Why Python?

Joseph Thomas

University of Arizona
Department of Mathematics

November 5, 2012

What is Python?

Python

What is Python?

Python

- ... is an interpreted programming language.

What is Python?

Python

- … is an interpreted programming language.
- … was invented around 1991.

What is Python?

Python

- ... is an interpreted programming language.
- ... was invented around 1991.
- ... is intended to emphasize code readability.

What is Python?

Python

- … is an interpreted programming language.
- … was invented around 1991.
- … is intended to emphasize code readability.
- … takes ideas from several programming *paradigms*

What is Python?

Python

- … is an interpreted programming language.
- … was invented around 1991.
- … is intended to emphasize code readability.
- … takes ideas from several programming *paradigms*

What is Python?

Python

- ... is an interpreted programming language.
- ... was invented around 1991.
- ... is intended to emphasize code readability.
- ... takes ideas from several programming *paradigms*
  - procedural languages (like C)

What is Python?

Python

- ... is an interpreted programming language.
- ... was invented around 1991.
- ... is intended to emphasize code readability.
- ... takes ideas from several programming *paradigms*
  - procedural languages (like C)
  - object oriented languages (like Java)

What is Python?

Python

- ... is an interpreted programming language.
- ... was invented around 1991.
- ... is intended to emphasize code readability.
- ... takes ideas from several programming *paradigms*
    - procedural languages (like C)
    - object oriented languages (like Java)
    - functional languages (like Lisp, Haskell)

What is Python?

Python

- ... is an interpreted programming language.
- ... was invented around 1991.
- ... is intended to emphasize code readability.
- ... takes ideas from several programming *paradigms*
    - procedural languages (like C)
    - object oriented languages (like Java)
    - functional languages (like Lisp, Haskell)
- ... is used by many industries (Google, etc.).

Why Python?

For the mathematical researcher, a software tool should:

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.
- Allow you to **naturally** convert math into code.

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.
- Allow you to **naturally** convert math into code.
- Example: Is a (mathematical) function

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.
- Allow you to **naturally** convert math into code.
- Example: Is a (mathematical) function

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.
- Allow you to **naturally** convert math into code.
- Example: Is a (mathematical) function
  - A procedure call?

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.
- Allow you to **naturally** convert math into code.
- Example: Is a (mathematical) function
  - A procedure call?
  - An array?

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.
- Allow you to **naturally** convert math into code.
- Example: Is a (mathematical) function
  - A procedure call?
  - An array?
  - A hash map / dictionary / associative array?

Why Python?

For the mathematical researcher, a software tool should:

- Let you implement an idea fast.
- Avoid reinventing the wheel.
- Allow you to **naturally** convert math into code.
- Example: Is a (mathematical) function
  - A procedure call?
  - An array?
  - A hash map / dictionary / associative array?
  - A set of tuples?

Why Python?

Python is a useful language to know if you want to...

Why Python?

Python is a useful language to know if you want to...

- quickly (and correctly) implement an abstract idea.

Why Python?

Python is a useful language to know if you want to...

- quickly (and correctly) implement an abstract idea.
- bring together many different libraries (which might not be in Python) to do something new.

Why Python?

Python is a useful language to know if you want to...

- quickly (and correctly) implement an abstract idea.
- bring together many different libraries (which might not be in Python) to do something new.
- easily link your code into a computer algebra system (namely Sage).

Why Python?

Python is a useful language to know if you want to...

- quickly (and correctly) implement an abstract idea.
- bring together many different libraries (which might not be in Python) to do something new.
- easily link your code into a computer algebra system (namely Sage).
- process data (particularly text) and/or crawl the internet.

Example: Bayesian Spam Filtering

Example: Bayesian Spam Filtering

- Q: Can we detect a statistical difference between the emails written by spammers and regular people?

Example: Bayesian Spam Filtering

- Q: Can we detect a statistical difference between the emails written by spammers and regular people?
- Fundamental Problem: Given a word w, estimate the probability w comes from a spam email.

Example: Bayesian Spam Filtering

- Q: Can we detect a statistical difference between the emails written by spammers and regular people?
- Fundamental Problem: Given a word w, estimate the probability w comes from a spam email.
- Context: Use Bayes' Theorem to estimate the probability an email is spam, based upon its words.

Example: Bayesian Spam Filtering

- Q: Can we detect a statistical difference between the emails written by spammers and regular people?
- Fundamental Problem: Given a word w, estimate the probability w comes from a spam email.
- Context: Use Bayes' Theorem to estimate the probability an email is spam, based upon its words.
- I made a *lot* of assumptions. Do I still have a useful approximation?

Example: Bayesian Spam Filtering

We need to do an experiment!

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\ \text{appearances of word } w \text{ in the corpus}}{\#\ \text{words in the corpus}}$$

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Programming Problem:

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Programming Problem:

- Input: A bunch of emails (text).

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Programming Problem:

- Input: A bunch of emails (text).
- Output: A dictionary mapping words to the conditional probability that they appear in an email from the spam corpus.

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Programming Problem:

- Input: A bunch of emails (text).
- Output: A dictionary mapping words to the conditional probability that they appear in an email from the spam corpus.
- Sub-Problems:

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Programming Problem:

- Input: A bunch of emails (text).
- Output: A dictionary mapping words to the conditional probability that they appear in an email from the spam corpus.
- Sub-Problems:
  - Reading a file (get a big string).

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Programming Problem:

- Input: A bunch of emails (text).
- Output: A dictionary mapping words to the conditional probability that they appear in an email from the spam corpus.
- Sub-Problems:
  - Reading a file (get a big string).
  - Break the text of the file into pieces.

Example: Bayesian Spam Filtering

We need to do an experiment!

- Get a collection of emails, classified into spam and not-spam.
- Estimate $P(w \text{ appears in a spam email})$ by

$$\frac{\#\text{ appearances of word } w \text{ in the corpus}}{\#\text{ words in the corpus}}$$

Programming Problem:

- Input: A bunch of emails (text).
- Output: A dictionary mapping words to the conditional probability that they appear in an email from the spam corpus.
- Sub-Problems:
  - Reading a file (get a big string).
  - Break the text of the file into pieces.
  - Building a dictionary data structure.

Python Code

Input: A file F of emails.
Output: A mapping of words to conditional probabilities.

```
W = open("emails.txt").read().split()
D = {}

for w in W:
    if w not in D:
        D[w] = 1
    else:
        D[w] = D[w] + 1

for w in D:
    D[w] = D[w]/len(W)
```

Example: Bayesian Spam Filtering — Java versus Python

Python Solution: 93 lines of code.
Java Solution: 321 lines of code.
Why the difference?

Example: Bayesian Spam Filtering — Java versus Python

Python Solution: 93 lines of code.
Java Solution: 321 lines of code.
Why the difference?

- Different languages have different goals!

Example: Bayesian Spam Filtering — Java versus Python

Python Solution: 93 lines of code.
Java Solution: 321 lines of code.
Why the difference?

- Different languages have different goals!
- Python: Get a small prototype working quickly.

Example: Bayesian Spam Filtering — Java versus Python

Python Solution: 93 lines of code.
Java Solution: 321 lines of code.
Why the difference?

- Different languages have different goals!
- Python: Get a small prototype working quickly.
- Java: Build **huge** applications from organized code.

Example: Bayesian Spam Filtering — Java versus Python

Python Solution: 93 lines of code.
Java Solution: 321 lines of code.
Why the difference?

- Different languages have different goals!
- Python: Get a small prototype working quickly.
- Java: Build **huge** applications from organized code.
- Matlab/Mathematica/GAP: Make working with math easy (without necessarily providing a lot of CS primitives).

Example: Bayesian Spam Filtering — Java versus Python

Python Solution: 93 lines of code.
Java Solution: 321 lines of code.
Why the difference?

- Different languages have different goals!
- Python: Get a small prototype working quickly.
- Java: Build **huge** applications from organized code.
- Matlab/Mathematica/GAP: Make working with math easy (without necessarily providing a lot of CS primitives).

Example: Bayesian Spam Filtering — Java versus Python

Python Solution: 93 lines of code.
Java Solution: 321 lines of code.
Why the difference?

- Different languages have different goals!
- Python: Get a small prototype working quickly.
- Java: Build **huge** applications from organized code.
- Matlab/Mathematica/GAP: Make working with math easy (without necessarily providing a lot of CS primitives).

Moral: If you want to know "Is my theory correct?", maybe code-correctness and ease of implementation matter more than speed and maintainability.

Python, Sage, and Visualization

Sage, a computer algebra system from the University of Washington, is written in Python.

Python, Sage, and Visualization

Sage, a computer algebra system from the University of Washington, is written in Python.

- The project takes advantage of *a lot* of good software engineering ideas, particularly when it comes to documentation.

Python, Sage, and Visualization

Sage, a computer algebra system from the University of Washington, is written in Python.

- The project takes advantage of *a lot* of good software engineering ideas, particularly when it comes to documentation.
- Unlike Mathematica, GAP, and Matlab, there is basically no distinction between Python code and Sage code.

Python, Sage, and Visualization

Sage, a computer algebra system from the University of Washington, is written in Python.

- The project takes advantage of *a lot* of good software engineering ideas, particularly when it comes to documentation.
- Unlike Mathematica, GAP, and Matlab, there is basically no distinction between Python code and Sage code.
- Theme: Prototype in SAGE, then move polished code out into python files.

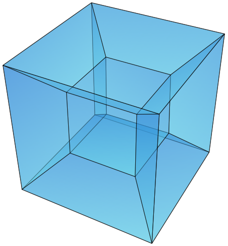
Python, Sage, and Visualization

Sage, a computer algebra system from the University of Washington, is written in Python.

- The project takes advantage of *a lot* of good software engineering ideas, particularly when it comes to documentation.
- Unlike Mathematica, GAP, and Matlab, there is basically no distinction between Python code and Sage code.
- Theme: Prototype in SAGE, then move polished code out into python files.
- Let's see an example!

Python, Sage, and Visualization

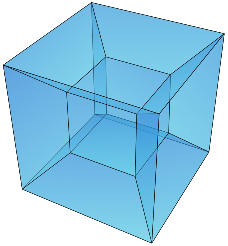
Common Geometry Problem: Visualizing examples in $\mathbb{R}^3$.
Example: Consider a *hyper-rectangular prism*.



Up to scaling, a hyper-rectangular prism is specified by 3 positive lengths (x, y, z).

Python, Sage, and Visualization

Common Geometry Problem: Visualizing examples in $\mathbb{R}^3$.
Example: Consider a *hyper-rectangular prism*.



Up to scaling, a hyper-rectangular prism is specified by 3 positive lengths (x, y, z).

Research Problem: Investigate geodesics on this space, based on (x, y, z).

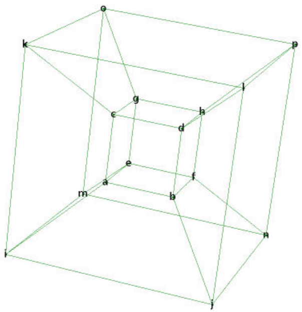
Hyper-rectangular Prism

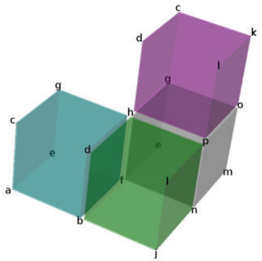
Calculation Problem:

Hyper-rectangular Prism

Calculation Problem:

- Easy: How does a geodesic move through **one** cell?



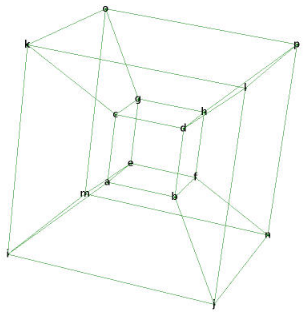


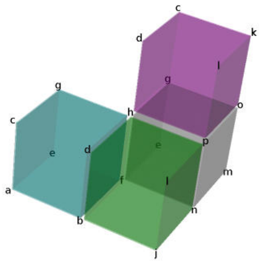
A “tumbling” of the hyper-rectangular prism.

Hyper-rectangular Prism

Calculation Problem:

- Easy: How does a geodesic move through **one** cell?
- Hard: Tracking how all eight cells are connected.



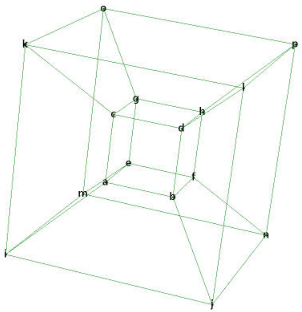


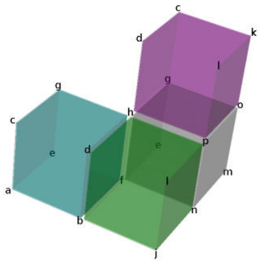
A “tumbling” of the hyper-rectangular prism.

Hyper-rectangular Prism

Calculation Problem:

- Easy: How does a geodesic move through **one** cell?
- Hard: Tracking how all eight cells are connected.
- Given a sequence of cells, is there a geodesic through them?



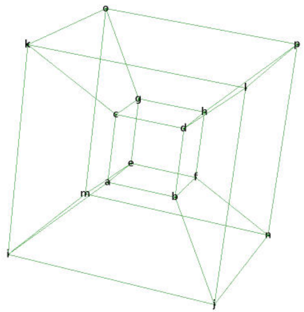


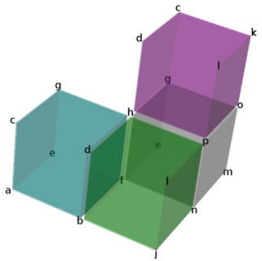
A “tumbling” of the hyper-rectangular prism.

Hyper-rectangular Prism

Calculation Problem:

- Easy: How does a geodesic move through **one** cell?
- Hard: Tracking how all eight cells are connected.
- Given a sequence of cells, is there a geodesic through them?
- I want to visualize this with a tumbling!





A “tumbling” of the hyper-rectangular prism.

Drawing Tumblings

A tumbling is just a sequence of adjacent rectangular prisms in $\mathbb{R}^3$.
We'll use `JMol` to visualize it.
I want to show different things in different situations:

Drawing Tumblings

A tumbling is just a sequence of adjacent rectangular prisms in $\mathbb{R}^3$.
We'll use `JMol` to visualize it.
I want to show different things in different situations:

- The labels we put on the vertices/edges.

Drawing Tumblings

A tumbling is just a sequence of adjacent rectangular prisms in $\mathbb{R}^3$.
We'll use `JMol` to visualize it.
I want to show different things in different situations:

- The labels we put on the vertices/edges.
- Only certain labels on the vertices.

Drawing Tumblings

A tumbling is just a sequence of adjacent rectangular prisms in $\mathbb{R}^3$.
We'll use `JMol` to visualize it.
I want to show different things in different situations:

- The labels we put on the vertices/edges.
- Only certain labels on the vertices.
- Particular colors of the cells.

Drawing Tumblings

A tumbling is just a sequence of adjacent rectangular prisms in $\mathbb{R}^3$.
We'll use `JMol` to visualize it.
I want to show different things in different situations:

- The labels we put on the vertices/edges.
- Only certain labels on the vertices.
- Particular colors of the cells.

Drawing Tumblings

A tumbling is just a sequence of adjacent rectangular prisms in $\mathbb{R}^3$.
We'll use `JMol` to visualize it.
I want to show different things in different situations:

- The labels we put on the vertices/edges.
- Only certain labels on the vertices.
- Particular colors of the cells.

How can I get all of this in one procedure call?

```
def drawCube( eCube,
              showEdgesTuple=(False,False,False),
              showVertexLabelPred=(lambda vert: True),
              colorCube=False ):
```

Using Libraries

What if I need to do [Computationally Intensive Process P] ?

Using Libraries

What if I need to do [Computationally Intensive Process P] ?
Anecdote:

Using Libraries

What if I need to do [Computationally Intensive Process P] ?
Anecdote:

- I once studied a bio-informatics problem in which I needed to solve a big, messy linear program at run-time.

Moral: Sometimes it's better to **specify** your problem in a descriptive language, **then** hand it over to a C library for solving.

Using Libraries

What if I need to do [Computationally Intensive Process *P*] ?
Anecdote:

- I once studied a bio-informatics problem in which I needed to solve a big, messy linear program at run-time.
- Linear programming packages are usually written in C.

Moral: Sometimes it's better to **specify** your problem in a descriptive language, **then** hand it over to a C library for solving.

Using Libraries

What if I need to do [Computationally Intensive Process *P*] ?
Anecdote:

- I once studied a bio-informatics problem in which I needed to solve a big, messy linear program at run-time.
- Linear programming packages are usually written in C.
- **Bad Assumption:** If I need a C library, my whole program must be in C.

Moral: Sometimes it's better to **specify** your problem in a descriptive language, **then** hand it over to a C library for solving.

Using Libraries

What if I need to do [Computationally Intensive Process P] ?
Anecdote:

- I once studied a bio-informatics problem in which I needed to solve a big, messy linear program at run-time.
- Linear programming packages are usually written in C.
- **Bad Assumption:** If I need a C library, my whole program must be in C.
- Price of this assumption: $\sim$ 3000 lines of hard-to-modify C code. (Replaced by $\sim$ 600 lines of easy Python code.)

Moral: Sometimes it's better to **specify** your problem in a descriptive language, **then** hand it over to a C library for solving.

Libraries

Libraries

- **Numpy** : Linear algebra, big arrays, Fourier analysis, etc.

Libraries

- **Numpy** : Linear algebra, big arrays, Fourier analysis, etc.
- **cvxopt/PyGLPK** : Linear programming solvers.

Libraries

- **Numpy** : Linear algebra, big arrays, Fourier analysis, etc.
- **cvxopt/PyGLPK** : Linear programming solvers.
- **Cython** : Connect *your* C/C++ code to Python.

Libraries

- **Numpy** : Linear algebra, big arrays, Fourier analysis, etc.
- **cvxopt/PyGLPK** : Linear programming solvers.
- **Cython** : Connect *your* C/C++ code to Python.

Libraries

- **Numpy** : Linear algebra, big arrays, Fourier analysis, etc.
- **cvxopt/PyGLPK** : Linear programming solvers.
- **Cython** : Connect *your* C/C++ code to Python.
  - Call C/C++ from Python

Libraries

- **Numpy** : Linear algebra, big arrays, Fourier analysis, etc.
- **cvxopt/PyGLPK** : Linear programming solvers.
- **Cython** : Connect *your* C/C++ code to Python.
  - Call C/C++ from Python
  - Call Python from C/C++

Libraries

- **Numpy** : Linear algebra, big arrays, Fourier analysis, etc.
- **cvxopt/PyGLPK** : Linear programming solvers.
- **Cython** : Connect *your* C/C++ code to Python.
  - Call C/C++ from Python
  - Call Python from C/C++
  - Pass data between languages (without writing/parsing text files).

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Question: What would the "genealogical tree" of the department look like?

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Question: What would the "genealogical tree" of the department look like?

Problems:

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Question: What would the "genealogical tree" of the department look like?

Problems:

- We can find out, for $\geq \$150 +$ (Shipping and Handling).

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Question: What would the "genealogical tree" of the department look like?

Problems:

- We can find out, for $\geq \$150 +$ (Shipping and Handling).
- The MGP website doesn't offer all the data in a convenient text file.

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Question: What would the "genealogical tree" of the department look like?

Problems:

- We can find out, for $\geq \$150 +$ (Shipping and Handling).
- The MGP website doesn't offer all the data in a convenient text file.
- We could do all the work by hand (a lot of labor).

Example: Gluing Code

The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Question: What would the "genealogical tree" of the department look like?

Problems:

- We can find out, for $\geq \$150 +$ (Shipping and Handling).
- The MGP website doesn't offer all the data in a convenient text file.
- We could do all the work by hand (a lot of labor).

Example: Gluing Code

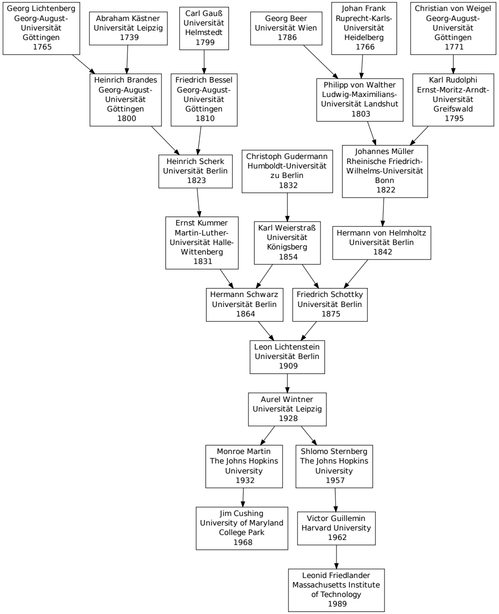
The Mathematics Genealogy Project catalogs people who received PhD's in math and data on their advisor(s).
All of this data is posted to the web.

Question: What would the "genealogical tree" of the department look like?

Problems:

- We can find out, for $\geq \$150 +$ (Shipping and Handling).
- The MGP website doesn't offer all the data in a convenient text file.
- We could do all the work by hand (a lot of labor).

Python Solution: ~ 300 lines of code.


Georg Lichtenberg Georg-August-Universität Göttingen 1765
Abraham Kästner Universität Leipzig 1739
Carl Gauß Universität Helmstedt 1799
Georg Beer Universität Wien 1786
Johan Frank Ruprecht-Karls-Universität Heidelberg 1766
Christian von Weigel Georg-August-Universität Göttingen 1771
Heinrich Brandes Georg-August-Universität Göttingen 1800
Friedrich Bessel Georg-August-Universität Göttingen 1810
Philipp von Walther Ludwig-Maximilians-Universität Landshut 1803
Karl Rudolphi Ernst-Moritz-Arndt-Universität Greifswald 1795
Heinrich Scherk Universität Berlin 1823
Christoph Gudermann Humboldt-Universität zu Berlin 1832
Johannes Müller Rheinische Friedrich-Wilhelms-Universität Bonn 1822
Ernst Kummer Martin-Luther-Universität Halle-Wittenberg 1831
Karl Weierstraß Universität Königsberg 1854
Hermann von Helmholtz Universität Berlin 1842
Hermann Schwarz Universität Berlin 1864
Friedrich Schottky Universität Berlin 1875
Leon Lichtenstein Universität Berlin 1909
Aurel Wintner Universität Leipzig 1928
Monroe Martin The Johns Hopkins University 1932
Shlomo Sternberg The Johns Hopkins University 1957
Jim Cushing University of Maryland College Park 1968
Victor Guillemin Harvard University 1962
Leonid Friedlander Massachusetts Institute of Technology 1989

Wrap Up

In Summary:

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.
- Python makes many good ideas from software engineering accessible for mathematical research.

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.
- Python makes many good ideas from software engineering accessible for mathematical research.
- It allows you to develop a correct prototype quickly ...

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.
- Python makes many good ideas from software engineering accessible for mathematical research.
- It allows you to develop a correct prototype quickly ...
- ... then improve your code's speed/usability when you know you've found a good idea.

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.
- Python makes many good ideas from software engineering accessible for mathematical research.
- It allows you to develop a correct prototype quickly ...
- ... then improve your code's speed/usability when you know you've found a good idea.
- It allows you to connect disparate tools and libraries...

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.
- Python makes many good ideas from software engineering accessible for mathematical research.
- It allows you to develop a correct prototype quickly …
- … then improve your code's speed/usability when you know you've found a good idea.
- It allows you to connect disparate tools and libraries…
- …and integrate them into a computer algebra system, Sage.

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.
- Python makes many good ideas from software engineering accessible for mathematical research.
- It allows you to develop a correct prototype quickly ...
- ... then improve your code's speed/usability when you know you've found a good idea.
- It allows you to connect disparate tools and libraries...
- ...and integrate them into a computer algebra system, Sage.

Wrap Up

In Summary:

- Mathematical research on the computer is not software engineering.
- Python makes many good ideas from software engineering accessible for mathematical research.
- It allows you to develop a correct prototype quickly ...
- ... then improve your code's speed/usability when you know you've found a good idea.
- It allows you to connect disparate tools and libraries...
- ...and integrate them into a computer algebra system, Sage.

Thanks!