GENERAL ASSEMBLY

CLASS 2: PYTHON FOR DATA SCIENCE

Recap

- What is Data Science?
- Data Science Workflow
- Unix class work

Agenda

1. Questions from last Exit Ticket
2. Why Python?
3. Programming workflow and tools review
4. Python in the terminal
5. Python Lab
6. Pandas review
7. Pandas Lab

Exit Ticket Questions

1. What is the *actual* curriculum? (class github)
2. Keys to success in class?
3. Difference between analytics & data science?
4. How to view markdown files?
5. When would I need to use something like python?
6. Advantages of python over other languages?
7. Why use ipython/Jupyter?
8. Why learn bash?
9. What are the appropriate models to use?

I. INTRO TO PYTHON

Q: What is Python?

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*

high-level: *interpreted (not compiled)*

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*

high-level: *interpreted (not compiled)*

dynamic: *things that would typically happen at compile time happen at runtime instead (eg, dynamic typing)*

DYNAMIC TYPING

```
>>> x = 1
>>> x
1
>>> x = 'horseshoe'
>>> x
'horseshoe'
>>> _
```

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*
high-level: *interpreted (not compiled)*
dynamic: *things that would typically happen at compile time happen at runtime instead (eg, dynamic typing)*
scripting language: *"middle-weight"*

Python is an open source project which is maintained by a large and very active community.

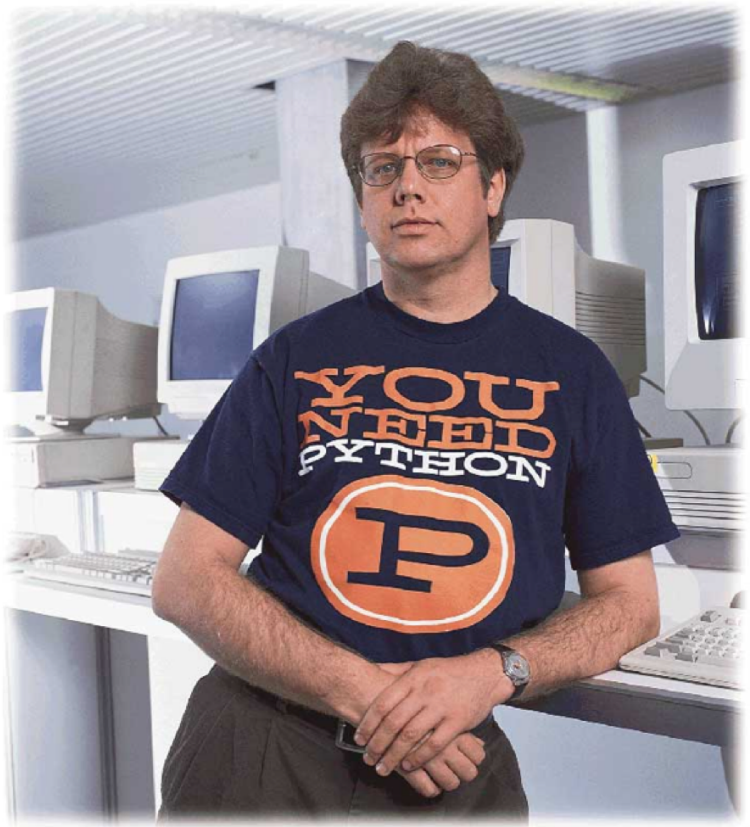
Python is an open source project which is maintained by a large and very active community.

*It was originally created by Guido Van Rossum in the 1990s, who currently holds the title of Benevolent Dictator For Life (***BDFL***).*

GUIDO

http://en.wikipedia.org/wiki/Guido_van_Rossum

GUIDO: THE EARLY YEARS



http://pytalent.zandstrasystems.com/meet_py1.html

The presence of a BDFL means that Python has a unified design philosophy.

The presence of a BDFL means that Python has a unified design philosophy.

This design philosophy emphasizes readability and ease of use, and is codified in PEP8 (the Python style guide) and PEP20 (the Zen of Python).

The presence of a BDFL means that Python has a unified design philosophy.

This design philosophy emphasizes readability and ease of use, and is codified in PEP8 (the Python style guide) and PEP20 (the Zen of Python).

NOTE

PEPs (or Python Enhancement Proposals) are the public design specs that the language follows.

II. PYTHON STRENGTHS & WEAKNESSES

STRENGTHS & WEAKNESSES

Python's popularity comes from the strength of its design.

The syntax looks like pseudocode, and it is explicitly meant to be clear, compact, and easy to read.

This is usually summarized by saying Python is an **expressive** *language.*

Another great strength is the **Python Standard Library**.

This is a collection of packages that ships with the standard Python distrubution, and "...covers everything from asynchronous processing to zip files".

The advantages of the PSL are usually described by saying that Python comes with **batteries included**.

STRENGTHS & WEAKNESSES

Ultimately, Python's most important strength is that it's easy to learn and easy to use.

STRENGTHS & WEAKNESSES

Ultimately, Python's most important strength is that it's easy to learn and easy to use.

Because there should be only one way to perform a given task, things frequently work the way you expect them to.

STRENGTHS & WEAKNESSES

Q: Python sounds amazing. What is it bad at?

STRENGTHS & WEAKNESSES

Q: Python sounds amazing. What is it bad at?

For one thing, Python is slower than a lower-level language (but keep in mind that this is a conscious tradeoff).

IPython Lecture

III. PANDAS

Pandas

Built by a quantitative trader at a firm called AQR.

Had certain requirements:

- Data structures with labeled axes
- Integrated time series functionality
- Same data structure to handle both time series and non-time series
- Arithmetic operations and reductions (like summations across axes)
- Flexible handling of missing data
- Merge and other relational operations found in SQL like databases

Pandas

Pandas is:

- A library that allows you to work with structured data quickly and easily
- Very fast. Combines high performance features of numpy with flexible data manipulation capabilities of spreadsheets and databases
- Heavily influenced by the R programming language

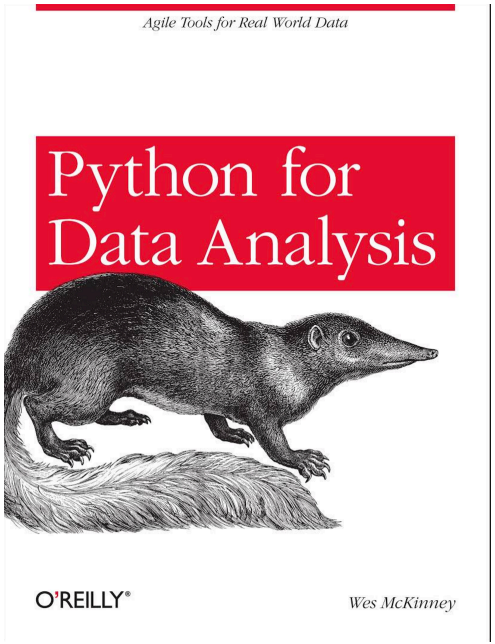
Pandas – Data Structures

Series	DataFrame
One dimensional – just an array of data, and an associated array of labels, called an *index*	Represents tabular spreadsheet-like data structure containing a collection of columns, each of which can be a different value type. It can be thought of as a dict of Series

Pandas – Data Structures

Pandas has some incredible functionality (as we'll soon see).

Becoming great with pandas is a huge step towards becoming great with data manipulation



IV. PANDAS IPTYHON LECTURE

V. EXIT TICKET

TODAY

- Deep dive into python and programming tools
- Described the programming landscape
- Created our first iPython Notebook from scratch became familiar with how to work with data in python with the pandas library
- Preview of plotting and time series