checkbox"/>	utils
<input type="checkbox"/>	 Index.ipynb

Notebook Dashboard



Files Running Clusters

To import a notebook, drag the file onto the listing below or [click here](#).

New ↕

<input type="checkbox"/>	🏠 / examples
<input type="checkbox"/>	..
<input type="checkbox"/>	Builtin Extensions
<input type="checkbox"/>	Customization
<input type="checkbox"/>	Embedding
<input type="checkbox"/>	IPython Kernel
<input type="checkbox"/>	Interactive Widgets
<input type="checkbox"/>	Notebook
<input type="checkbox"/>	Parallel Computing
<input type="checkbox"/>	images
<input type="checkbox"/>	utils
<input type="checkbox"/>	Index.ipynb

New ▼ ↺

Text File

Folder

Terminal

Notebooks

IForth

IPython Debug

Python 2

Python 3

R

Notebook Dashboard



Notebook Dashboard



Files

Running

Clusters










Currently running Jupyter processes



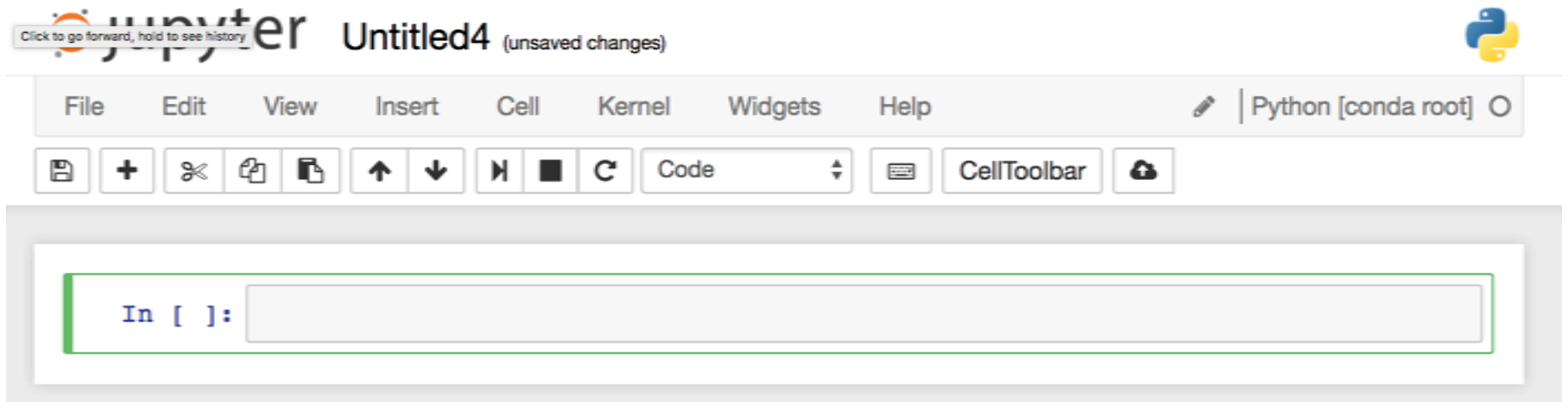
Terminals ▾

There are no terminals running.

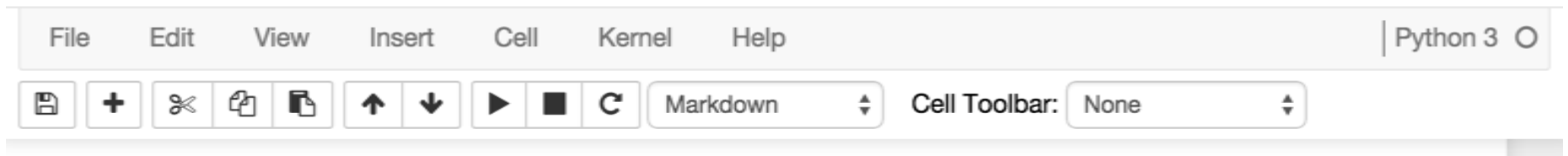
Notebooks ▾

 examples/Notebook/Index.ipynb	Shutdown
 examples/Notebook/What is the IPython Notebook.ipynb	Shutdown
 examples/Notebook/Running the Notebook Server.ipynb	Shutdown
 examples/Notebook/Notebook Basics.ipynb	Shutdown
 examples/Notebook/Running Code.ipynb	Shutdown
 examples/Notebook/Working With Markdown Cells.ipynb	Shutdown
 examples/Notebook/Custom Keyboard Shortcuts.ipynb	Shutdown
 examples/Notebook/JavaScript Notebook Extensions.ipynb	Shutdown
 examples/Notebook/Notebook Security.ipynb	Shutdown

Notebook User Interface (UI)



Menu and Toolbar



Modes

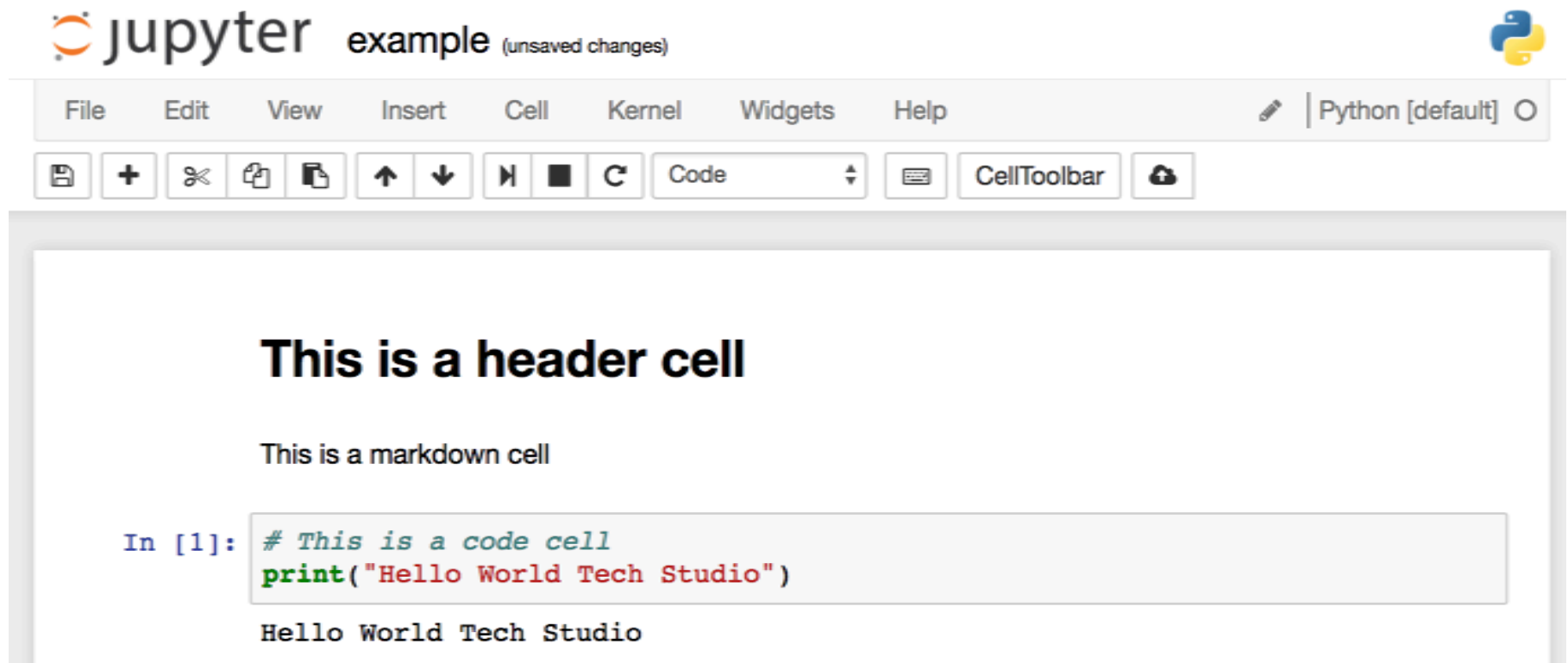
Edit Mode

```
In [1]: a = 10|
```

Command Mode

```
In [1]: a = 10
```


Headers, and markdown, and code (oh my!)



The image shows a Jupyter Notebook interface. At the top, the title bar says "jupyter example (unsaved changes)" with the Python logo on the right. Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and a dropdown menu currently set to "Code". Below the toolbar is a "CellToolbar" button. The main content area contains three cells: a header cell with the text "This is a header cell", a markdown cell with the text "This is a markdown cell", and a code cell. The code cell contains the following code:

```
In [1]: # This is a code cell
print("Hello World Tech Studio")
```

 The output of the code cell is "Hello World Tech Studio".

jupyter example (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Python [default]

Save Add Delete Copy Paste Undo Redo Code CellToolbar

This is a header cell

This is a markdown cell

```
In [1]: # This is a code cell
print("Hello World Tech Studio")
```

Hello World Tech Studio

Keyboard shortcuts

Go to Help > Keyboard Shortcuts (or Cmd + Shift + P/Ctrl + Shift + P)

Some useful ones:

- Esc will take you into command mode where you can navigate around your notebook with arrow keys....While in command mode:
 - A to insert a new cell above the current cell, B to insert a new cell below.
 - M to change the current cell to Markdown, Y to change it back to code
 - D + D (press the key twice) to delete the current cell
- Enter will take you from command mode back into edit mode for the cell.

Keyboard shortcuts

Go to Help > Keyboard Shortcuts (or Cmd + Shift + P/Ctrl + Shift + P)

Some useful ones:

- Shift + Tab will show you the Docstring (documentation) for the the object you have just typed in a code cell
 - keep pressing this short cut to cycle through a few modes of documentation.
- Ctrl + Shift + - will split the current cell into two from where your cursor is.
- Esc + F Find and replace on your code but not the outputs.

Keyboard Shortcuts

Go to Help > Keyboard Shortcuts (or Cmd + Shift + P/Ctrl + Shift + P)

Some useful ones:

- **Select Multiple Cells:**
 - Shift + J or Shift + Down selects the next sell in a downwards direction. You can also select sells in an upwards direction by using Shift + K or Shift + Up.
 - Once cells are selected, you can then delete / copy / cut / paste / run them as a batch. This is helpful when you need to move parts of a notebook.
 - You can also use Shift + M to merge multiple cells.

Jupyter Notebook Demo

Jupyter Notebook Obstacle Course

1. Open a new notebook
2. Name the file something specific and useful
3. Add a header for your name
4. Add a code cell
5. Print a short sentence about yourself
6. Add some markdown
7. Explore!

CTRL + C to close out of
the Jupyter Notebook from
the Terminal or Git Bash


Getting Help

Help menu has documentation for some common libraries



Place ? before a library, method, or variable for a quick syntax reference

You can execute shell commands in notebooks by prepending the command with !



Hello, Colaboratory 

File Edit View Insert Runtime Tools Help

 CODE  TEXT  CELL  CELL  COPY TO DRIVE

COPY

Table of contents Code snippets Files 

Getting Started

Highlighted Features

TensorFlow execution

GitHub

Visualization

Forms

Examples

Local runtime support

 SECTION



Welcome to Colaboratory!

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. See

Getting Started

- [Overview of Colaboratory](#)
- [Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage](#)
- [Importing libraries and installing dependencies](#)
- [Using Google Cloud BigQuery](#)
- [Forms, Charts, Markdown, & Widgets](#)
- [TensorFlow with GPU](#)
- [TensorFlow with TPU](#)
- [Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow](#)
- [Using Colab with GitHub](#)

Highlighted Features

Seedbank

Looking for Colab notebooks to learn from? Check out [Seedbank](#), a place to discover interactive machine learning examples.

 CODE

 TEXT

nbviewer

A simple way to share Jupyter Notebooks

Enter the location of a Jupyter Notebook to have it rendered here:

Programming Languages

Python

```
In [1]: display()
```

IP[y]: IPython
Interactive Computing

```
In [2]: from IPython.display import SVG  
SVG(filename='python-logo.png')
```

```
Out[2]:
```

 python™

IRuby



```
File.open("lib/ruby/static/base/images/ipyblogo.png")
```

 **IRuby: Notebook**

Julia

An Julia Preview

This notebook is a preview demo of Julia, a [Julia language](#) backend combined with the [Jupyter](#) interactive environment. This combination allows you to interact with the Julia language using Python's powerful [interactive notebooks](#), which combines code, formatted text, math, and multimedia in a single document.



→ Note: this is a preview, because it relies on pre-release bleeding-edge versions of Julia, Python, and several Julia packages, as explained on the [Julia GitHub page](#), and functionality is evolving rapidly. We hope to have a more polished release soon.

Basic Julia interaction

Self-Diagnosis: Comfort With Python

Line up according to how well you think you'd be able to teach Python basics to someone else (e.g. functions, for loops, if/else, lists, dictionaries...)

Then:

Mob Coding (5 groups of 3) on Part 1 and 2

1 driver, 1 navigator, 1 thinker

Goal is to get every member of the group to understand

Then:

Do Python Foundations Practice on your own.