Introduction to Coding with Python

Smithsonian Institution
Paul Frandsen (FrandsenP@si.edu)
Michael Trizna (TriznaM@si.edu)

Let’s take a survey!

http://goo.gl/forms/2NpUuiYLz6SYxJt53

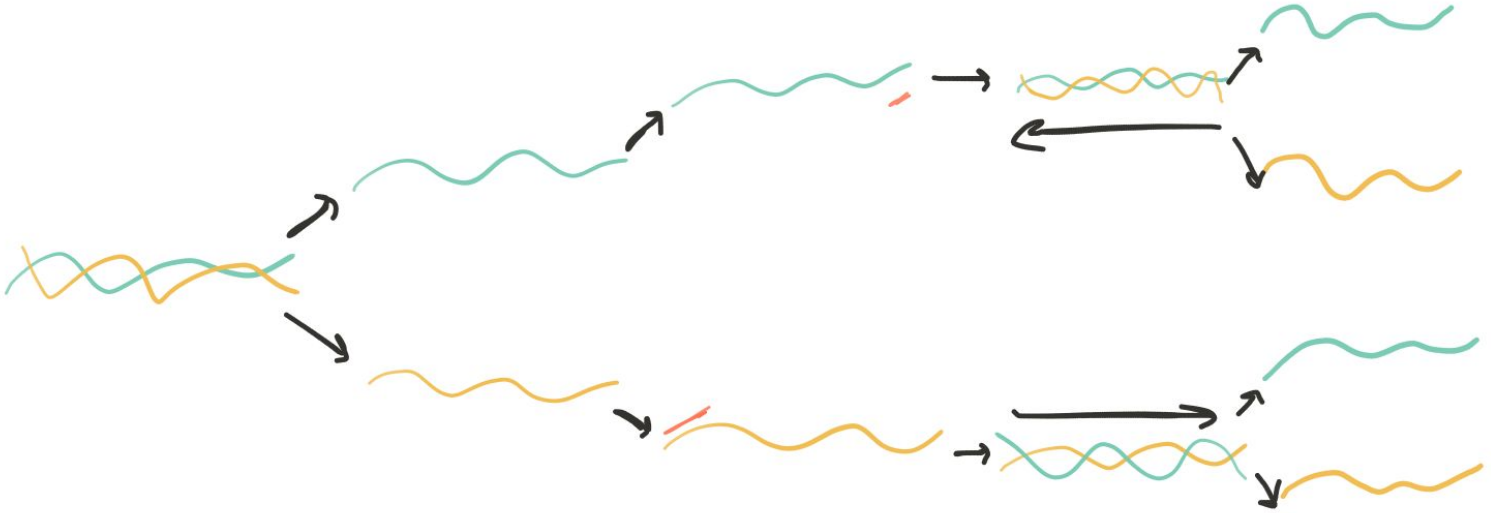
Introduction

- Rationale
- Goals
- What is coding?

Rationale

- More than ever, biology is becoming a computational field
- Datasets are reaching an unprecedented size

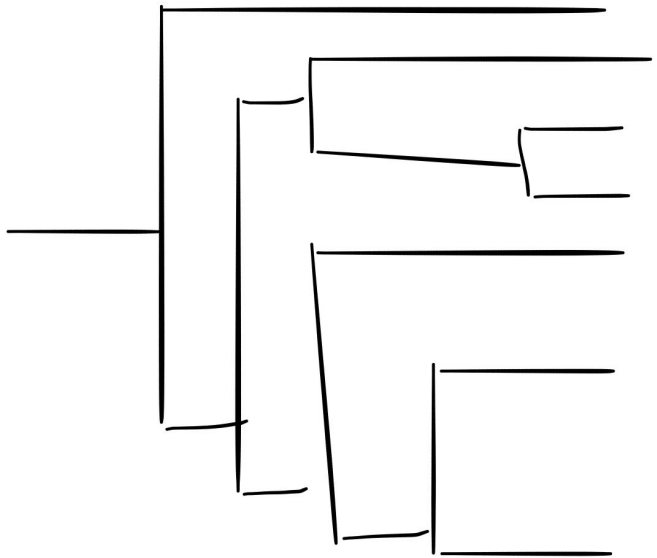
A personal example:



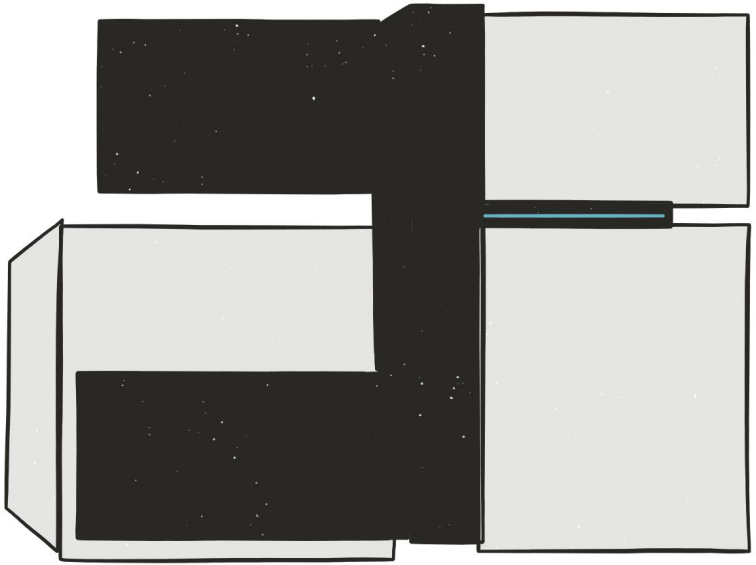
A personal example:

A personal example:

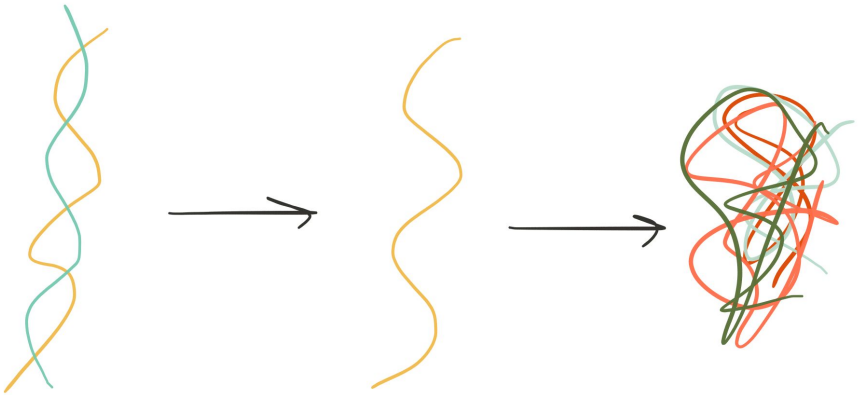
A personal example:



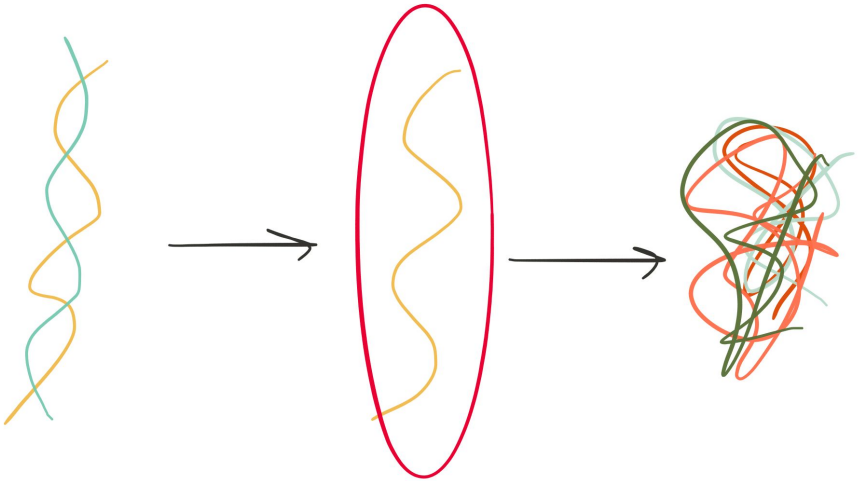
A personal example:



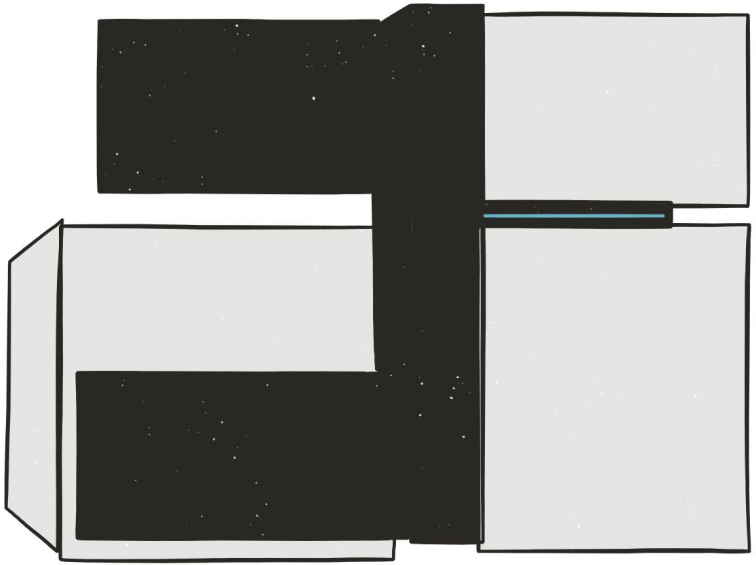
A personal example:



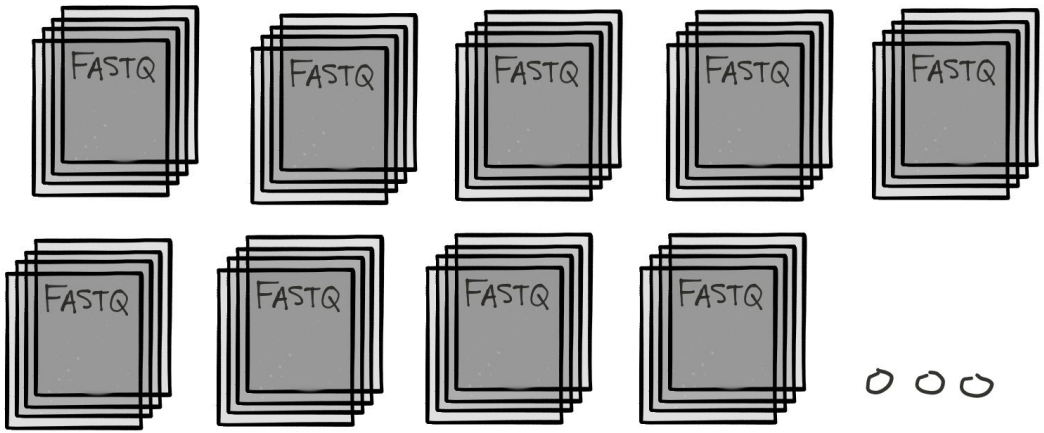
A personal example:



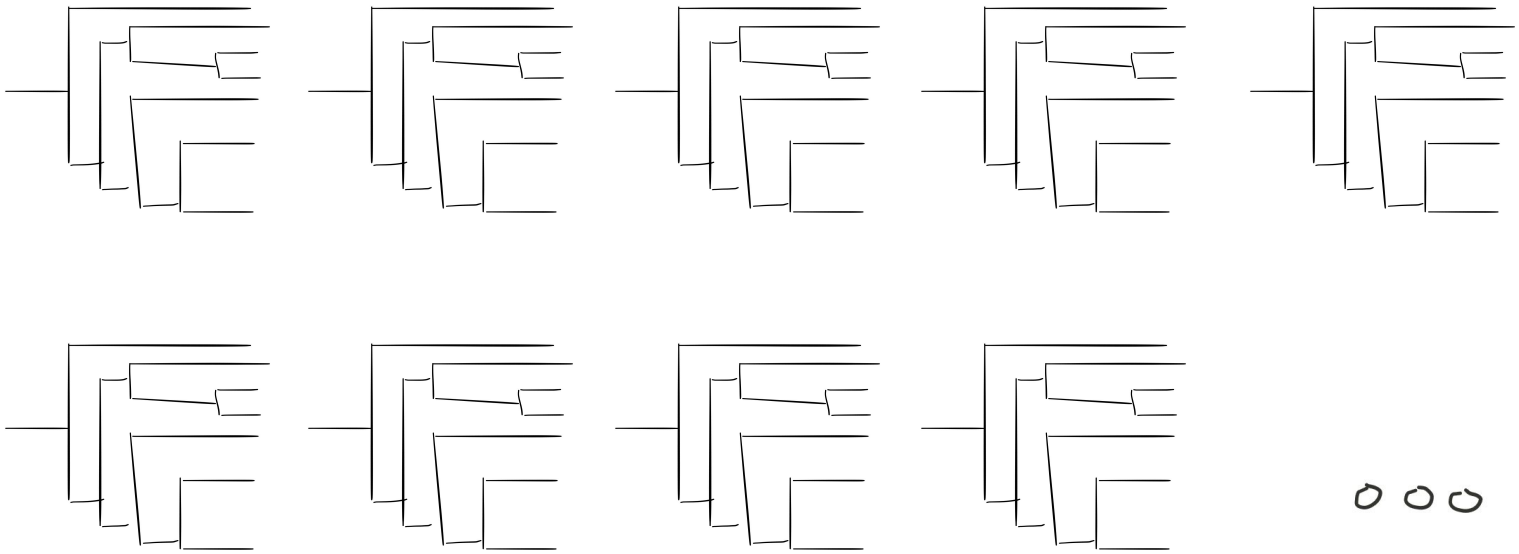
A personal example:



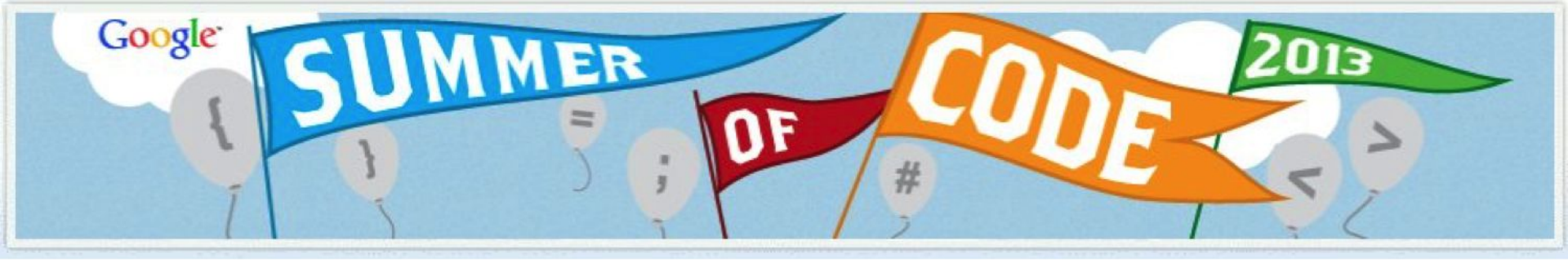
A personal example:



A personal example:



A personal example:


Google
SUMMER
OF
CODE
2013

Google

Goals

- Introduce the basics behind coding and computational thinking
- Give a firm grounding in basic Python syntax and functionality
- Disambiguate confusing computer science terms
- Help participants ‘cross the hump’ into being confident enough to try coding

What is coding?

- Coding is a hip way to refer to the “implementation portion” of computer programming (programmers will argue that programming is much more than simply writing code)
- One of the pioneers of coding was Alan Turing who showed that any computational problem could be solved by his ‘Turing machine’
- In a similar manner, transistors on a microchip can take simple instructions (on or off) to do complicated computations
- Coding is the process of writing those instructions, most often in an abstract way through ‘code’, which is then translated into machine instructions for the microprocessor to compute

* If you find this sort of thing fascinating, a good, readable history can be found in Walter Isaacson’s “**The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution**”

What is Python?

- Interpreted language:
  - You write code, and Python "interpreter" converts this into machine commands each time you run your program
  - Pros:
    - Generally much faster to write, and easier to read
  - Cons:
    - Generally does not run as fast, and harder to package and distribute by itself
  - Examples:
    - Javascript, Ruby, Perl, PHP, R
- vs. Compiled language:
  - Converted into machine commands once, but this process take a long time
  - Examples:
    - C, Java, Go

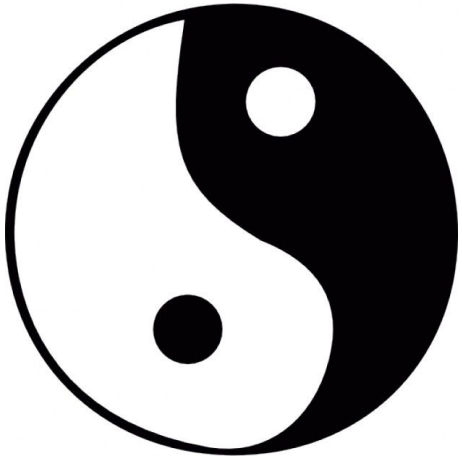
History of Python

Created by Guido van Rossum in Netherlands in the late 1980s

Now called the "Benevolent Dictator for Life"

Name of Python comes from Monty Python

Why Python?



Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one -- obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.

Why Python?

It's easy to read

Java

```java
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}

Or also:

     class HelloWorldApp {
public static void main(String[] args) {
  System.out.println("Hello World!");     }
    }
```

Python

```python
print("Hello World!")
```

Why Python?

Universal

Mac OS X and most Linux operating systems come with Python pre-installed.

MicroPython is a project to create microcontrollers that run Python by default.

This will make Python a major part of the "Internet of Things"

Why Python?

Not just for scripting:

- Web Frameworks (like Django and Flask)

- Sphinx for documentation (readthedocs.org)

- GUI programs like Dropbox

Why Python?

"Batteries included"

The Python "Standard Library" comes with over 300 packages already available by default.

For example: csv, xml, ftp

Full list: https://docs.python.org/3/library/

Why Python?

Great external package ecosystem:

- Pip and conda
- Especially in scientific community:
  - Biology-specific packages
  - Variety of plotting packages
  - Scikit-learn: the industry standard in machine learning
  - Several GIS packages

List of scientific packages here: https://www.scipy.org/topical-software.html#topic-guides-organized-by-scientific-field

Python 2 vs Python 3

- Python 3.0 released in 2008
- Small changes, but **backwards-incompatible**
- Python 2 -- although still widely used -- will stop being supported in 2020
  - Pythonclock.org has a countdown
- Well-written code can still work on both Python 2 and 3