# Introduction to Python for Social Science

## Lecture 1 - Introduction to Python

Musashi Harukawa, DPIR

1st Week Hilary 2020

# Course Overview

# Schedule

8-week long course will take place on Wednesdays at the IT Lab in the Manor Road Building. Each week will consist of a three-hour session divided roughly as follows:

- 1600-1645: Lecture
- 1645-1730: Coding Tutorial
- 1730-1745: Break
- 1745-1845: Workshop
- 1845-1900: Clinic

- This course runs for eight weeks, usually on Wednesday, but in week 3 will be moved to Monday as I will be going to Japan for some family matters.
- I've bokoked the room for three hours, but only the first 1.5 hours are taught. Thus you can leave at 5:45pm if you need to.
- The timings are roughly as follows: **Go through breakdown**

# Topics:

1. Introduction to Python and the Development Environment
2. Data Structures and `pandas` I
3. Data Structures and `pandas` II
4. Data Visualisation
5. Machine Learning with `scikit-learn` I
6. Machine Learning with `scikit-learn` II
7. Mining the Web
8. Introduction to Natural Language Processing

- I've chosen six topics for the eight weeks.

1. The first week will be focused on getting you to understand a bit better what Python is, and will teach some basics.
2. The second and third week focus on working with data; cleaning, reshaping, writing and reading, and most importantly *understanding*.
3. The fourth week will introduce two libraries for making data-based visuals; these will be publication grade, and can be used for academic papers or in a professional setting.

# Lecture

- Lectures focus on the methodological motivation and context for the techniques.
- Where relevant, some computer science and information theory will be discussed.
- The slides will be made available at the following:
  - https://github.com/muhark/dpir-intro-python
  - Canvas?
- Please feel free to stop me and ask questions! I have alloted time for this.

Explain lecture flow. In the lecture, we look at the intersection of social science methodology and computer science theory.

# Coding Tutorial

- In this section I explain the nitty-gritty of actually implementing these problems in code.
- Pull out your laptops and code along with me while we work through examples!

# Workshop

- In the workshop, you will work through a number of set programming problems and discussion questions.
- This part is optional: my goal is to give everyone a space where they can work collaboratively on these problem sets, and I will be around to try and help.

# Clinic

- The final 30 minutes are reserved for one-on-one help. Sign-up will be formalised from next week onwards.

# Feedback

- This is my first try at this course, and I'm looking for continuous feedback.
- Feedback can either be:
  - sent to me at `musashi.harukawa@politics.ox.ac.uk`
  - or communicated via a Google survey (link on website)
- This is a brand new course, and far from perfect.
- I would appreciate and welcome any feedback on any aspect.
- Please also feel free to come talk to me in person!

# Week 1: Introduction to Python and the Development Environment

# Overview:

This week will cover the following points:

1. What is Python?...
2. ... and what can I use it for?
3. What are the tools I have to write, test and run Python code?
4. Coding tutorial I: Base Data Types and Structures

# What is Python?...

Python is an *open-source, general-purpose scripting language*.

# Open-Source

- Built by a community
- Maintained by a community
- Free to use for all

# General-Purpose

- If you're doing it on a computer and there's some repetitive element, then you can automate it in Python.
- Python isn't limited to Data Science, but it's very popular with data scientists!

Large community means that a larger number of people create, contribute to, and maintain the data analysis tools that we all use.

# Scripting

- No strict definition for what a “script” is.
- Series of commands to automate some task.
- Like a pipeline: takes some inputs, does some things to these inputs, and gives back some outputs.

It’s good to keep the input-output framework in your head.

# Language

- Python is a language, and not an application.
- Practical difference for you:
  - most applications provide you options to select from.
  - languages require to generate commands from accepted rules.
- Upshot is that you can do nearly anything with Python!

and what can I use Python for?

# I want to...

- Clean up my messy data!
- Run analyses with (hundreds of) millions of data points
  - it won't fit into an excel spreadsheet!
- I want to automate downloading several decades of newspaper articles!
- I want to create beautiful (interactive) visuals to accompany my analyses!
- I want to uncover hidden structures linking parliamentary committees!
- I want to track the changing meaning of a concept over a century!
- Again: *any repetitive task done on a computer can be automated with Python*.

# Comparison: Python vs R

| Task | python | R |
|---|---|---|
| General Purpose Programming | Yes | No |
| Regression Analysis | Kinda | Yes |
| Machine Learning | Yes | Yes |
| Web Scraping | Yes | Kinda |
| Natural Language Processing[1] | Yes | Yes |
| Data Visualisation | Yes | Yes |

Conclusion: . . . it depends, but ideally you want to learn both!

[1] Python and R both provide extensive and powerful natural language processing libraries, e.g. `nltk`, `gensim` in Python; `tm`, `quanteda` in R, and `spaCy` in both. Unfortunately, there are many techniques that are only implemented in one language but not the other.

# Tools of the Trade

# Anaconda

- Environment and software manager.
- Can be used from the command line (`cli`) or browser-like interface (anaconda-navigator).

At this point, I switch windows to open up the anaconda navigator, and go through the various relevant tabs. These include the launcher for various apps and the environment manager.

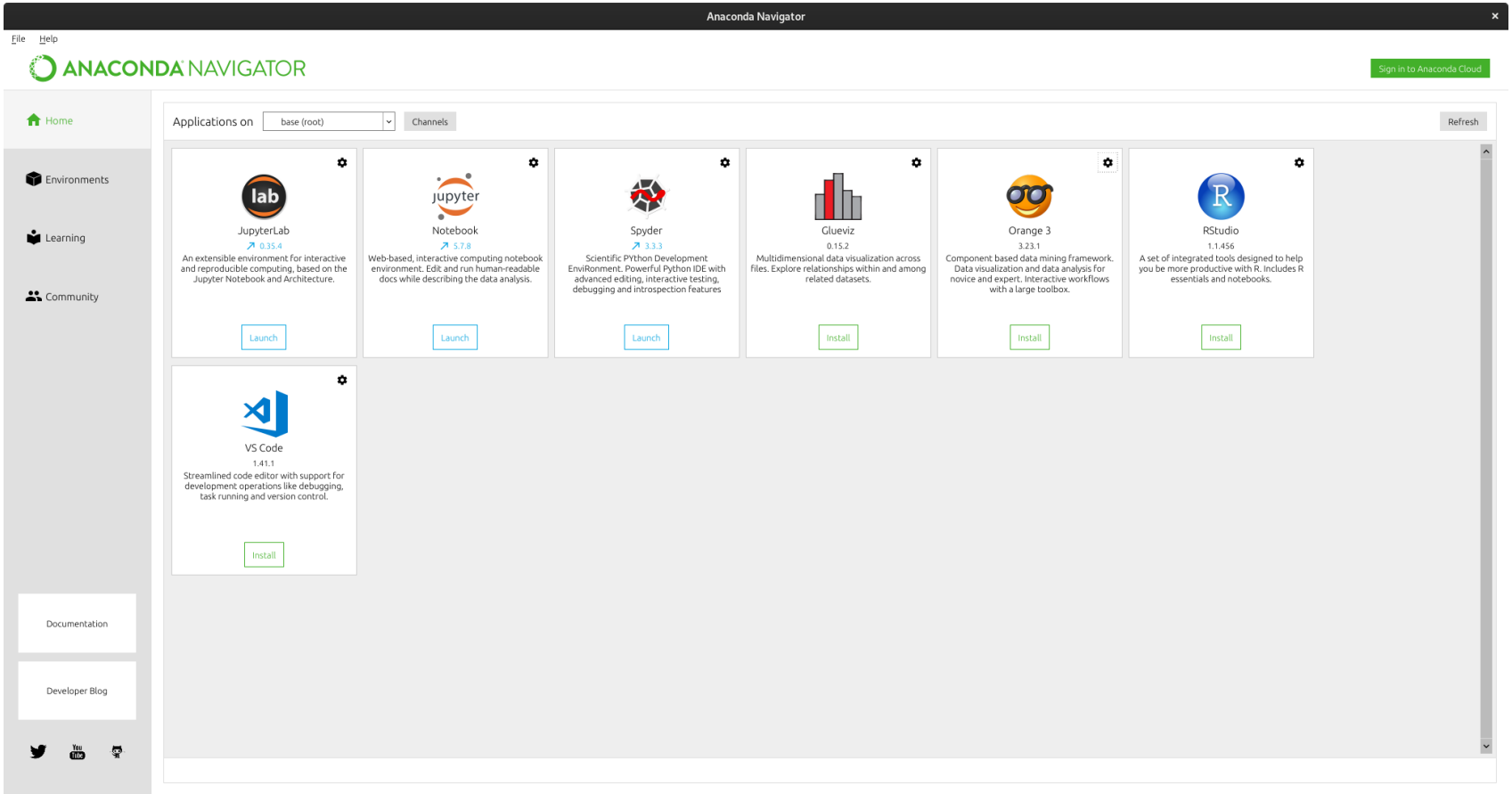



Figure 1: Anaconda Navigator

- Make sure students can find navigator, and open applications from it.

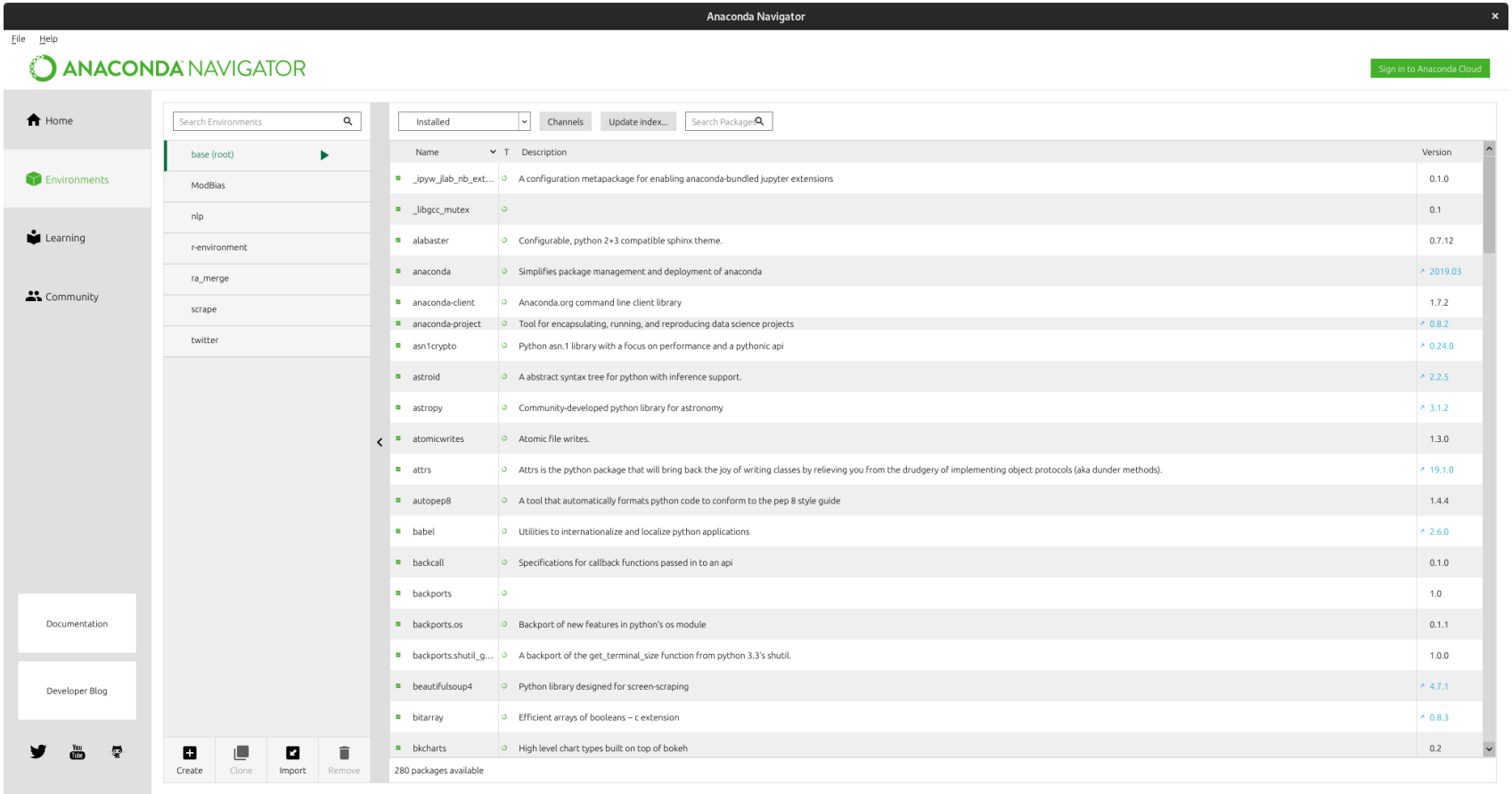



Figure 2: Environment Manager

- Show students how to install packages here. Also mention that this can be done from the command line.

# Jupyter

- Interactive code editor.
- Popular, but has its detractors.
- Multiple options: console, notebook and lab

Explain that we are using lab in this course, but notebook or console are options.

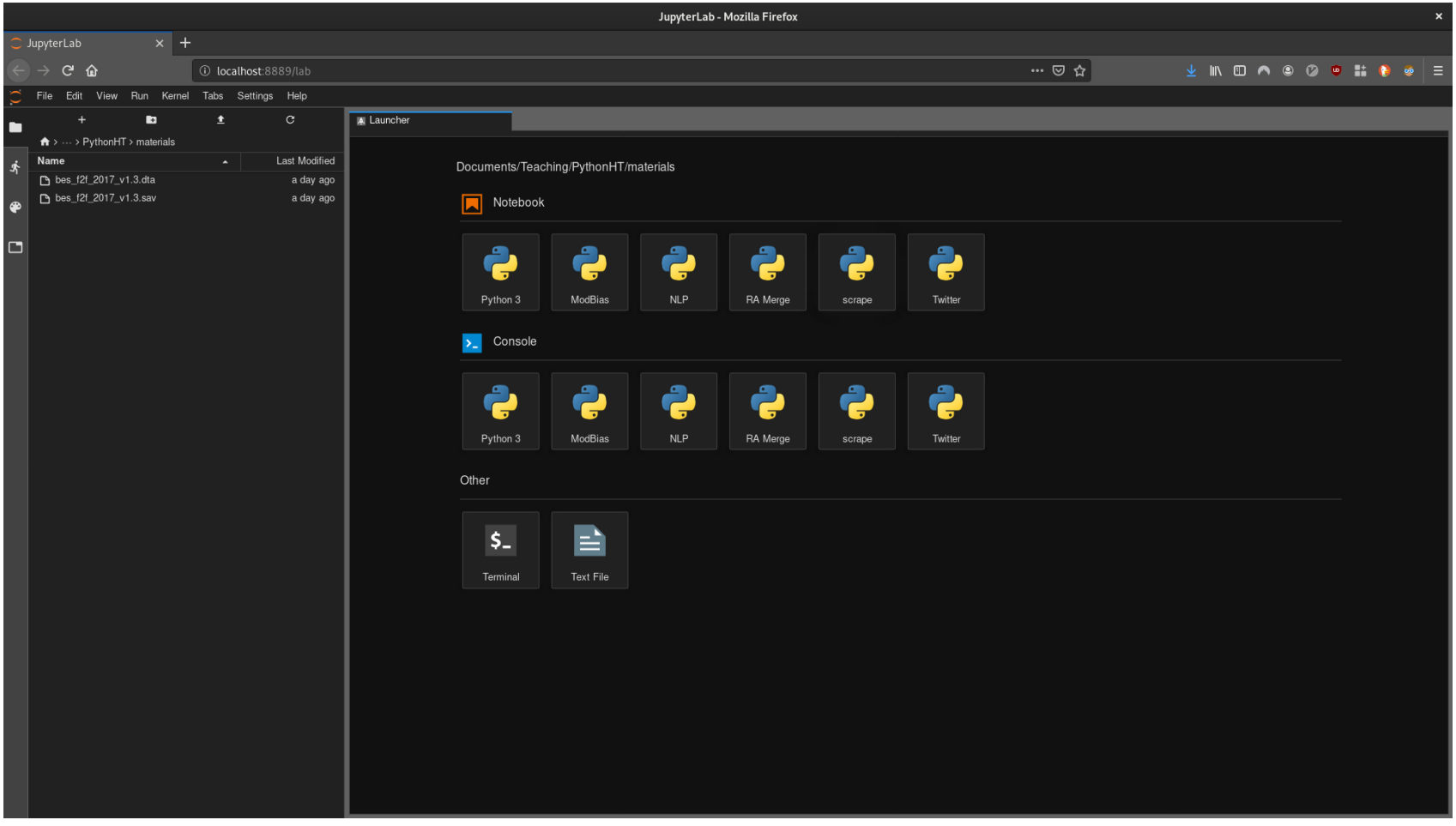



Figure 3: Jupyter Lab Launcher

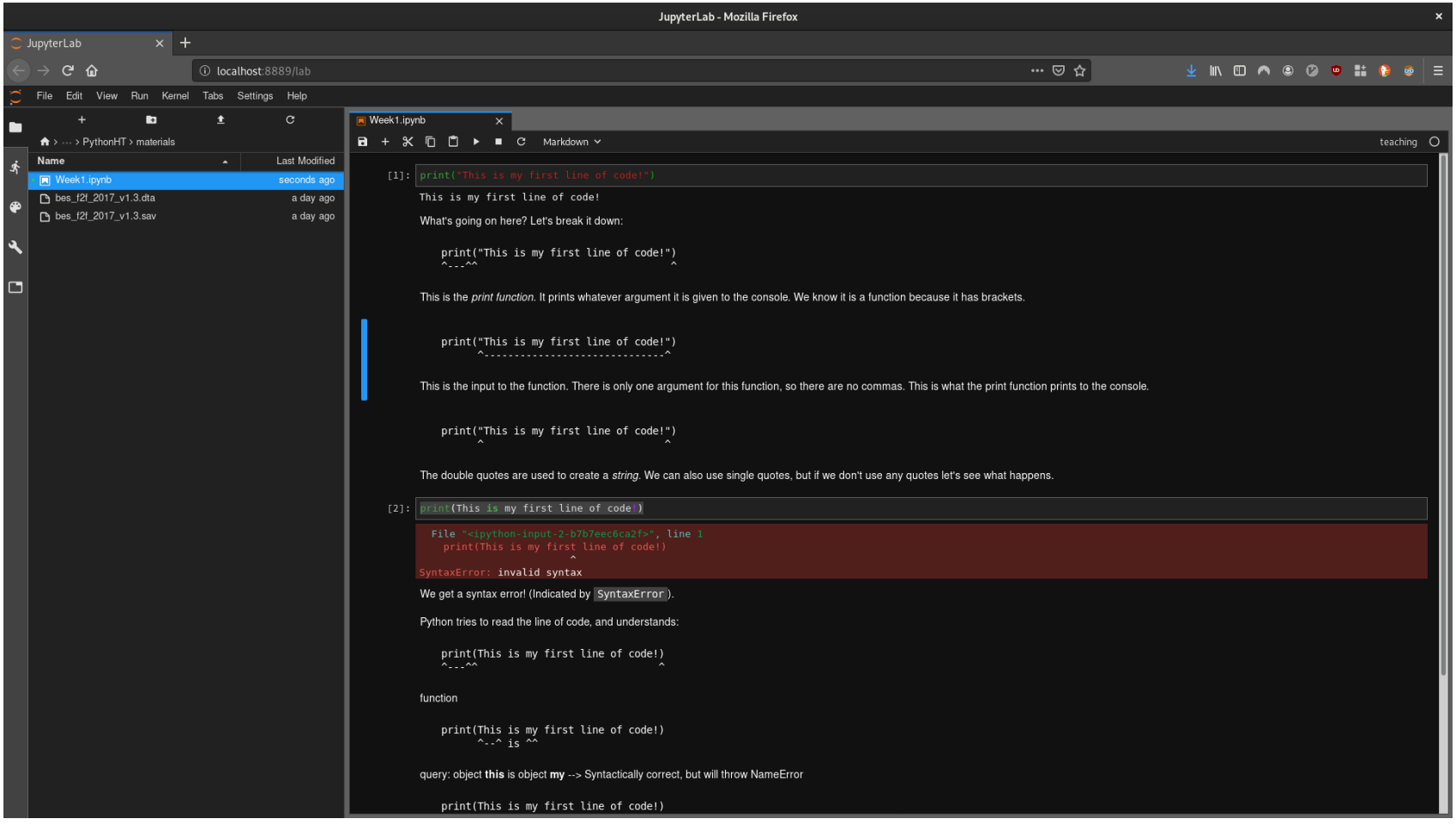



Figure 4: Jupyter Lab Editor

Launch Jupyter Lab session. Show how you can navigate a file tree, and then create a new notebook and name it.

# Other Options

- I prefer to use Atom with Hydrogen
- PyCharm is popular with developers
- If you've spent a lot of time with RStudio, you may prefer Spyder.

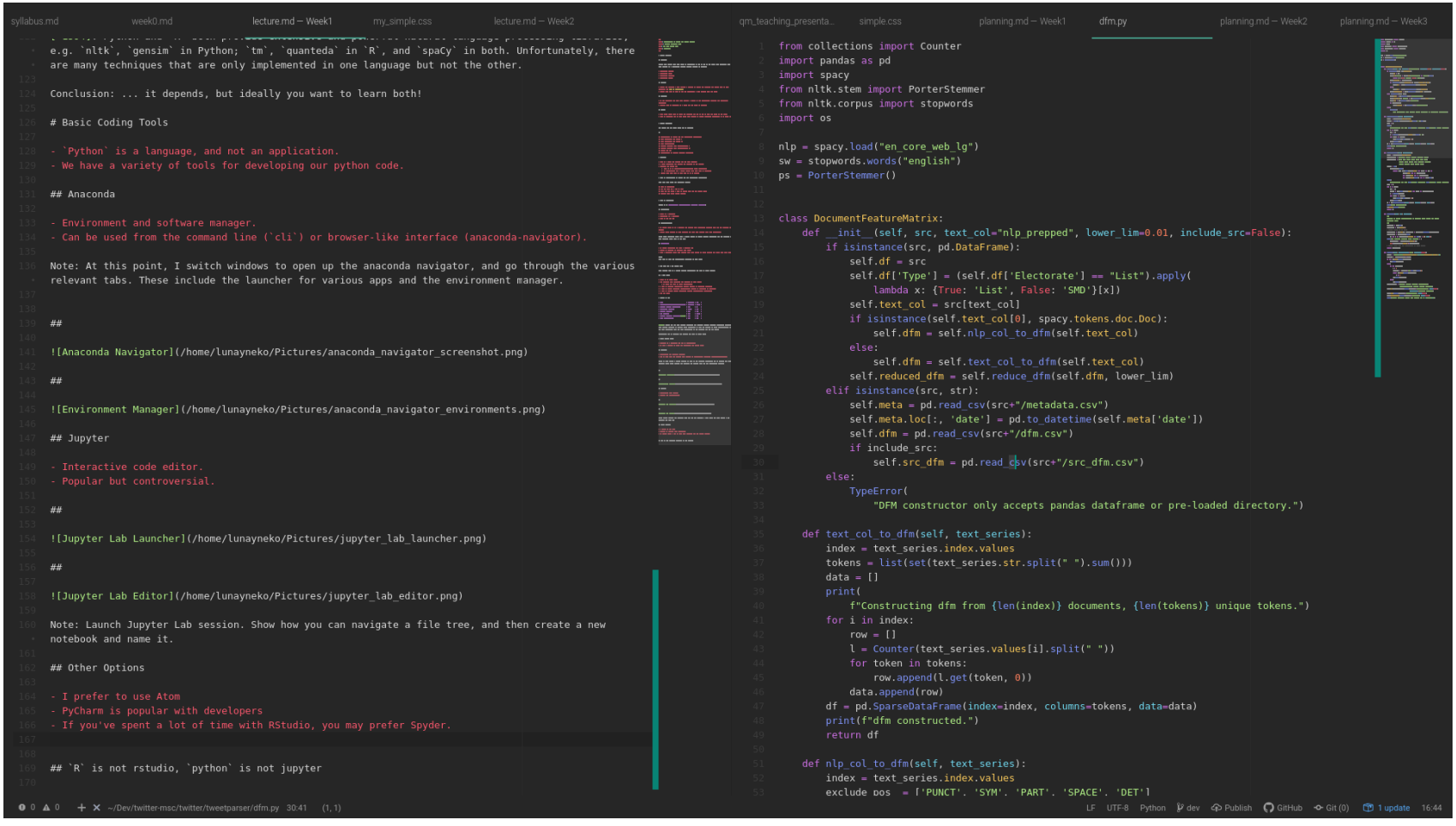



Figure 5: Atom

# Basic Workflow

1. Open up Anaconda Navigator
2. Open up Jupyter Lab (or Notebook)
3. Navigate to relevant directory
4. Open pre-existing notebook, or create new one.
5. Start coding!

# This Week

# Data is more than data

- Today I introduce two fundamental, but abstract aspects of coding:
  - Data Types
  - Data Structures

Note that we will continue to discuss these topics for 3 weeks.

# Why Automate?

- In general, as social scientists using computational methods, our goal is to automate some component of our analysis.
- The advantage of automation is cost, scale, and scope.
- But in order to harness these methods, we need to represent our observations in a way that algorithms and programs can utilise.
  - This process of quantifying and structuring our observations usually entails the loss of some information.
- This automation is not limited to collecting data. Running a regression or sorting your responses by data is the automation of doing this by hand.
- Some qualitative scholars I speak to contend that the validity of the quantitative endeavour ends there.
  - Are there unquantifiable things?
- I'm more optimistic about what is possible, and think that the key to having valid quantitative inferences is to be extremely clear on the connection between the data in your analysis and the actual events you are measuring.

# Bridging the Gap between Qualitative and Quantitative Methods

- Choosing a representation of your information that retains relevant properties is key.
- To read more about this particular debate, a good starting point is Stevens (1946).
- I would love to discuss this further, but this isn't really that kind of methods class.

# Data Types

Some (statistical) data types:

- Logical
- Numerical
- Categorical
- Text
- Date and time

- What do I mean when I talk about different "types" of data?
- Small exercise: what are instances of each of these?

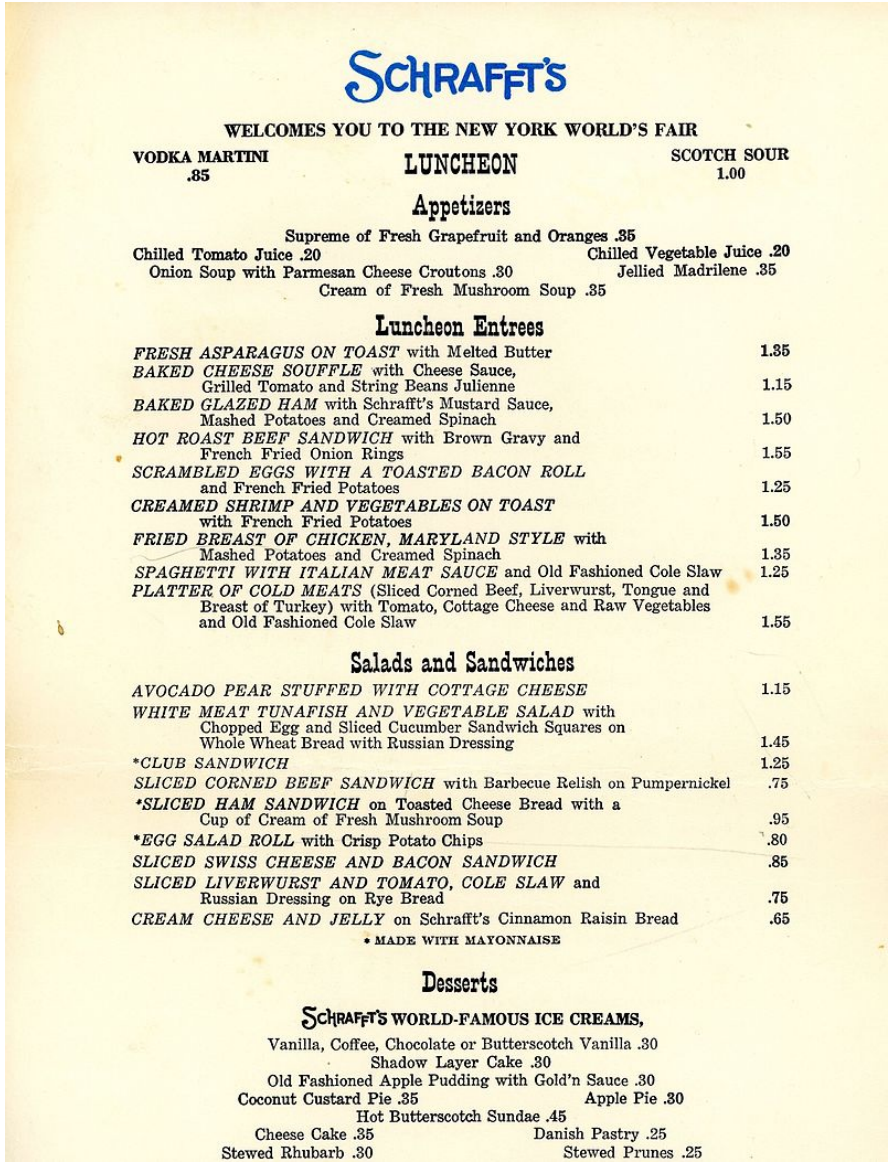
# Representing Data on a Computer

- *Good news:* Python, like most modern programming languages, has ways to represent each of the data types listed above.
- *Bad news:* At a fundamental level, this is being stored as 0's and 1's.
- *Take away:* Take the time to understand the relationship between:
  - your empirical observations,
  - the abstracted representation of them in your mathematical model,
  - the approximation of this in your computational model.
- I'm making a bit of an assumption here about the theory-generating workflow, in that a stylised mathematical model is usually prior to a computational/empirical approach.

# Data Structures

- Data types are concerned with the representation of individual data points, or observations.
- Data structures are concerned with the relations between observations.
  - Are the data points members of the same set?
  - Are the data points members of the same sequence?
  - Are the data points different features of single empirical unit?

# Exercise



SCHRAFFT'S

WELCOMES YOU TO THE NEW YORK WORLD'S FAIR

VODKA MARTINI .85 — SCOTCH SOUR 1.00

## LUNCHEON

### Appetizers

Supreme of Fresh Grapefruit and Oranges .35

Chilled Tomato Juice .20 — Chilled Vegetable Juice .20

Onion Soup with Parmesan Cheese Croutons .30 — Jellied Madrilene .35

Cream of Fresh Mushroom Soup .35

### Luncheon Entrees

| | |
|---|---|
| *FRESH ASPARAGUS ON TOAST* with Melted Butter | 1.35 |
| *BAKED CHEESE SOUFFLE* with Cheese Sauce, Grilled Tomato and String Beans Julienne | 1.15 |
| *BAKED GLAZED HAM* with Schrafft's Mustard Sauce, Mashed Potatoes and Creamed Spinach | 1.50 |
| *HOT ROAST BEEF SANDWICH* with Brown Gravy and French Fried Onion Rings | 1.55 |
| *SCRAMBLED EGGS WITH A TOASTED BACON ROLL* and French Fried Potatoes | 1.25 |
| *CREAMED SHRIMP AND VEGETABLES ON TOAST* with French Fried Potatoes | 1.50 |
| *FRIED BREAST OF CHICKEN, MARYLAND STYLE* with Mashed Potatoes and Creamed Spinach | 1.35 |
| *SPAGHETTI WITH ITALIAN MEAT SAUCE* and Old Fashioned Cole Slaw | 1.25 |
| *PLATTER OF COLD MEATS* (Sliced Corned Beef, Liverwurst, Tongue and Breast of Turkey) with Tomato, Cottage Cheese and Raw Vegetables and Old Fashioned Cole Slaw | 1.55 |

### Salads and Sandwiches

| | |
|---|---|
| *AVOCADO PEAR STUFFED WITH COTTAGE CHEESE* | 1.15 |
| *WHITE MEAT TUNAFISH AND VEGETABLE SALAD* with Chopped Egg and Sliced Cucumber Sandwich Squares on Whole Wheat Bread with Russian Dressing | 1.45 |
| **CLUB SANDWICH* | 1.25 |
| *SLICED CORNED BEEF SANDWICH* with Barbecue Relish on Pumpernickel | .75 |
| **SLICED HAM SANDWICH* on Toasted Cheese Bread with a Cup of Cream of Fresh Mushroom Soup | .95 |
| **EGG SALAD ROLL* with Crisp Potato Chips | .80 |
| *SLICED SWISS CHEESE AND BACON SANDWICH* | .85 |
| *SLICED LIVERWURST AND TOMATO, COLE SLAW* and Russian Dressing on Rye Bread | .75 |
| *CREAM CHEESE AND JELLY* on Schrafft's Cinnamon Raisin Bread | .65 |

* MADE WITH MAYONNAISE

### Desserts

SCHRAFFT'S WORLD-FAMOUS ICE CREAMS,

Vanilla, Coffee, Chocolate or Butterscotch Vanilla .30

Shadow Layer Cake .30

Old Fashioned Apple Pudding with Gold'n Sauce .30

Coconut Custard Pie .35 — Apple Pie .30

Hot Butterscotch Sundae .45

Cheese Cake .35 — Danish Pastry .25

Stewed Rhubarb .30 — Stewed Prunes .25

Figure 6: Menu

- Question to discuss with partner, how would you represent this

# And now. . .

Let's get to coding!

# Coding Recap

# Variable Assignment

Variables can be assigned with =.

# Four Basic Data Types

There are four basic data types in Python. These are:

- String
- Integer
- Float
- Boolean

# String

- *A sequence of characters*.
- Behaves like a sequence; can be indexed with `[index]`

# Integer

- Whole numbers.
- Can be positive or negative.

# Float

- Decimal numbers.
- Behave unexpectedly. Remember: `0.1*3==0.3` returns `False`.

# Boolean

- True/False
- Behaves similarly to integers 0 and 1.

# Two basic data structures

We learned about two basic data structures:

- Lists
- Dictionaries

# Lists

- Lists are an ordered sequence of values.
- Created by writing a sequence of comma-separated values between square brackets:
  - i.e. `[1, 2, 5, "some string"]`
- Lists are mutable; values can be changed in place without creating a new variable.
- Lists can be indexed the same way as strings:
  - `[n]` to get the n+1th element.
  - `[m:n]` to get all elements from m+1 to n.

# Dictionaries

- Unordered mapping of *keys* to *values*.
  - Cannot be indexed numerically, and if iterated over, will not return values in the same order.
- Created by writing a list of `key:value` pairs separated by commas between curly braces.
  - i.e. `{"cat": "meow", "dog": "bork"}`
- `some_dict[some_key]` returns the corresponding value for `some_key` in `some_dict`
- To see all of the keys, use the `.keys()` method of the dict, i.e. `some_dict.keys()`
- To see all of the values, use the `.values()` method of the dict, i.e. `some_dict.values()`

# Next Week: Data Structures and `pandas` I

# Readings

The following sections of *Python for Data Analysis: Data Wrangling with Pandas, NumPy and IPython, 2nd edition* are relevant to this lecture:

*Useful*: - 2.2: IPython Basics - 3.1: Data Structures and Sequences

*Interesting*: - 1.2 Why Python for Data Analysis?