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:

- 1615-1715: Lecture
- 1715-1725: Break
- 1725-1825: Workshop
- 1825-1925: Clinic

Mention that class in 3rd week (5th Feb) will be moved to Monday (3rd Feb).

Lecture

- During the lectures I will discuss a mixture of theory and methods. 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.

Slides are generated with `reveal.js`. Should be viewable on any device with a browser.

Workshop

- In the workshop, you will work through a number of set programming problems and discussion questions.
- Answers will be discussed as a class for the final 10 minutes.

Clinic

- Each week, there will be tasks and projects for you to try in your own time, if you wish.
- I'll be available for an hour after each session to answer questions one-to-one or in small groups.

Course Structure

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

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 (to be sent each week).
- 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.

and what can I use Python for?

I want to. . .

Some reasons you, as a social science researcher, may want to learn Python:

- 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!
- . . . and more!

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.

Basic Coding Tools

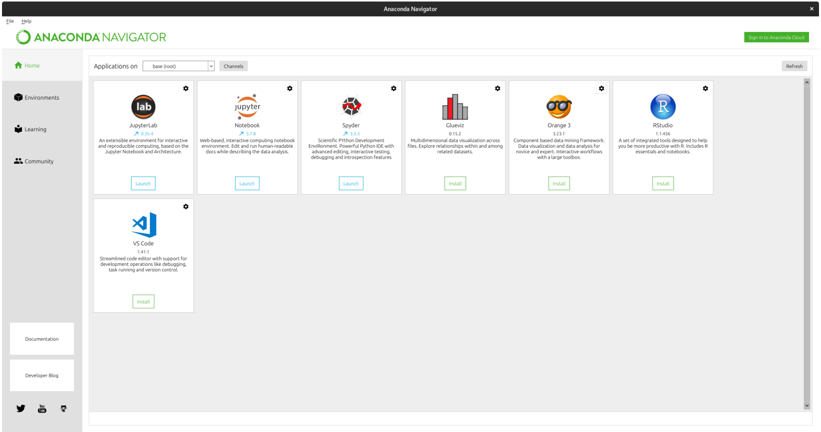
Language $\neq$ Editor

- `Python` is a language, and not an application.
- We have a variety of tools for developing our python code.

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.



JupyterLab
0.35.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
5.7.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Spyder
3.3.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

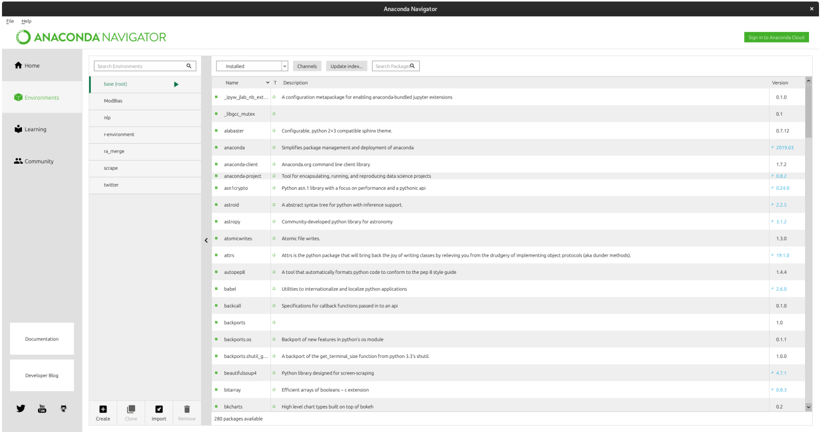
Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code
1.41.1
Streamlined code editor with support for development operations like debugging, task running and version control.

Figure 1: Anaconda Navigator

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



Name	T	Description	Version
_ipyw_jlab_nb_ext...	○	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
_libgcc_mutex	○		0.1
alabaster	○	Configurable, python 2+3 compatible sphinx theme.	0.7.12
anaconda	○	Simplifies package management and deployment of anaconda	↗ 2019.03
anaconda-client	○	Anaconda.org command line client library	1.7.2
anaconda-project	○	Tool for encapsulating, running, and reproducing data science projects	↗ 0.8.2
asn1crypto	○	Python asn.1 library with a focus on performance and a pythonic api	↗ 0.24.0
astroid	○	A abstract syntax tree for python with inference support.	↗ 2.2.5
astropy	○	Community-developed python library for astronomy	↗ 3.1.2
atomicwrites	○	Atomic file writes.	1.3.0
attrs	○	Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).	↗ 19.1.0
autopep8	○	A tool that automatically formats python code to conform to the pep 8 style guide	1.4.4
babel	○	Utilities to internationalize and localize python applications	↗ 2.6.0
backcall	○	Specifications for callback functions passed in to an api	0.1.0
backports	○		1.0
backports.os	○	Backport of new features in python's os module	0.1.1
backports.shutil_g...	○	A backport of the get_terminal_size function from python 3.3's shutil.	1.0.0
beautifulsoup4	○	Python library designed for screen-scraping	↗ 4.7.1
bitarray	○	Efficient arrays of booleans -- c extension	↗ 0.8.3
bkcharts	○	High level chart types built on top of bokeh	0.2

280 packages available

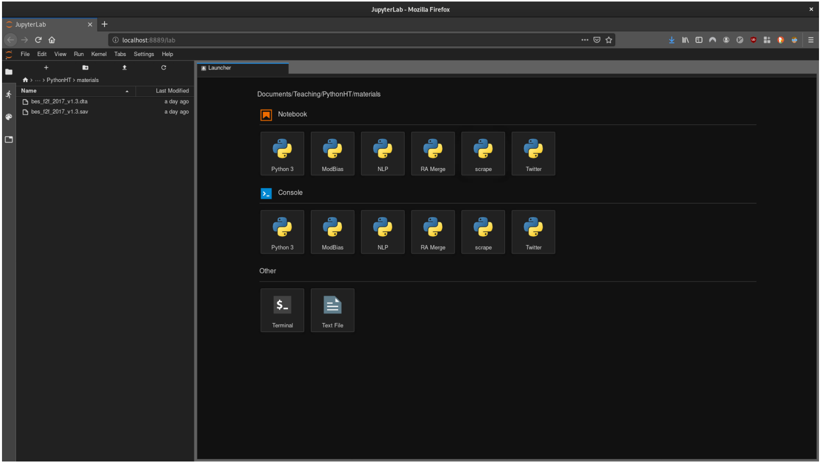
Figure 2: Environment Manager

- 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.

- I don't use it, but I started off with it. I recommend that you give it a go.



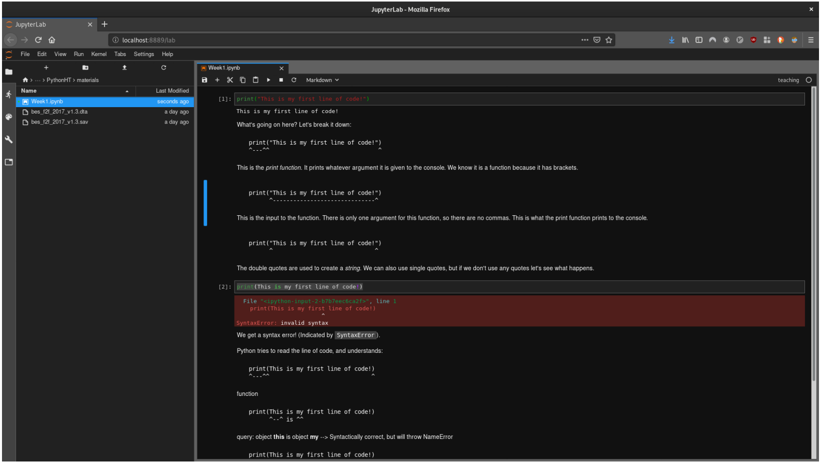
localhost:8889/lab

› ··· › PythonHT › materials

Name	Last Modified
bes_f2f_2017_v1.3.dta	a day ago
bes_f2f_2017_v1.3.sav	a day ago

Documents/Teaching/PythonHT/materials

Figure 3: Jupyter Lab Launcher



Name	Last Modified
Week1.ipynb	seconds ago
bes_f2f_2017_v1.3.dta	a day ago
bes_f2f_2017_v1.3.sav	a day ago

```
[1]: print("This is my first line of code!")
```

```
This is my first line of code!
```

What's going on here? Let's break it down:

```
print("This is my first line of code!")
^---^^                               ^
```

This is the *print function*. It prints whatever argument it is given to the console. We know it is a function because it has brackets.

```
print("This is my first line of code!")
      ^-----------------------------^
```

This is the input to the function. There is only one argument for this function, so there are no commas. This is what the print function prints to the console.

```
print("This is my first line of code!")
      ^                             ^
```

The double quotes are used to create a *string*. We can also use single quotes, but if we don't use any quotes let's see what happens.

```
[2]: print(This is my first line of code!)
```

```
  File "<ipython-input-2-b7b7eec6ca2f>", line 1
    print(This is my first line of code!)
                         ^
SyntaxError: invalid syntax
```

We get a syntax error! (Indicated by `SyntaxError`).

Python tries to read the line of code, and understands:

```
print(This is my first line of code!)
^---^^                              ^
```

function

```
print(This is my first line of code!)
      ^--^ is ^^
```

query: object **this** is object **my** --> Syntactically correct, but will throw NameError

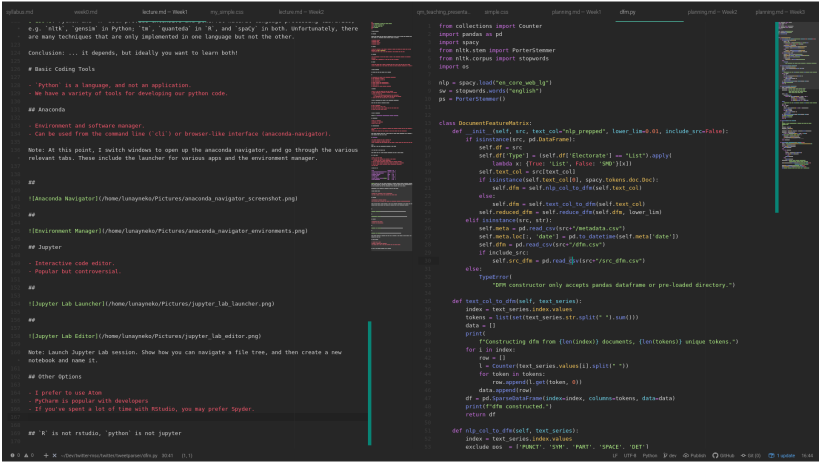
```
print(This is my first line of code!)
```

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.



```
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.

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

# Basic Coding Tools

- `Python` is a language, and not an application.
- We have a variety of tools for developing our python code.

## Anaconda

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

Note: 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.


##

![Anaconda Navigator](/home/lunayneko/Pictures/anaconda_navigator_screenshot.png)

##

![Environment Manager](/home/lunayneko/Pictures/anaconda_navigator_environments.png)

## Jupyter

- Interactive code editor.
- Popular but controversial.

##

![Jupyter Lab Launcher](/home/lunayneko/Pictures/jupyter_lab_launcher.png)

##

![Jupyter Lab Editor](/home/lunayneko/Pictures/jupyter_lab_editor.png)

Note: 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
- PyCharm is popular with developers
- If you've spent a lot of time with RStudio, you may prefer Spyder.


## `R` is not rstudio, `python` is not jupyter

```

```python
from collections import Counter
import pandas as pd
import spacy
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import os

nlp = spacy.load("en_core_web_lg")
sw = stopwords.words("english")
ps = PorterStemmer()


class DocumentFeatureMatrix:
    def __init__(self, src, text_col="nlp_prepped", lower_lim=0.01, include_src=False):
        if isinstance(src, pd.DataFrame):
            self.df = src
            self.df['Type'] = (self.df['Electorate'] == "List").apply(
                lambda x: {True: 'List', False: 'SMD'}[x])
            self.text_col = src[text_col]
            if isinstance(self.text_col[0], spacy.tokens.doc.Doc):
                self.dfm = self.nlp_col_to_dfm(self.text_col)
            else:
                self.dfm = self.text_col_to_dfm(self.text_col)
            self.reduced_dfm = self.reduce_dfm(self.dfm, lower_lim)
        elif isinstance(src, str):
            self.meta = pd.read_csv(src+"/metadata.csv")
            self.meta.loc[:, 'date'] = pd.to_datetime(self.meta['date'])
            self.dfm = pd.read_csv(src+"/dfm.csv")
            if include_src:
                self.src_dfm = pd.read_csv(src+"/src_dfm.csv")
        else:
            TypeError(
                "DFM constructor only accepts pandas dataframe or pre-loaded directory.")

    def text_col_to_dfm(self, text_series):
        index = text_series.index.values
        tokens = list(set(text_series.str.split(" ").sum()))
        data = []
        print(
            f"Constructing dfm from {len(index)} documents, {len(tokens)} unique tokens.")
        for i in index:
            row = []
            l = Counter(text_series.values[i].split(" "))
            for token in tokens:
                row.append(l.get(token, 0))
            data.append(row)
        df = pd.SparseDataFrame(index=index, columns=tokens, data=data)
        print(f"dfm constructed.")
        return df

    def nlp_col_to_dfm(self, text_series):
        index = text_series.index.values
        exclude_pos  = ['PUNCT', 'SYM', 'PART', 'SPACE', 'DET']
```

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!

Motivation

- Today we learn about two fundamental, but abstract aspects of coding:
  - Data Types
  - Data Structures

I want to spend a brief moment to explain why this is relevant.

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 structure our information in a way that algorithms and programs can utilise.
  - This process of quantifying and structuring our observations usually entails the loss of some information.
- 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:

- Numerical
  - Interval
  - Ratio
  - Count
- Text
- Date and time
- Logical
- Categorical

(*Before points*) - 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

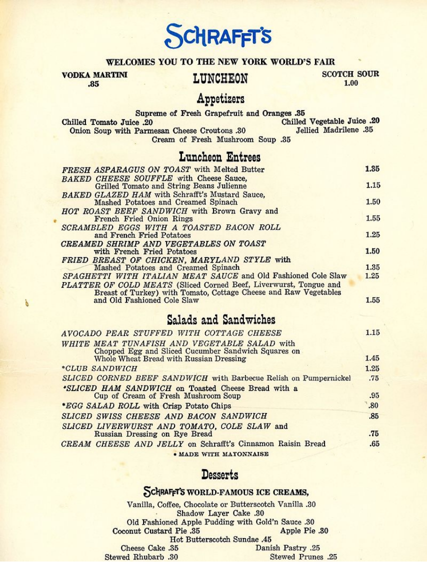
- *Good news:*
  - Python, like most modern programming languages, has ways to represent each of the data types listed above.
- *Bad news:*
  - At a fundamental level, this is being stored as 0's and 1's.
- *Take away:*
  - Take the time to understand the relationship between:
    - your empirical observations,
    - 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

LUNCHEON

SCOTCH SOUR 1.00

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?