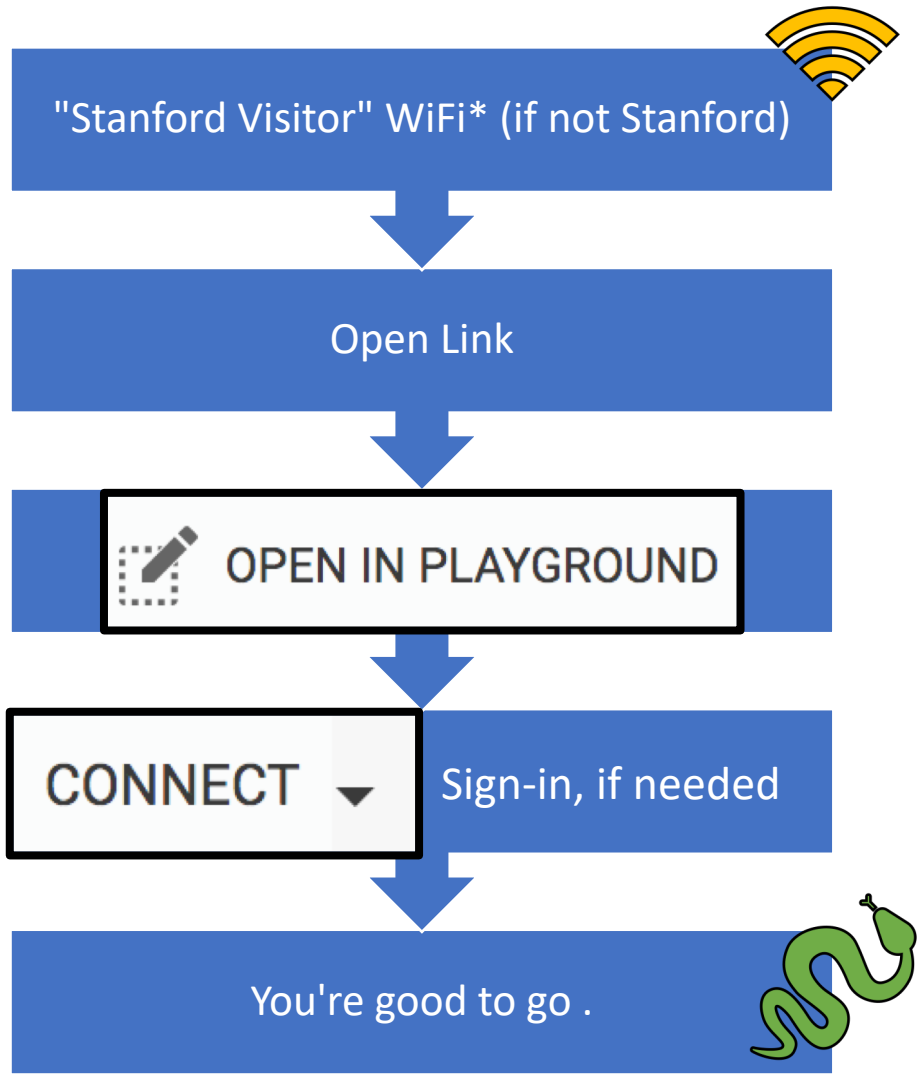
# Welcome to Introduction to Python

While we wait...

- Basic Python
  - `goo.gl/a83hej`
- Scientific Computing
  - `goo.gl/vzUdho`
- Data Science
  - `goo.gl/ggkU8Y`
- Machine Learning (probably not)
  - `goo.gl/ednjp6`




* Can't load "login" page? Navigate to `captive.apple.com` (yes, on Windows too ☺ - works in hotels too !)



# Introduction to Python

## (A gentle but hands-on approach)

Léopold Cambier
lcambier@stanford.edu

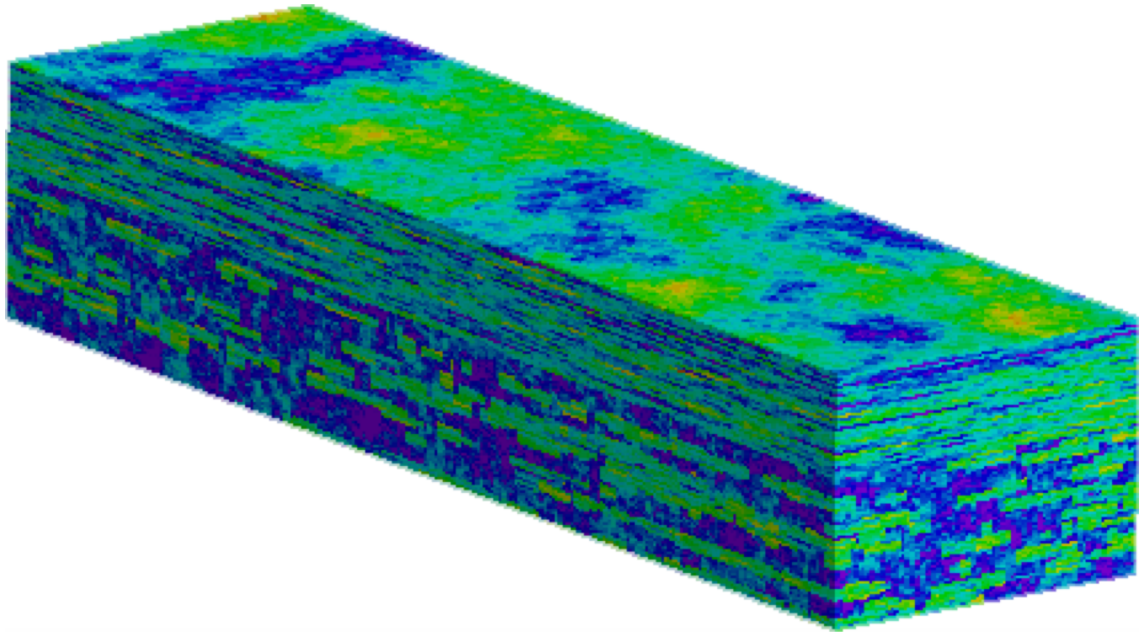
ICME Summer Workshops

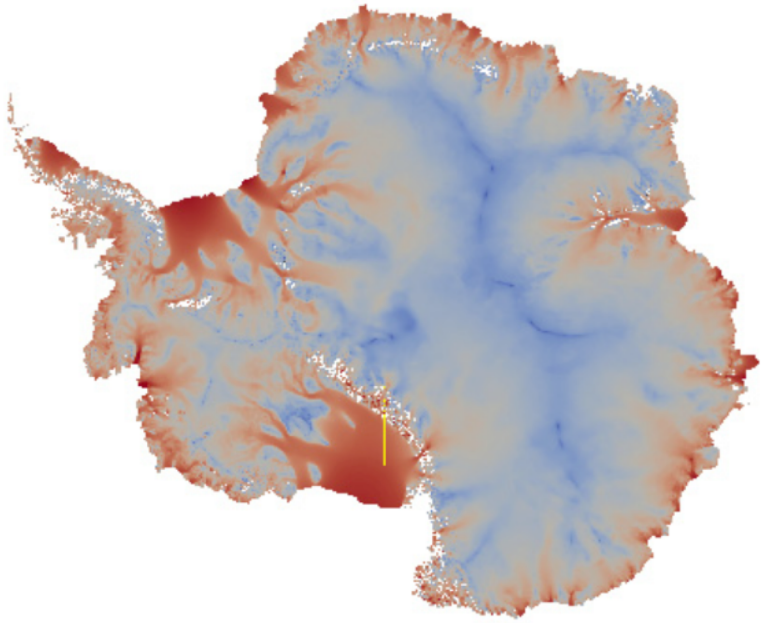
August 14, 2018

# Just one word about me

3rd year PhD student at ICME

Numerical linear algebra for (very large) sparse linear systems





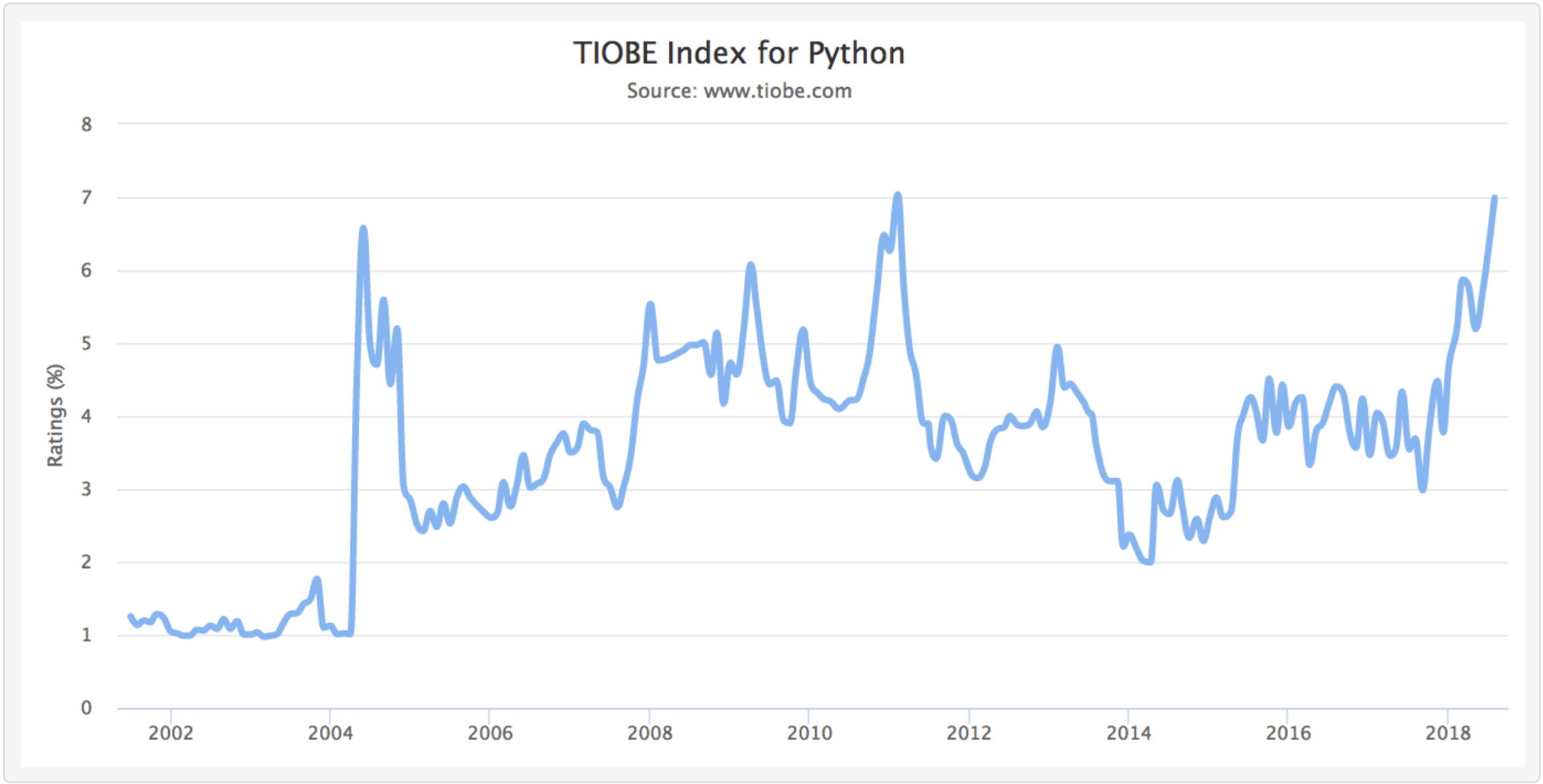
# Python


- A very popular programming language
  - Web applications
  - Scientific computing
  - Data science and Machine Learning
  - General purpose applications

# Python


| Aug 2018 | Aug 2017 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 16.881% | +3.92% |
| 2 | 2 | | C | 14.966% | +8.49% |
| 3 | 3 | | C++ | 7.471% | +1.92% |
| 4 | 5 | ^ | Python | 6.992% | +3.30% |
| 5 | 6 | ^ | Visual Basic .NET | 4.762% | +2.19% |
| 6 | 4 | v | C# | 3.541% | -0.65% |
| 7 | 7 | | PHP | 2.925% | +0.63% |
| 8 | 8 | | JavaScript | 2.411% | +0.31% |
| 9 | - | ^^ | SQL | 2.316% | +2.32% |
| 10 | 14 | ^^ | Assembly language | 1.409% | -0.40% |

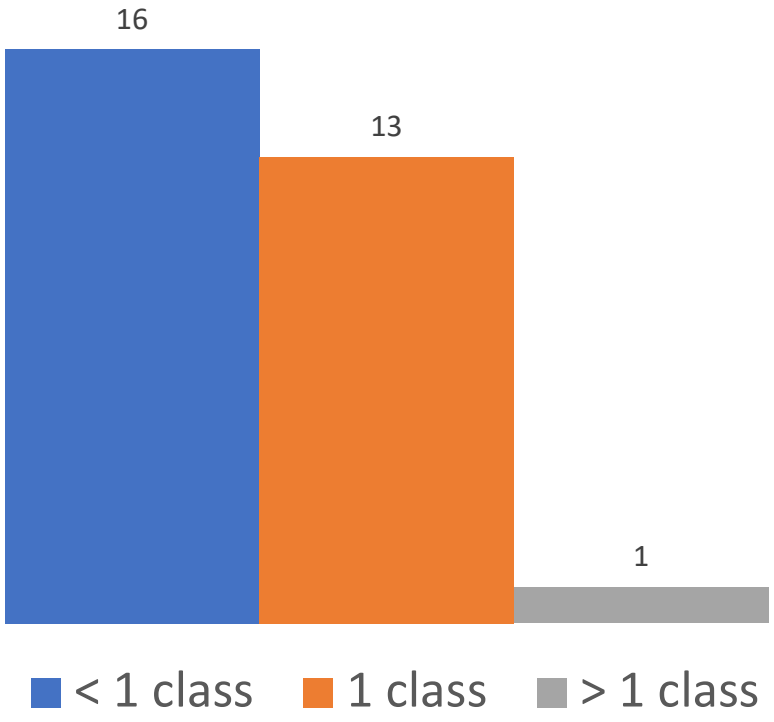
# Python

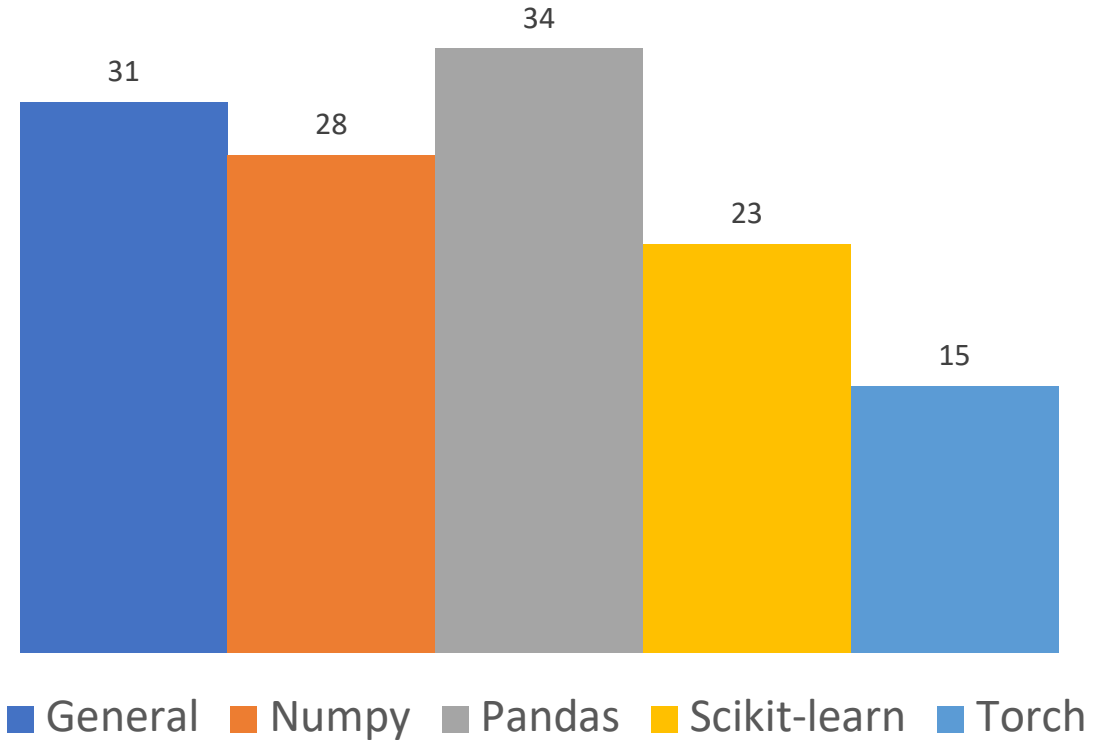

TIOBE Index for Python
Source: www.tiobe.com
Ratings (%)
0
1
2
3
4
5
6
7
8
2002
2004
2006
2008
2010
2012
2014
2016
2018

# Python


- High-level
- Portable
- Interpreted
- Extensible
- Object-oriented
- Dynamically typed
- Garbage collected

# The class


16
13
1
< 1 class
1 class
> 1 class


31
28
34
23
15
General
Numpy
Pandas
Scikit-learn
Torch

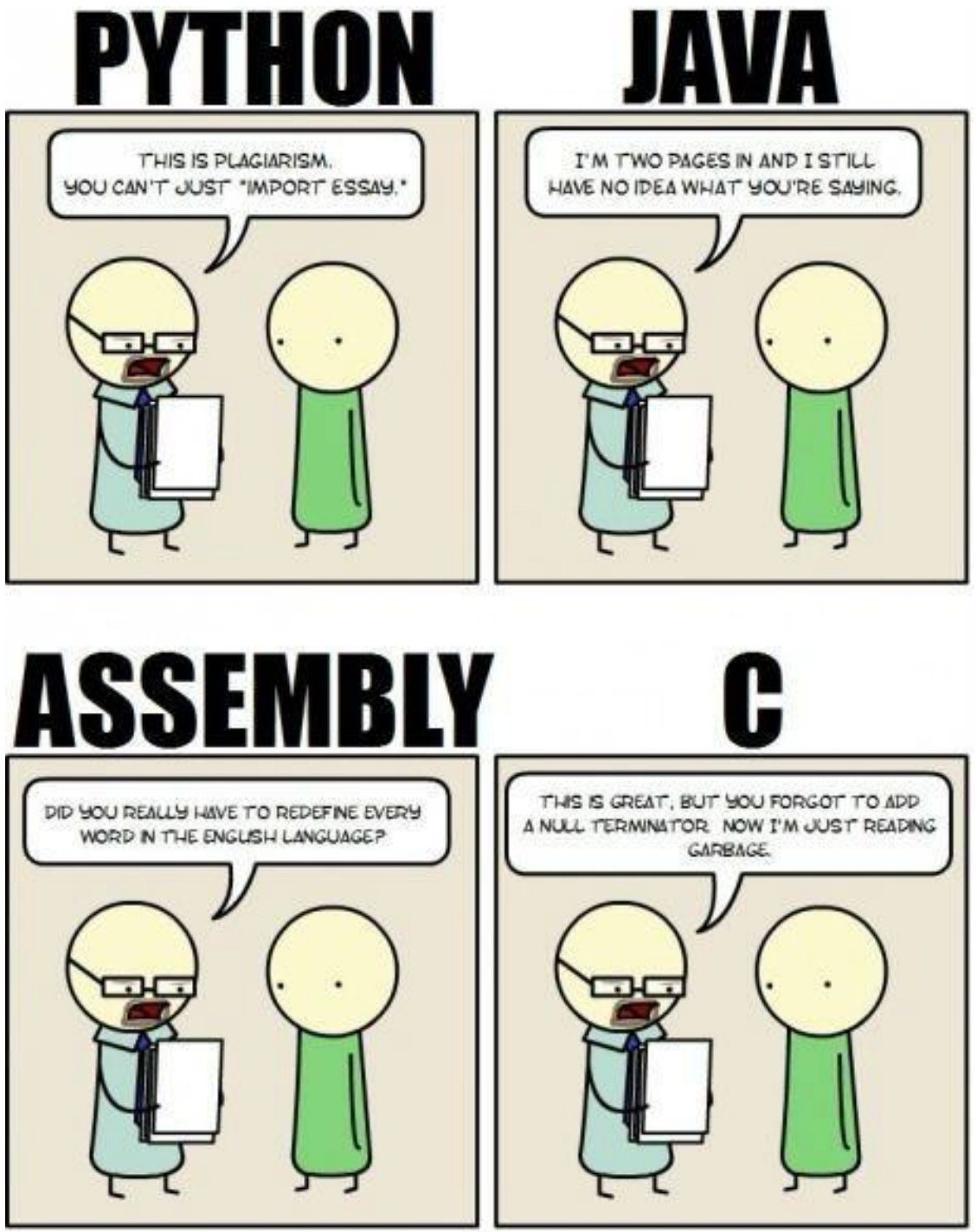
# The class


- Python basics
  - Variables, control-flow, containers, I/O
  - Functions, iterables, classes
  - Memory model, modules
- Scientific computing
  - Numpy & Scipy
- Data science
  - Pandas

# The class


- We will go through code *together.*
- Many exercises.
- To many exercises. Attempting just one is totally fine.

# Python Basics



# Python 2 vs Python 3

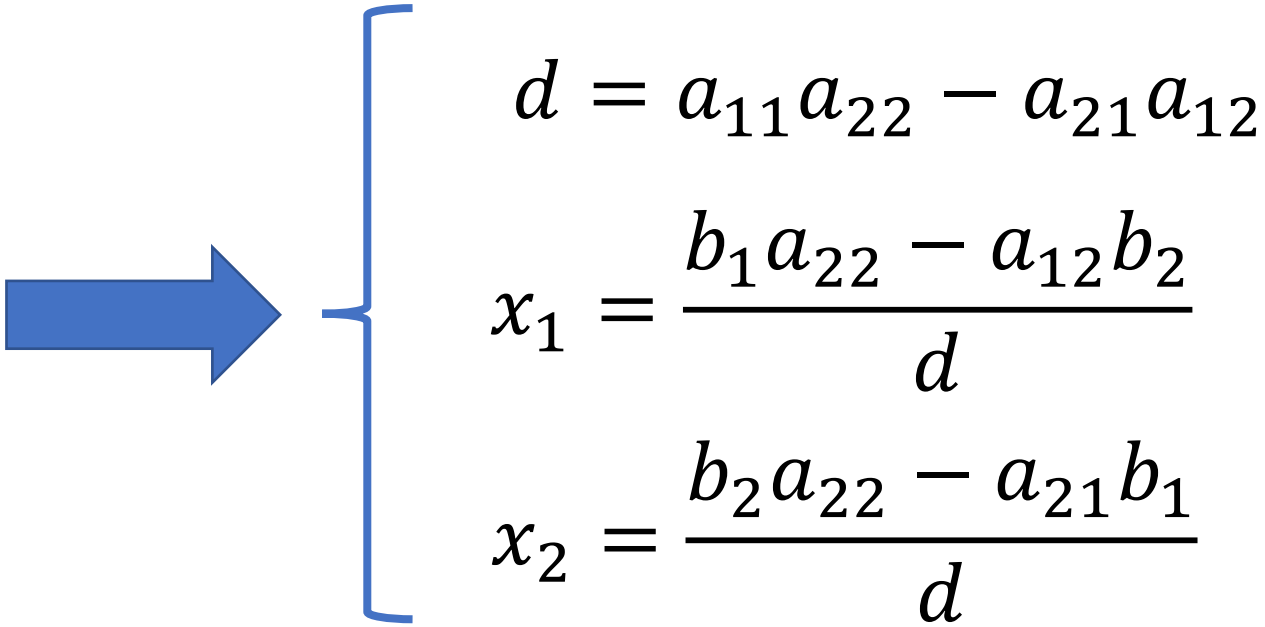
- We are using v3
- Not 100% backward compatible with v2
- The future of Python
- Supported by >95% of popular extensions
- v2 support ends in 2020

# Variables

- Numbers
- Strings (text)
- Booleans (true/false)
- Arbitrary other objects

# Exercise: a calculator

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$$

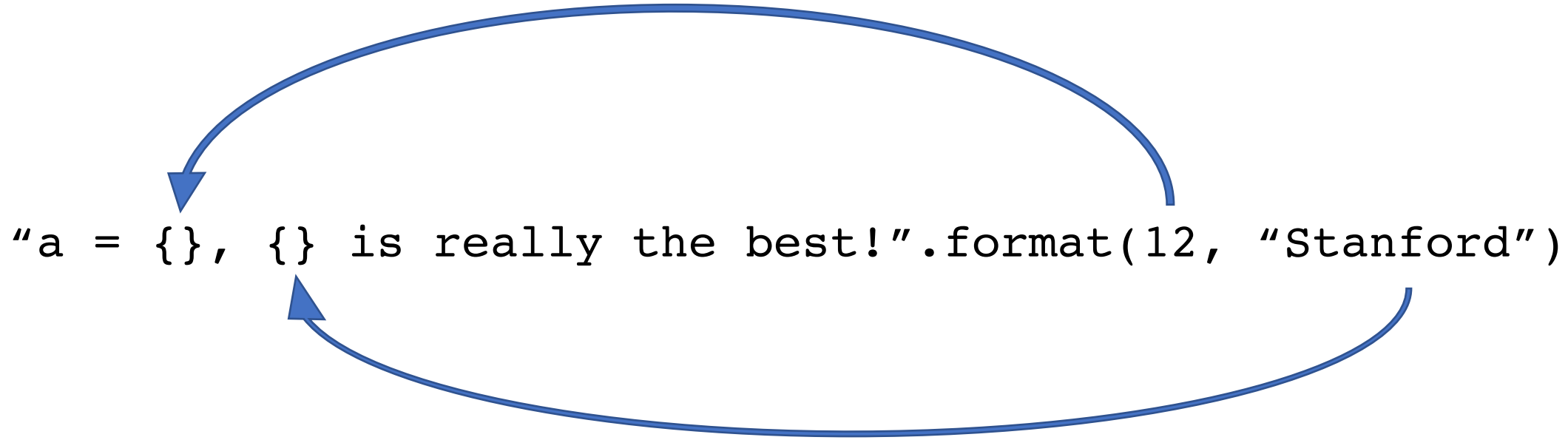


$$d = a_{11}a_{22} - a_{21}a_{12}$$

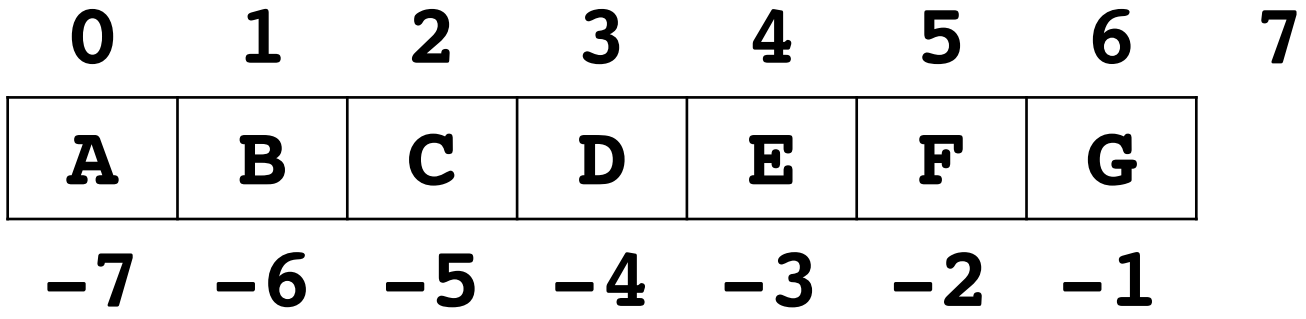
$$x_1 = \frac{b_1 a_{22} - a_{12}b_2}{d}$$

$$x_2 = \frac{b_2 a_{22} - a_{21}b_1}{d}$$

# Strings



# Strings



# Control-Flow

```
if (...):
    (...)
    pass
elif (...):
    (...)
else:
    (...)
```

```
for i in (...):
    (...)
    break
    continue
```

```
while (...):
    (...)
    break
    continue
```

# Exercise: approximating $\pi$

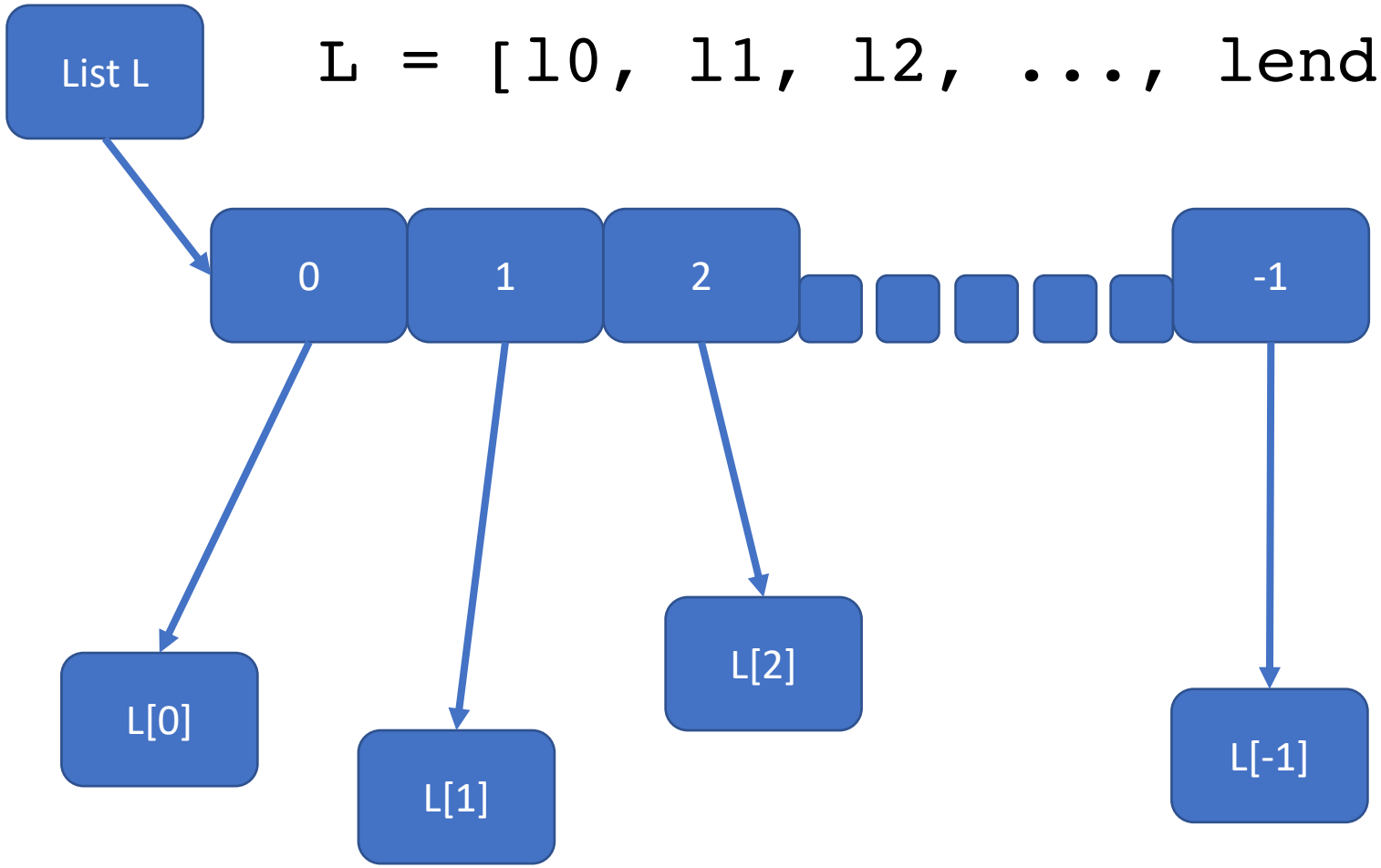
$$\pi = \sum_{k=0}^{\infty} \frac{12^{0.5}\left(-\frac{1}{3}\right)^{k}}{2k+1}$$

1) Compute approximation using 20 terms
2) Stop when error is less than $10^{-6}$
3) Plot error(k) as a function of k (use Google for that; we haven't seen everything needed yet)

# Containers

- List: sequence of elements
- Tuples: fixed-length sequence of elements
- Dictionaries: set of (key, value)

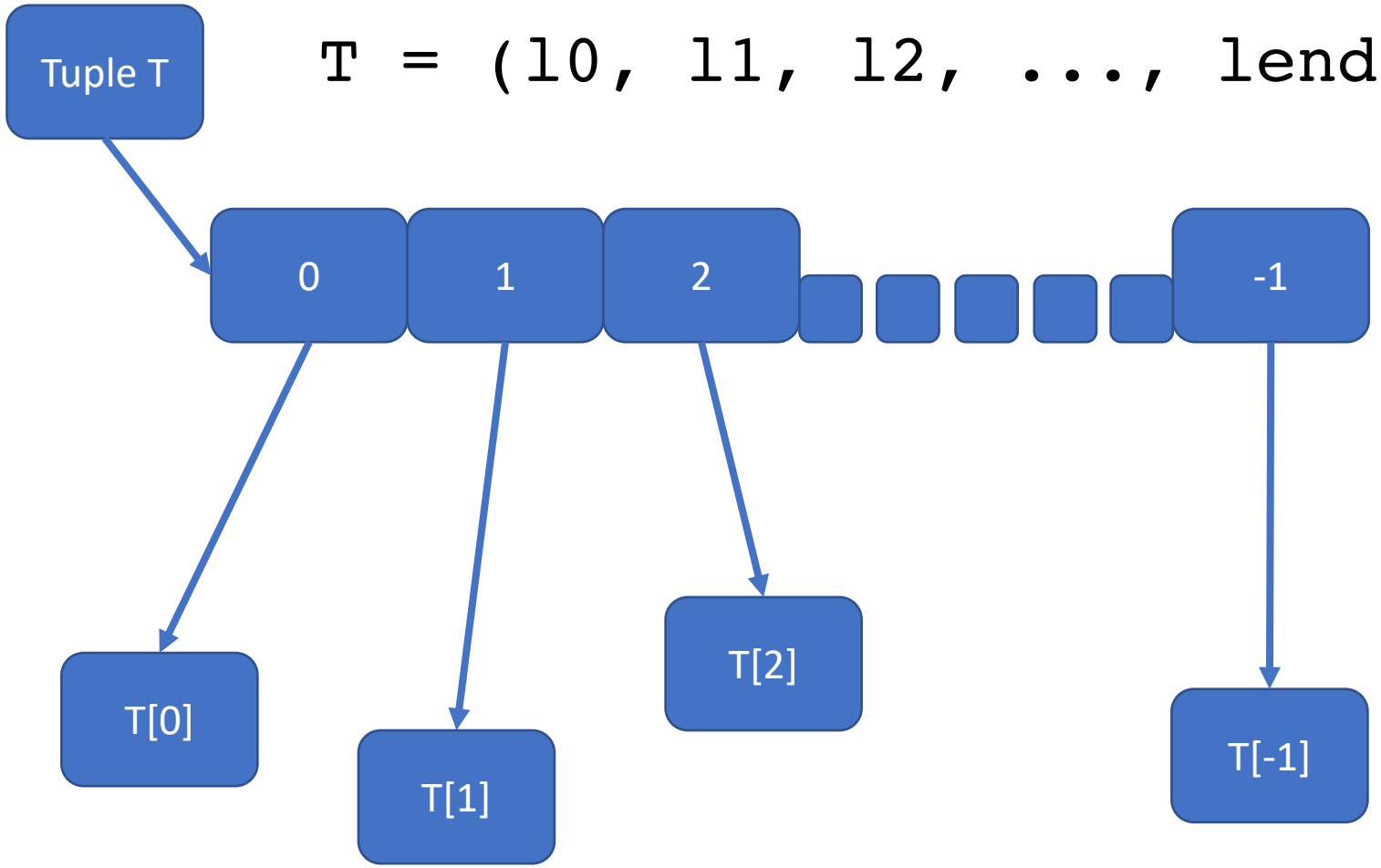
# List: mutable sequence of arbitrary objects



# Exercises: Dot, matrix-vector and matrix-matrix product

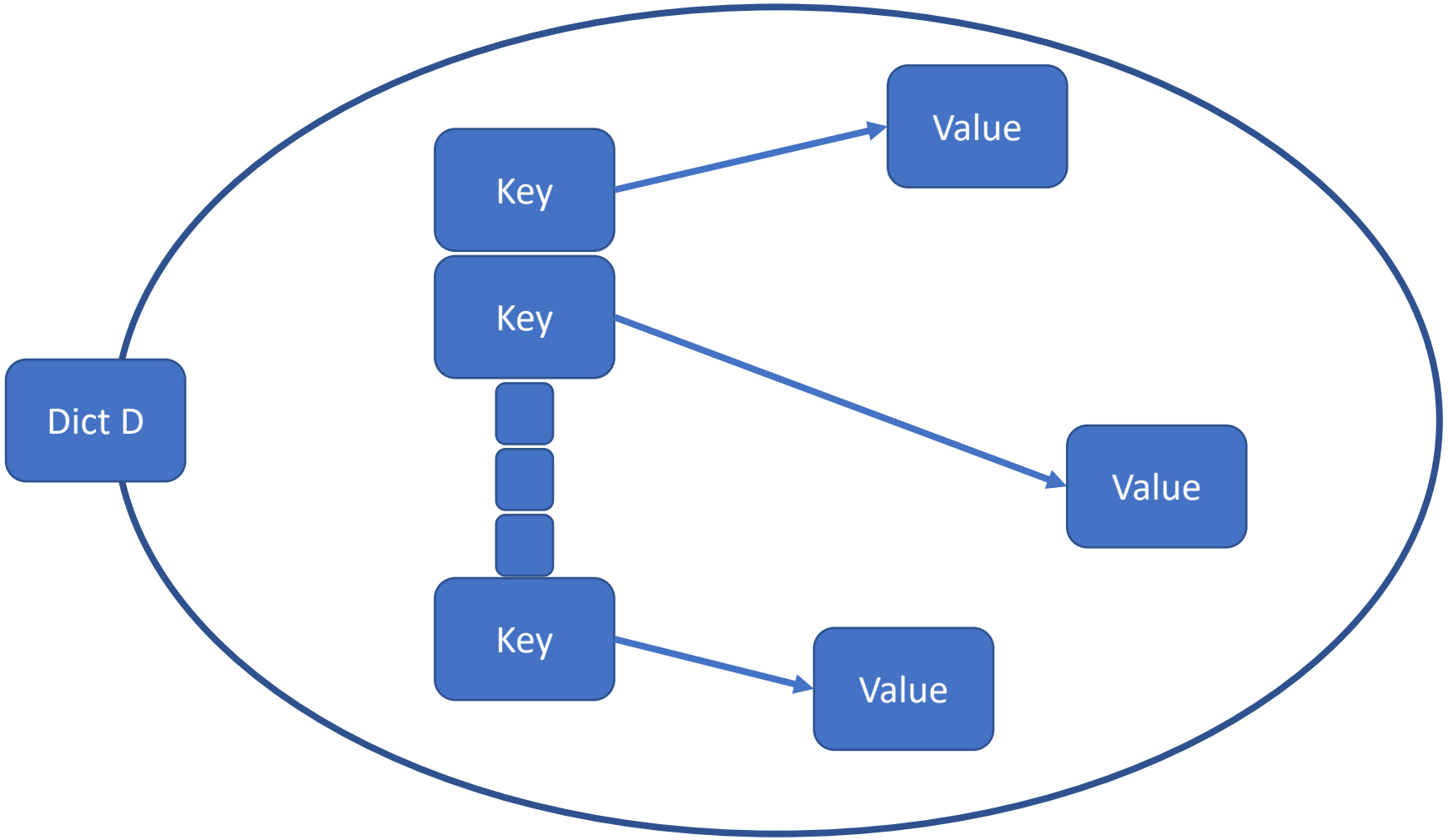
$$s = \sum_{i=1}^{N} a_i b_i \qquad y_i = \sum_{j=1}^{N} A_{ij} x_j \qquad C_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj}$$

# Tuple: immutable list



# Dictionary: collection of (key → value) pairs

```
D = {k0:v0, k1:v1, ..., kend:vend}
```



# Functions

```
def fun(x, y, z=2):
    # Do something with x, y, z
    return (...) # Optional
```

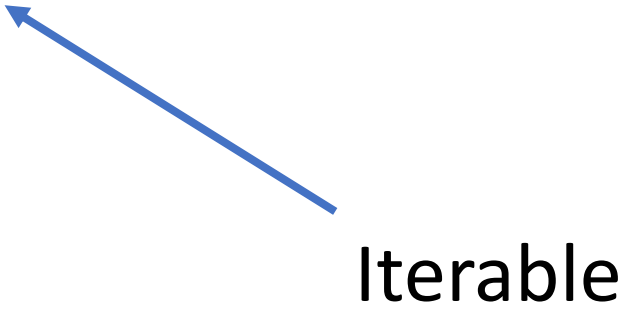
# Exercise: custom greetings

```
greet("John", "Bob")
    Hello John!
    Hello Bob!

greet("Alex", "Kelly", msg="Good morning")
    Good morning Alex!
    Good morning Kelly!
```

# Iterables

```
for x in I:
    (...)
```



# Generators

```
def gen(...):
    (...)
    yield (...) # ← Will stop and generate (...)
    (...)
    yield (...) # ← Same
    (...)
    yield (...) # ← Same
```

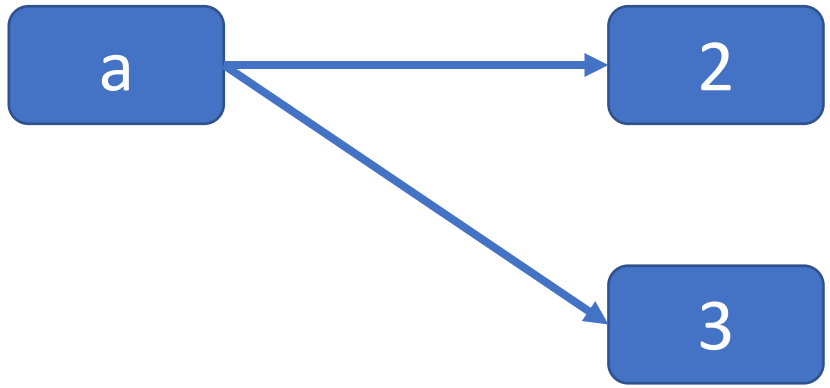
# Exercise: Fibonacci (finally!)

$$n_1 = 1$$
$$n_2 = 1$$
$$n_{i+1} = n_i + n_{i-1}$$

# Variables (1)

```
a = 2

a = 3
```



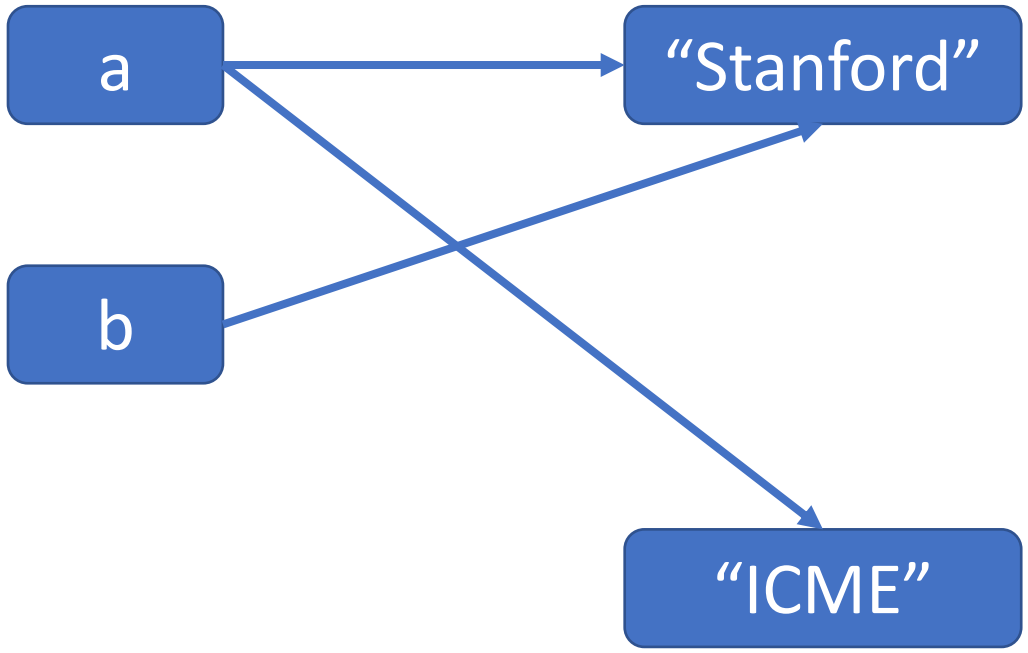
# Variables (2)

```
a = “Stanford”

b = a

a = “ICME”

print(b)
```




```
“Stanford”
```

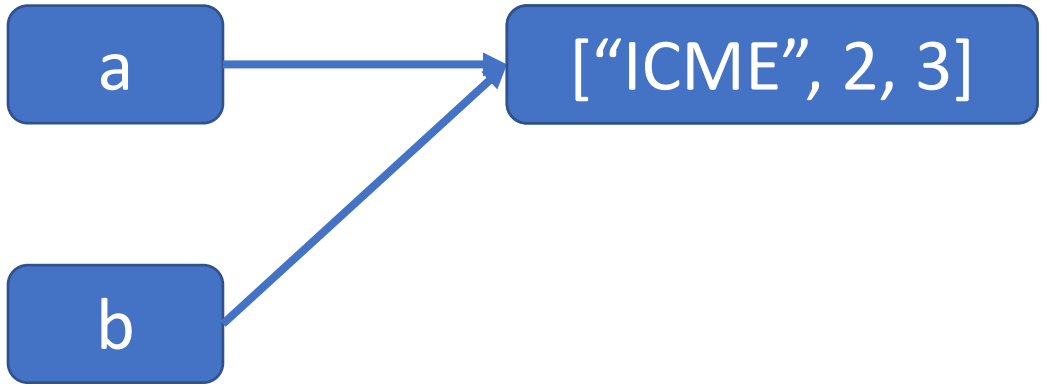
# Variables (3)

```
a = [1, 2, 3]

b = a

a[0] = “ICME”

print(b)
```




```
[“ICME”, 2, 3]
```

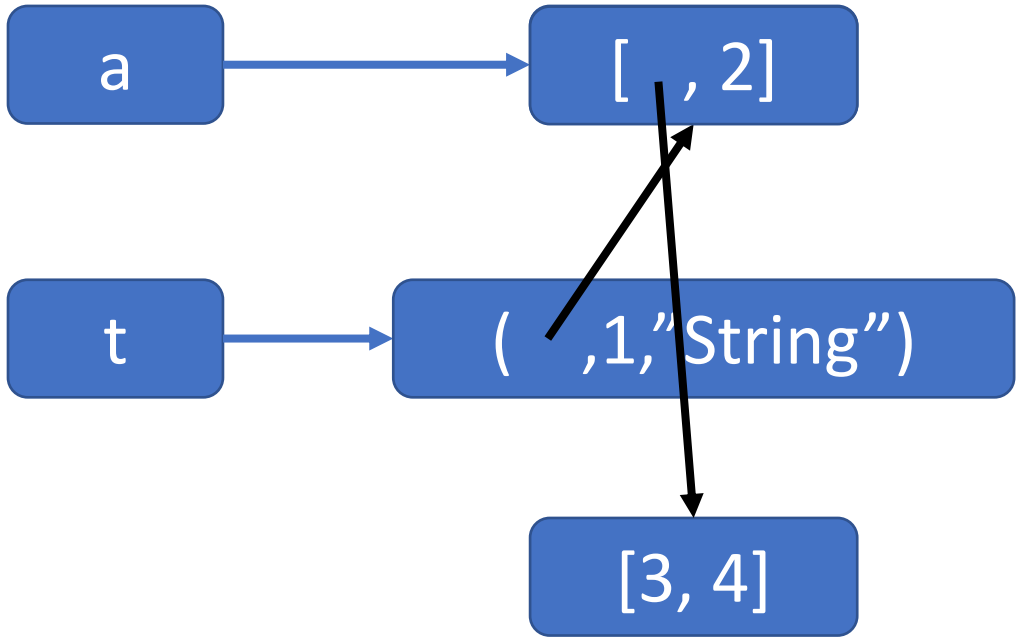
# Variables (4)

```
a = [1, 2]

t = (a, 1,“String”)

a[0] = [3, 4]

print(t)
```




```
([[3, 4], 2], 1, ”String”)
```

# Mutable vs Immutables

- List and most complex objects are mutables
- Int, Float, Strings, Tuple are immutable

# Functions arguments

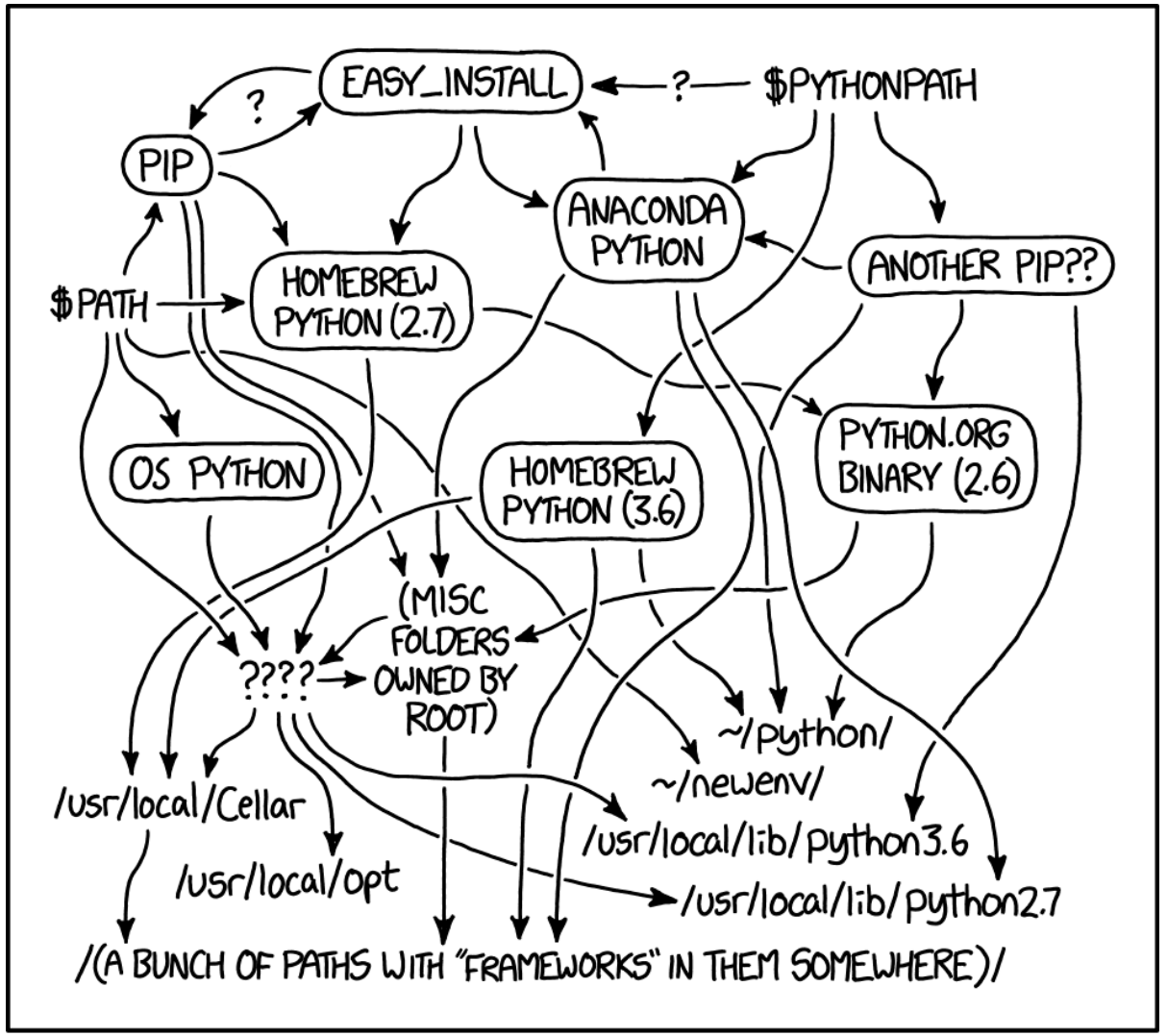
```
def fun(x):
    # x points to some object created outside
    # If x is mutable, outside can be changed
    # If x is immutable, cannot be changed
    #   (-but whatever x contains could be)
```

# Classes

```
class Complex:
    def __init__(self, real, imag):
        self.r = real
        self.i = imag
    def add(self, other):
        c = Complex(0, 0)
        c.r = self.r + other.r
        c.i = self.i + other.i
        return c
```

# More Advanced Concepts

(https://www.xkcd.com/1987/)




MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

# Modules

```
# (1)
import math
print(math.pi)
print(math.sqrt(2))

# (2) [~ (1)]
import math as m
print(m.pi)
print(m.sqrt(2))
```

```
# (3)
from math import pi, sqrt
print(pi)
print(sqrt(2))

# (4) [Bad]
from math import *
print(sqrt(2))
print(exp(1))
```

# On your laptop

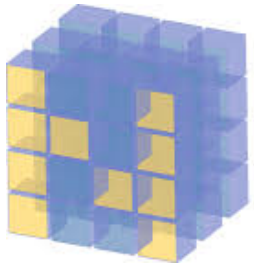
- Install Python. An easy way is through Anaconda
  - https://www.anaconda.com/
- In a terminal, (Powershell on Windows)
  - `python script.py`


# Python for Scientific Computing

(Summit HPC at ORNL)

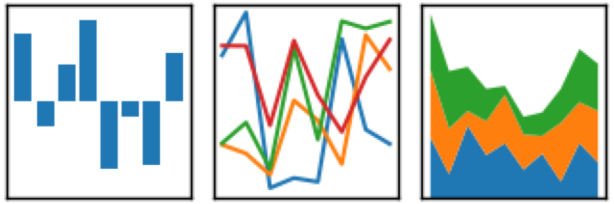
# SciPy


- An ecosystem for Scientific Computing in Python
- Includes many packages

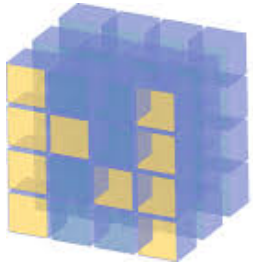




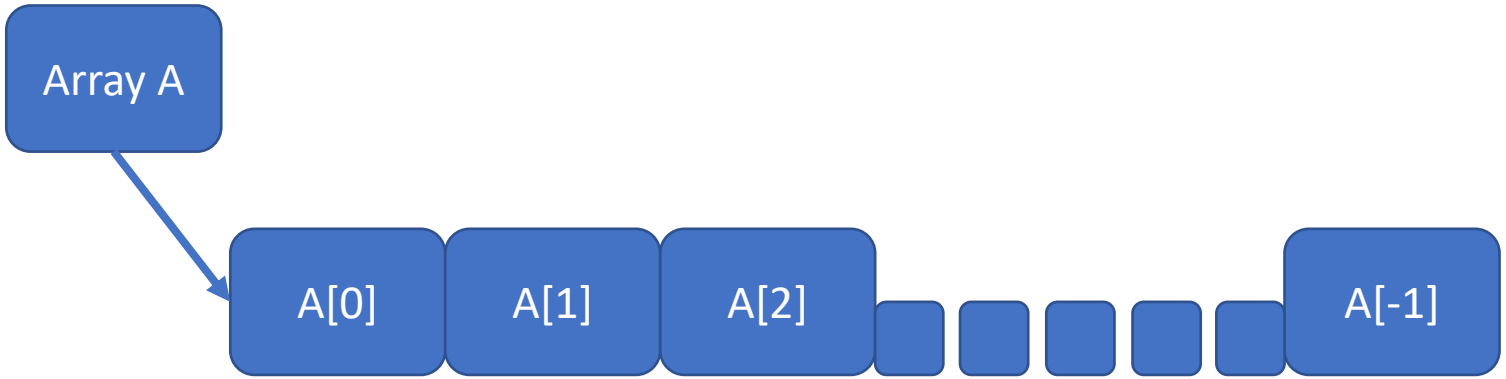




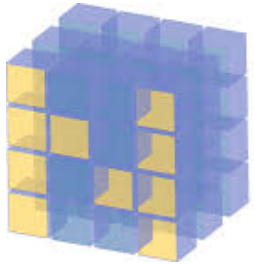
# Numpy



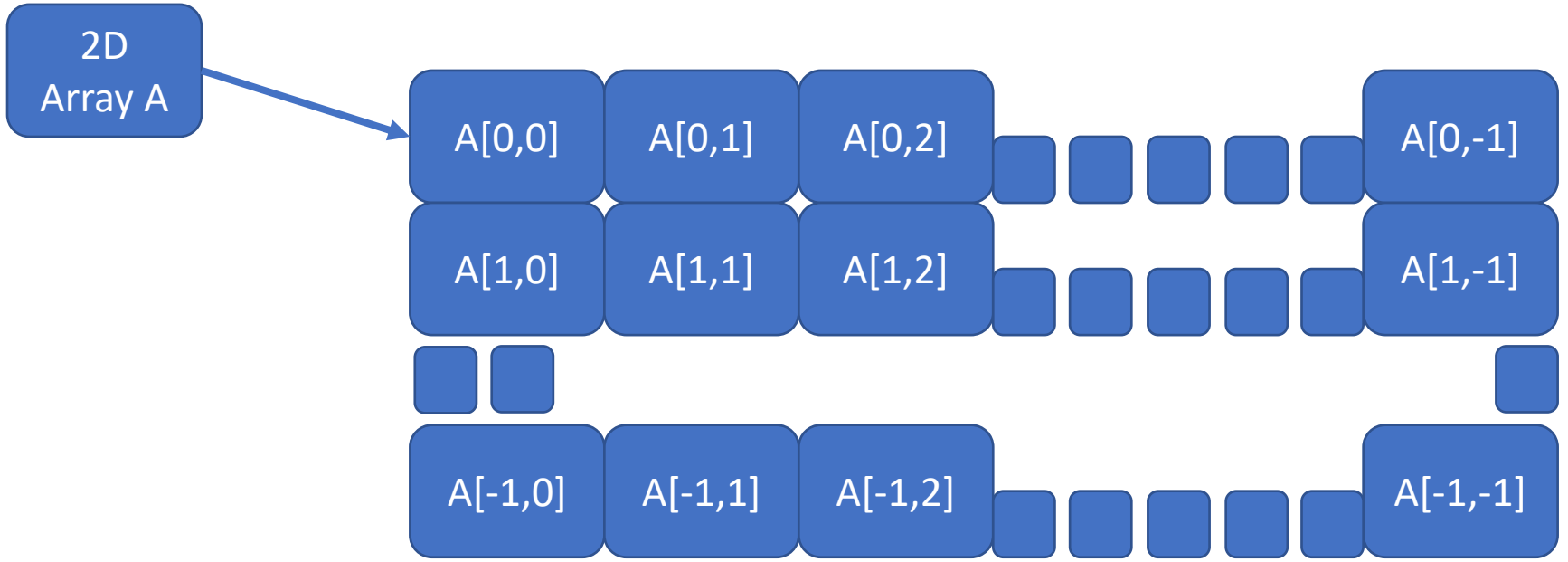
Introduces arrays !



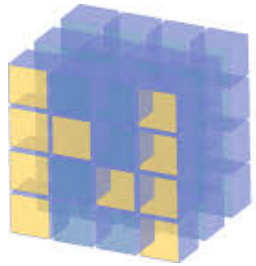
# Numpy arrays



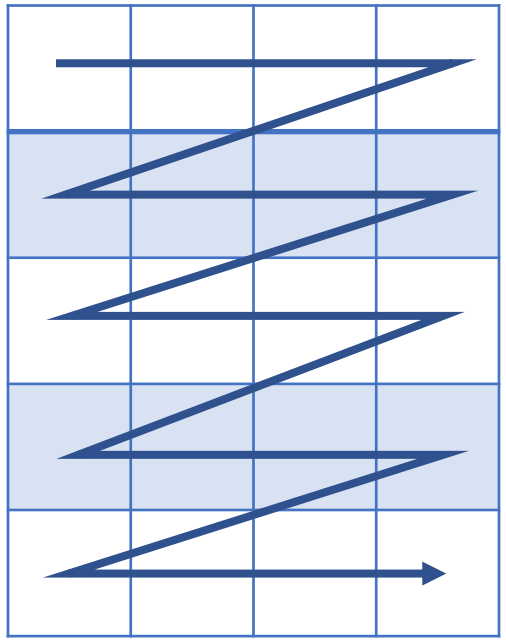
1d or 2d or ... nd arrays



# Array reshapes

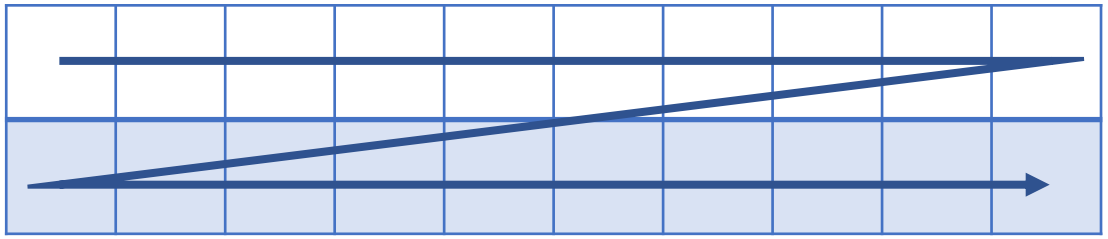


5 x 4

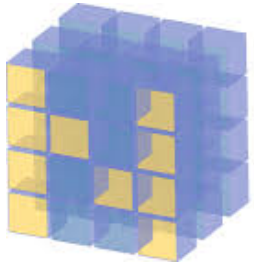


.reshape((2, 10)) =

2 x 10



# Array broadcasting



3 x 2-array

| | |
|---|---|
| | |
| | |
| | |

\+

2-array (vector)

| | |
|---|---|
| a | b |

=

3 x 2-array

| | |
|---|---|
| | |
| | |
| | |

\+

3 x 2-array

| | |
|---|---|
| a | b |
| a | b |
| a | b |

# Exercise: Mandelbrot

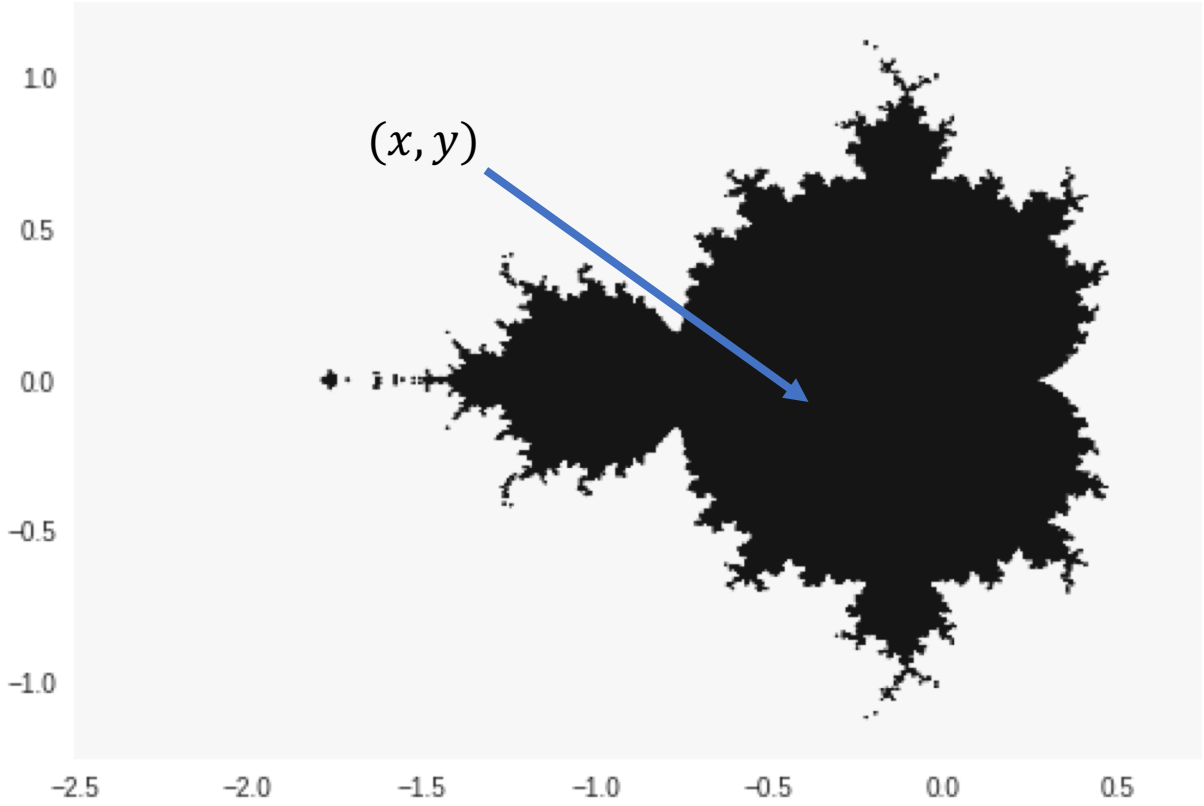
Given a point $(x, y)$. Color $c(x, y) = ?$

1. Generate sequence $(x_n, y_n)$ by

$$x_0 = 0, y_0 = 0$$

$$\begin{cases} x_n = & x_{n-1}^2 - y_{n-1}^2 + x \\ y_n = & 2x_{n-1}y_{n-1} + y \end{cases}$$

2. If $x_n^2 + y_n^2$ grows above 2 at any point, stop and set $v(x, y) = 0$ (white). Otherwise, $v(x, y) = 1$ (black).



# Exercise: least-square

Find $a_0, a_1, a_2$

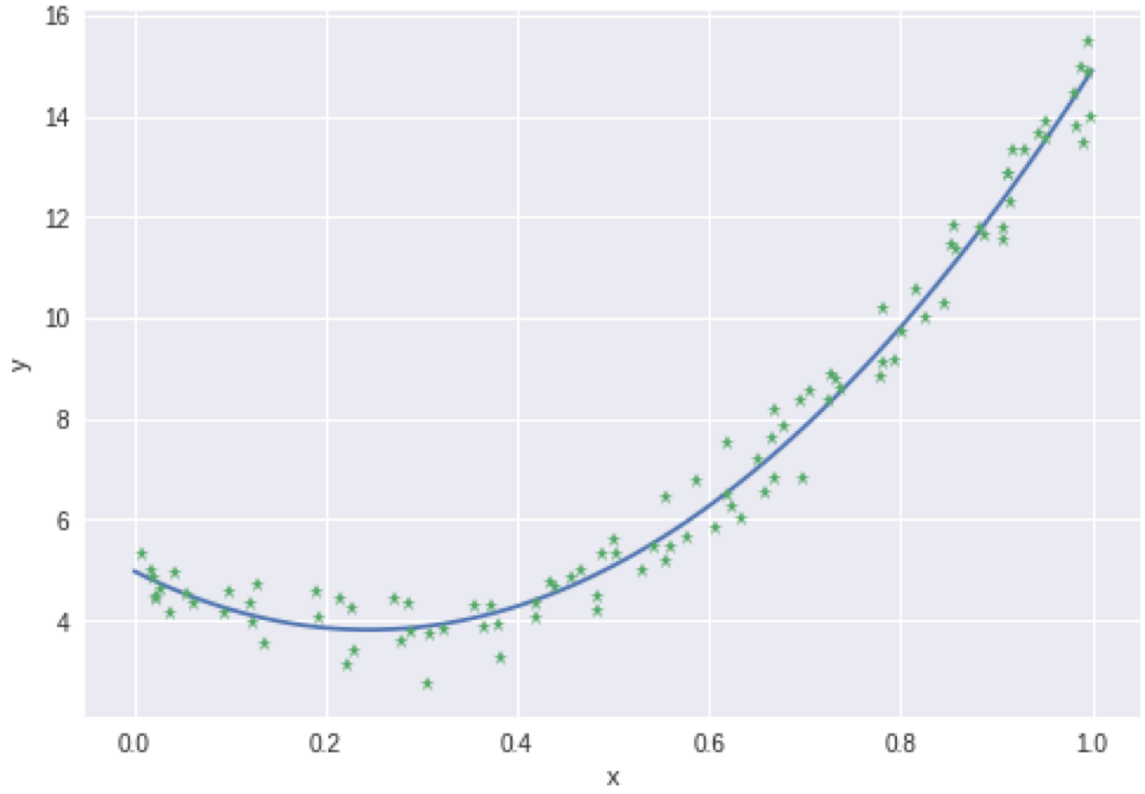
$$\{(x_i, y_i)\}_{i=1}^{N}$$

$$y(x_i) \approx a_0 + a_1 x_i + a_2 x_i^2$$

How ? Solve least square

$$\min ||Aa - b||$$

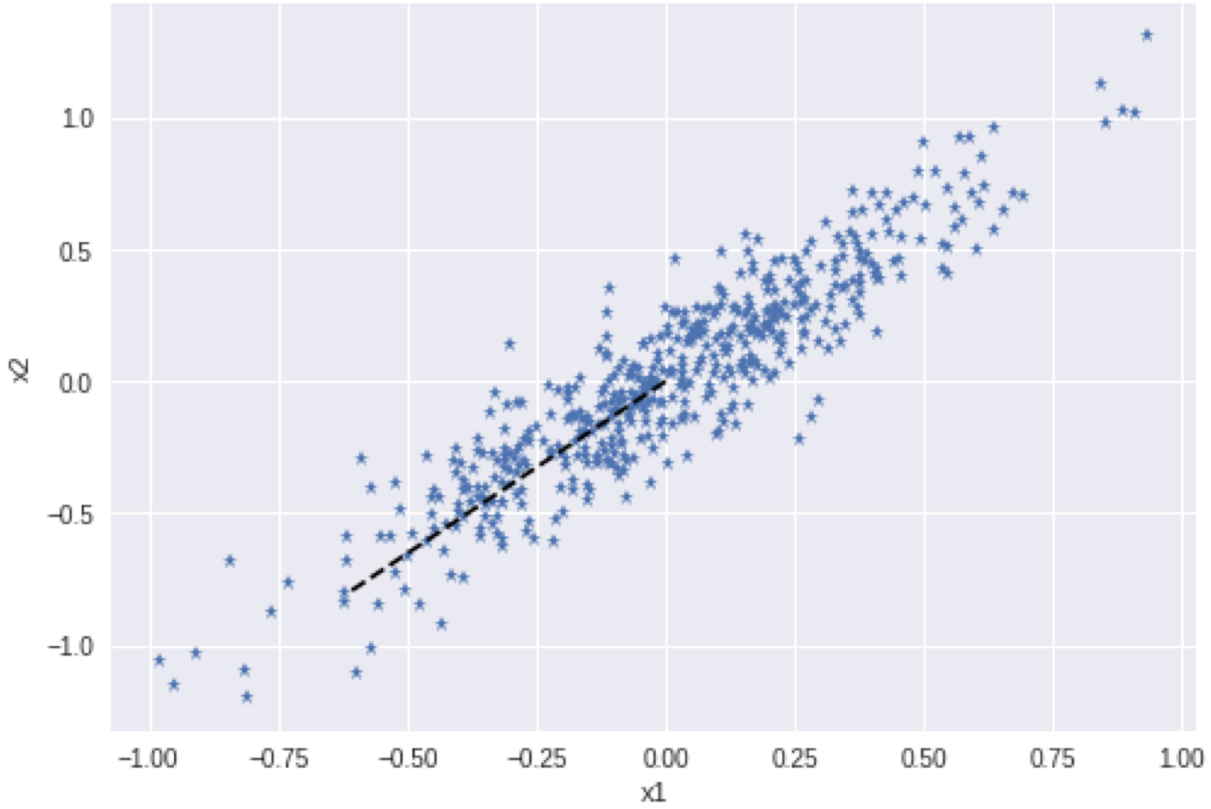
$$A = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix} \text{ and } b = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$



# Exercise: PCA

Given a data matrix

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_N \\ | & | & & | \end{bmatrix}$$




the leading singular vector, `U[:,0]`

```
(U,s,V) = numpy.linalg.svd(X)
```

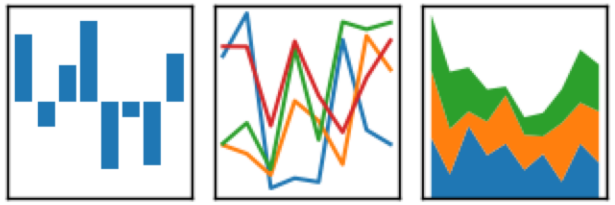
returns the principal direction

# Python for Data Science

# Pandas

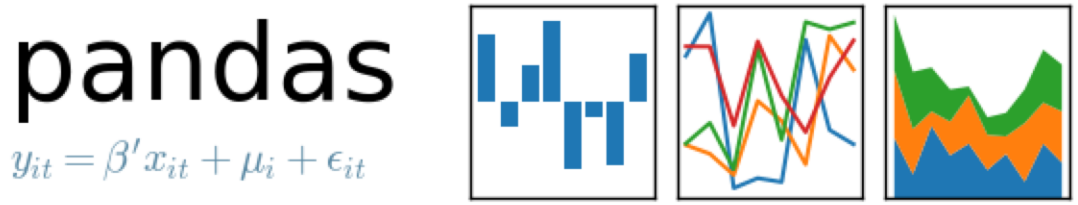
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



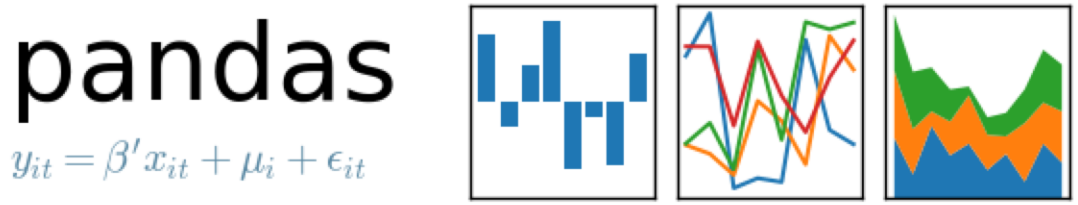
- An open-source, high-performances & easy-to-use data structures package
- Labelled Series (1D) & DataFrames (2D) objects
- All kinds of aggregation, grouping, reductions, statistics, etc.
- Powerful dates support
- All kinds of read/write functions (csv, HDF5, etc.)

# Pandas Series




```
import pandas as pd
s = pd.Series(data, index=...)
```

# Pandas DataFrame

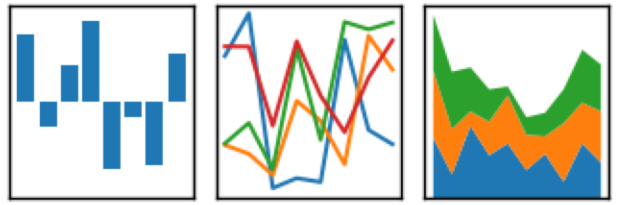



```
import pandas as pd
s = pd.DataFrame(data, index=..., columns=...)
```

# Accessing a DataFrame

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



- By Label
  - `df[..]` `# Get some columns or rows`
  - `df.loc[..,..]` `# End-points INCLUDED`
- By position
  - `df.iloc[..,..]` `# End-points NOT INCLUDED`

# Scikit-learn


- The package for machine learning
- Supervised learning
  - Classification
  - Regression
- Unsupervised

# Scikit-learn


- A typical supervised learning problem
- Given

$$S = \{x_i, y_i\}_{i=1}^{N}$$

- Learn a function (mapping)

$$y = F(x)$$

- Scikit-learn has all kinds of models and algorithm

# Scikit-learn workflow


1. Setup `x_train`, `y_train`
2. Choose a model
3. Train a model
4. Use model to predict

# PyTorch


- One of (many) Neural Net packages in Python
  - TensorFlow, Theano, Caffe, CNTK, PyTorch
- One of the simplest to use
- Key components
  - Tensor (like numpy Ndarrays), with CPU & GPU support
  - Neural Net with automatic differentiation
  - Optimization routines

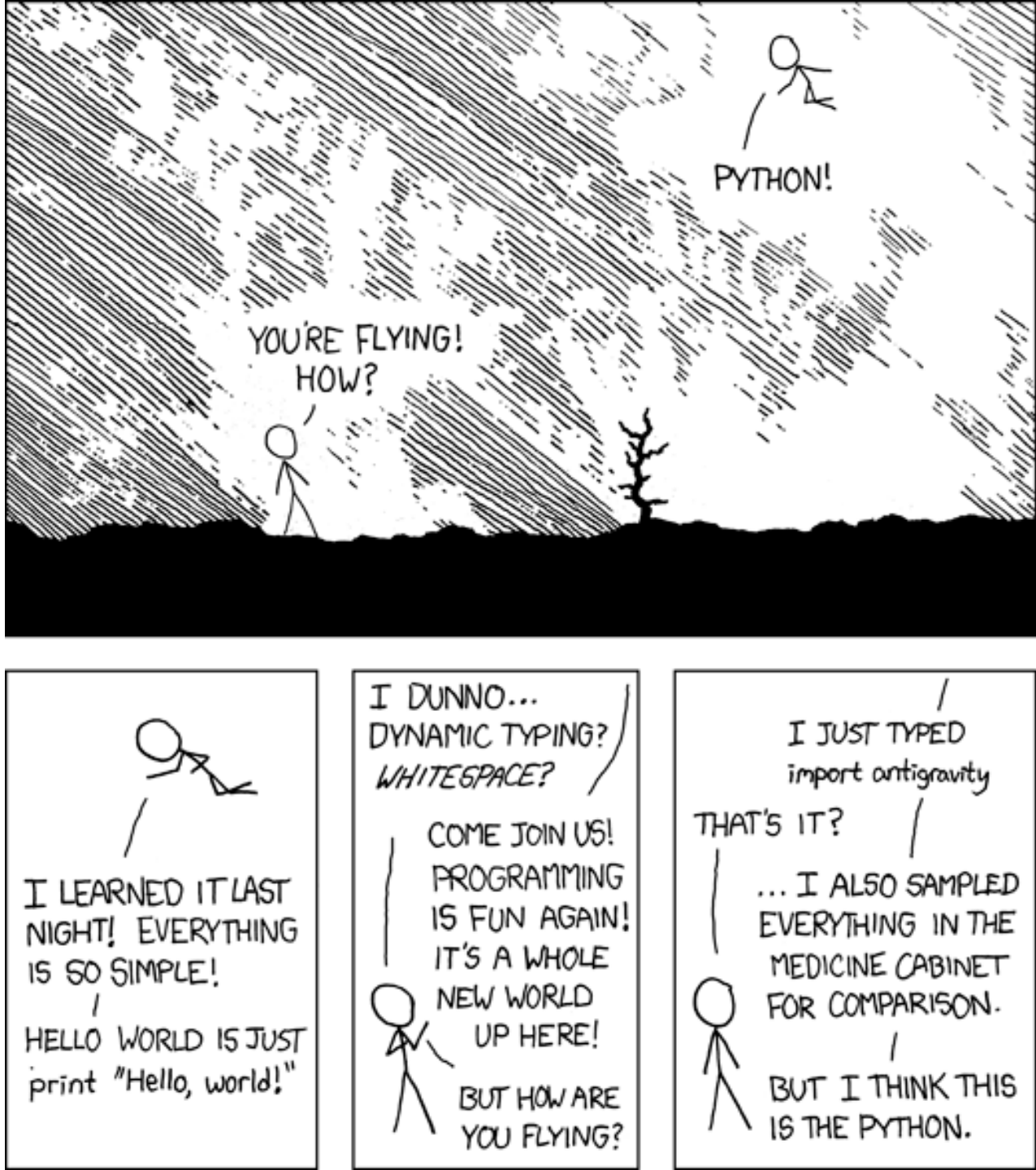
# PyTorch workflow


1. Define neural net topology
2. Choose loss criterion and optimization algorithm
3. For epochs ...
   - For batches $(x, y)$...
     - $y = F_w(x)$
     - $L_w = \text{loss}(x, y)$
     - $g_w = \frac{\partial L_w}{\partial w}(x)$
     - $w^+ = w - \alpha g_w$

# Recap

```
import antigravity
# Try it on your laptop,
# In a Python interpreter
```

(https://xkcd.com/353/)



# Recap: What did we learn?

- Basic Python
- Numpy
- Matplotlib.pyplot
- Pandas

# References

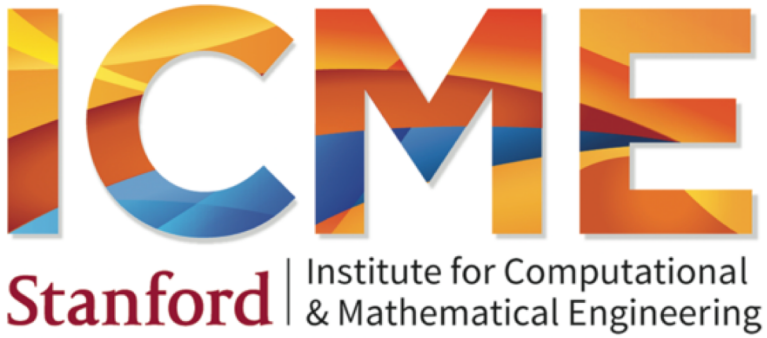
- Python
  - Google & Stackoverflow
  - https://docs.python.org/3/
  - https://developers.google.com/edu/python/
  - http://www.practicepython.org/
- Numpy & Scipy
  - https://docs.scipy.org/doc/numpy/user/quickstart.html
- Pandas
  - https://pandas.pydata.org/pandas-docs/stable/10min.html
  - https://github.com/jvns/pandas-cookbook
- Pytorch
  - https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
- Scikit-learn
  - http://scikit-learn.org/stable/tutorial/basic/tutorial.html

# At Stanford

- CME211
  - Software Development for Scientists and Engineers
- CS106AP
  - Programming Methodology in Python
- CS102, CS131, CS230, CS231N, CS375: Machine Learning (using Python)
- CME302: Linear Algebra (using Python)

# Any question after the class?

lcambier@stanford.edu




https://icme.stanford.edu/