Next steps

Eric Franzosa (franzosa@hsph.harvard.edu)

Kevin Bonham (kbonham@broadinstitute.org)

http://franzosa.net/bst273

Learning to Program

- Why do it?
  - Make easy tasks easy
  - Make hard tasks possible
  - Improve accuracy and efficiency in your work
  - It’s empowering!
- What does it take?
  - Learn to identify problems that computers can solve
  - Learn to describe those problems in a way that computers can understand
  - *Learn a programming language to translate those descriptions into code*

How to keep learning

- Take additional courses (we'll talk about a few)
- Read additional books (we'll talk about a few)
- Read/watch videos online (we'll suggest some places to look)
- ***Practice, practice, practice***

Learning through coursework

- A few different types of courses will be accessible with your new coding skills
- Computer science
  - Theory of computing, algorithms, data structures
  - Practical applications: why is `'Bob' in dict` faster than `"Bob" in list`?
- Software engineering
  - Best practices for making code that will be used more than once
  - Documenting, testing, working as a team
- Applied computing
  - Using computers to solve practical problems
  - Bioinformatics, statistical computing, data science

Classes at HSPH

- BST 267: Introduction to Social and Biological Networks
  - Fall 2 with Jukka-Pekka Onnela
  - Uses the Python NetworkX module
- BST 262: Computing for Big Data
  - Fall 2 with Christine Choirat
  - Methods and best practices for programming against big data (in R or Python)
- BST 234: Introduction to Data Structures and Algorithms
  - Spring with Christoph Lange and Curtis Huttenhower
  - Data structures and computer algorithms for statistical computing

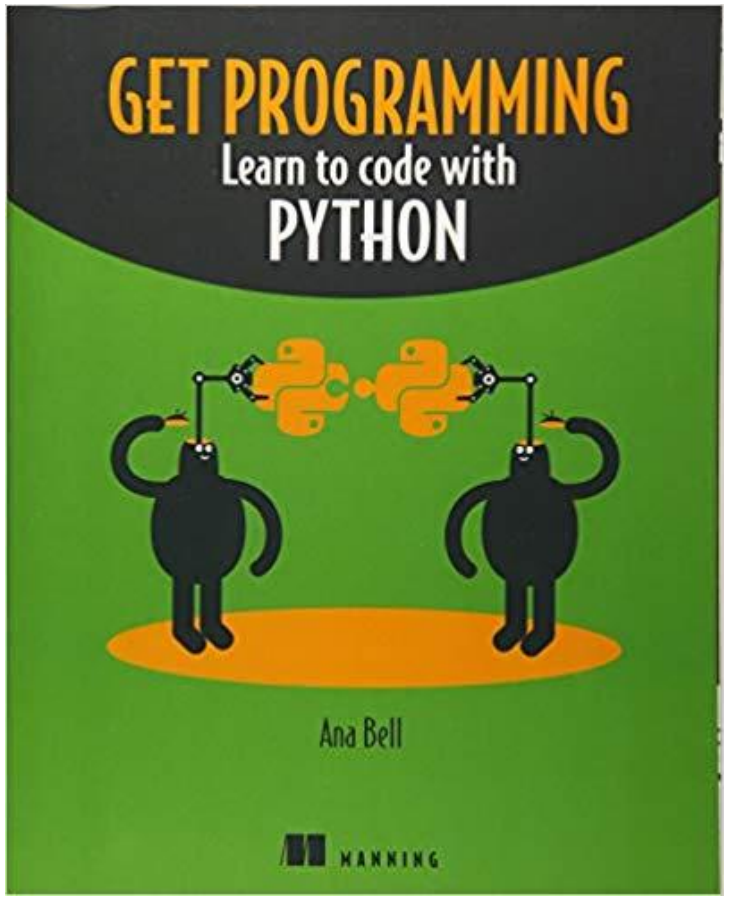
Classes at HSPH

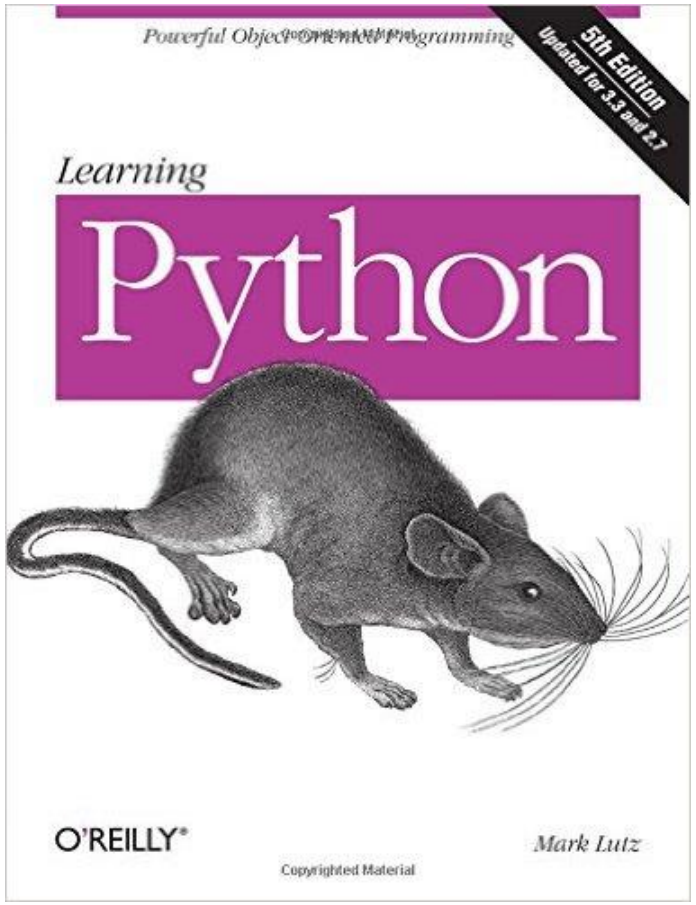
- BST 281: Genomic Data Manipulation
  - Spring with Curtis Huttenhower and Eric Franzosa
  - Methods for studying high-throughput molecular biological data
  - In-class Python activities with Jupyter notebooks

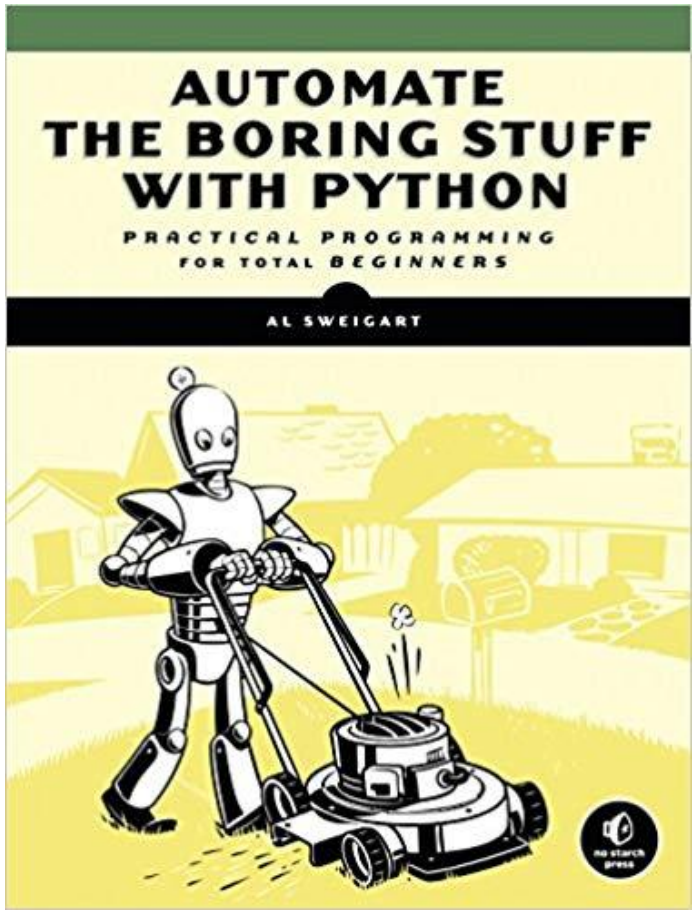
Classes outside of HSPH

- CS 50: Introduction to Computer Science (Harvard University)
  - Very broad introduction to topics in computer science
  - Explores facets of a number of different programming languages, including Python
  - Also available online (via Edx)
- 6.009: Fundamentals of Programming (MIT)
  - Offered in Fall and Spring
  - Expands on 6.0001, Intro to Programming in Python (~this course)

Books


GET PROGRAMMING
Learn to code with
PYTHON
Ana Bell
MANNING


Powerful Object-Oriented Programming
5th Edition
Updated for 3.3 and 2.7
Learning
Python
O'REILLY
Mark Lutz
Copyrighted Material


AUTOMATE
THE BORING STUFF
WITH PYTHON
PRACTICAL PROGRAMMING
FOR TOTAL BEGINNERS
AL SWEIGART
no starch press

Online materials

- https://learnpythonthehardway.org/
  - Another online textbook
- http://www.learnpython.org/
  - Interactive Python tutorials (similar to our Juptyer notebooks)
- https://www.reddit.com/r/learnpython/
  - A subreddit devoted to learning Python in particular
- https://stackoverflow.com/
  - Questions and answers for computing and programming
- https://www.youtube.com/user/Computerphile
  - Videos on all sorts of topics in computing

Practice, Practice, Practice

- The best way to keep developing coding skills is to keep using them
- If you encounter a computing problem, try to solve it with Python
  - Works especially well for tasks in data analysis or organization
  - Or anything where you think “I wish I could automate this”
- When you get stuck, research the problem online
- Once you’re over the initial learning curve, this is the best way to learn

thanks.py


Curtis Huttenhower
Jason Lloyd-Price
Melanie Schirmer
George Weingart
Cesar Arze
Jeremy Wilkinson
Lauren McIver
Anahit Mailyan
Kelsey Thompson
Himel Mallick
Yan Yan
Kevin Bonham
Yancong Zhang
Brantley Hall
Tommi Vatanen
Dmitry Shungin
Long Nguyen
Siyuan Ma
Emma Accorsi
Shimin Bi
Jingjing Tang
Casey DuLong
Hera Vlamakis

Websites that will *give you* problems to solve

- http://www.pythonchallenge.com/
  - Old, but very Python-focused
- https://projecteuler.net/
  - Math puzzles that require coding to solve
- http://rosalind.info/
  - Bioinformatics problems that require coding to solve
- Advent of code

Meaning vs Syntax - reprise

Overview

- A programming "language" is really a translation
  - Human intent --> machine code
- There are two basic features in any language:
  - Data (information)
  - Instructions (actions)
- The concepts you have learned in the course are extensible to **any** programming language
  - Think in terms of inputs and outputs
  - Don’t repeat yourself (write functions!)
  - Pay attention to error messages (and google!)

Write functions

Python

```
def weird_addition(number1, number2):
    result = 2 * (number1 + number2)
    return result
```

R

```
weird_addition <- function(number1,number2){
    result <- 2 * (number1 + number2)
    return(result)
}
```

Julia

```
function weird_addition(number1, number2)
    result = 2 * (number1 + number2)
    return result
end
```

Use code that others have written

Python

```
import pandas as pd
import argparse
```

R

```
library("dyplr")
library("argparse")
```

Julia

```
using DataFrames
using ArgParse
```

Use the REPL

Python

```
>>> x = "look at me, I'm a string!”
>>> x
"look at me, I'm a string!”
>>> 5 ** 2
25
```

R

```
> x <- ”look at me, I'm a string!”
> x
[1] ”look at me, I'm a string!”
> 5 ^ 2
[1] 25
```

Julia

```
julia> x = ”look at me, I'm a string!”
”look at me, I'm a string!”
julia> 5 ^ 2
25
```

Use collections

Python

```
d = {"apple": "green", "banana": "yellow", "orange": "orange"}
d["banana"] # "yellow”
l = [1, 1.2, "a"]
l[1] # 1.2
```

R

```
l = c(1, 1.2, "a")
l[2] # "1.2"
```

Julia

```
d = Dict("apple"=> "green", "banana"=> "yellow", "orange"=> "orange")
d["banana"] # "yellow”
l = [1, 1.2, "a"]
l[2] # 1.2
```

Read (and google) the error messages

Python

```
>>> from math import sqrt
>>> sqrt(-2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: math domain error
```

R

```
> sqrt(-2)
[1] NaN
Warning message:
In sqrt(-2) : NaNs produced
```

Julia

```
julia> sqrt(-2)
ERROR: DomainError with -2.0:
sqrt will only return a complex result if called with a complex argument.
Try sqrt(Complex(x)).
Stacktrace:
 [1] throw_complex_domainerror(::Symbol, ::Float64) at ./math.jl:31
 [2] sqrt at ./math.jl:479 [inlined]
 [3] sqrt(::Int64) at ./math.jl:505
 [4] top-level scope at none:0
```