SCIENTIFIC PROGRAMMING IN PYTHON

IAN HOFFECKER
ian.hoffecker@ki.se

Department of Medical Biochemistry and Biophysics
Karolinska Institutet, Stockholm Sweden

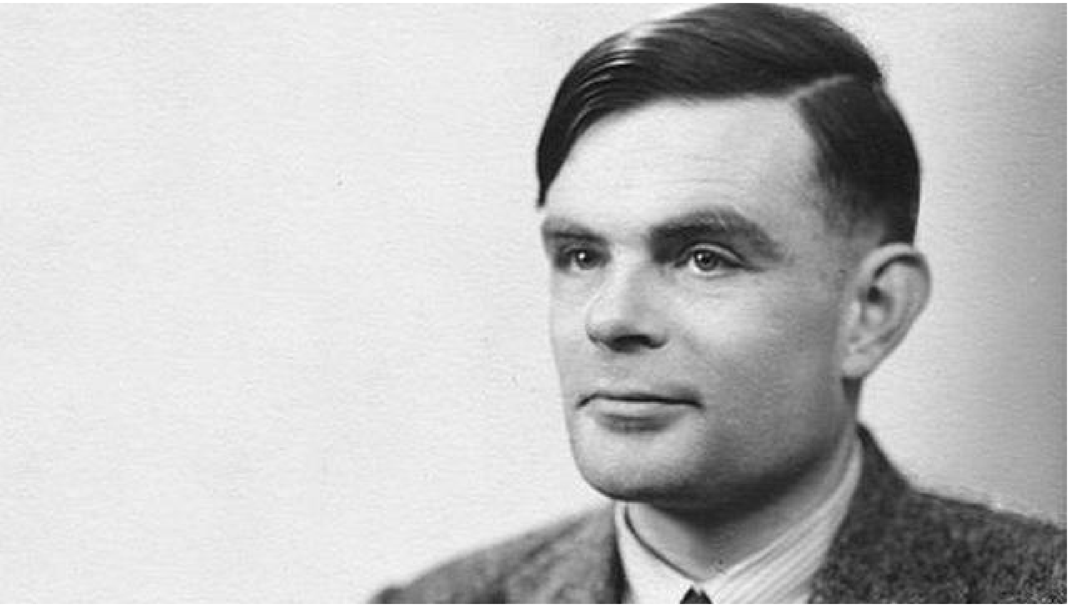
Outline

- **Motivation**
  - scientific programming
  - Python vs other languages
- **The anatomy of a program**
  - fundamentals
  - flow diagrams
- **Basic concepts - demo**
  - variables
  - lists
  - conditional statements
  - loops
  - files, input and output,
- **Solving scientific problems with programming**
  - analyzing and visualizing data
- **Tips to get started on your own**
  - editors and consoles
  - anaconda - scientific programming packages
  - learning resources
  - finding your first “personal” project

Get Inspired

A new, a vast, and a powerful language is developed for the future use of analysis, in which to wield its truths so that these may become of more speedy and accurate practical application for the purposes of mankind than the means hitherto in our possession have rendered possible.

- Ada Lovelace

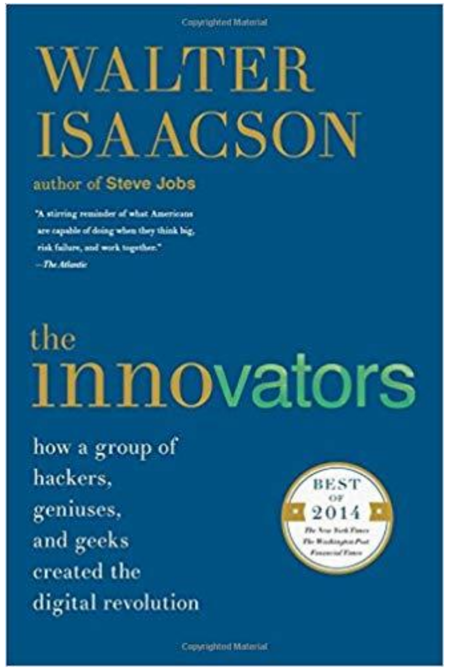


Programming is a skill best acquired by practice and example rather than from books.

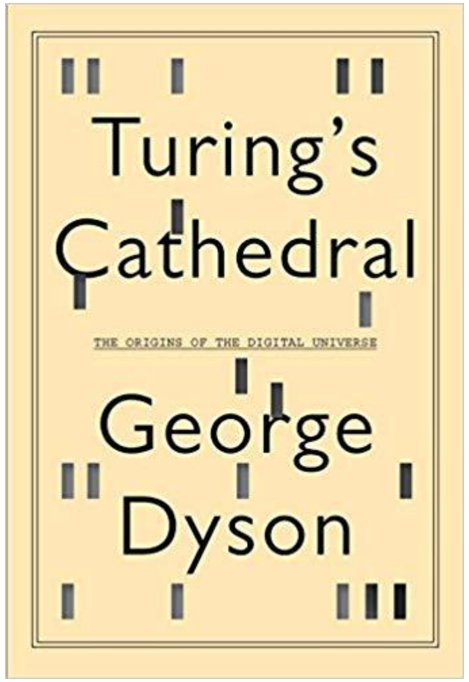
- Alan Turing

Inspirational reading about the history of computing

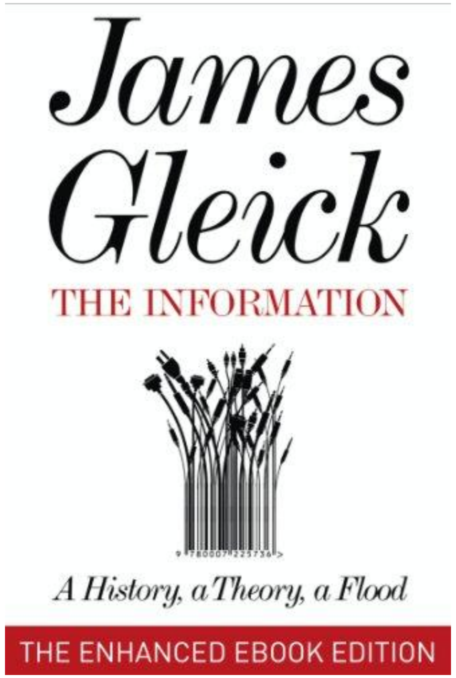
- **The Innovators - Walter Isaacson**
  - about the history of computing, programming, transistors, the internet...


WALTER ISAACSON
author of Steve Jobs
"A stirring reminder of what Americans are capable of doing when they think big, risk failure, and work together."
—The Atlantic
the innovators
how a group of hackers, geniuses, and geeks created the digital revolution
BEST OF 2014

- **Turing's Cathedral - George Dyson**
  - about the first stored memory digital electronic computers and the role of John Von Neumann



- **The Information - James Gleick**
  - about the history of information theory



Expectations and Plan for the Course

Some of you...

  - are already confident, competent scientific programmers
  - have some experience programming but are not confident about it
  - know a programming language, but it is not the one we are doing
  - have zero experience programming

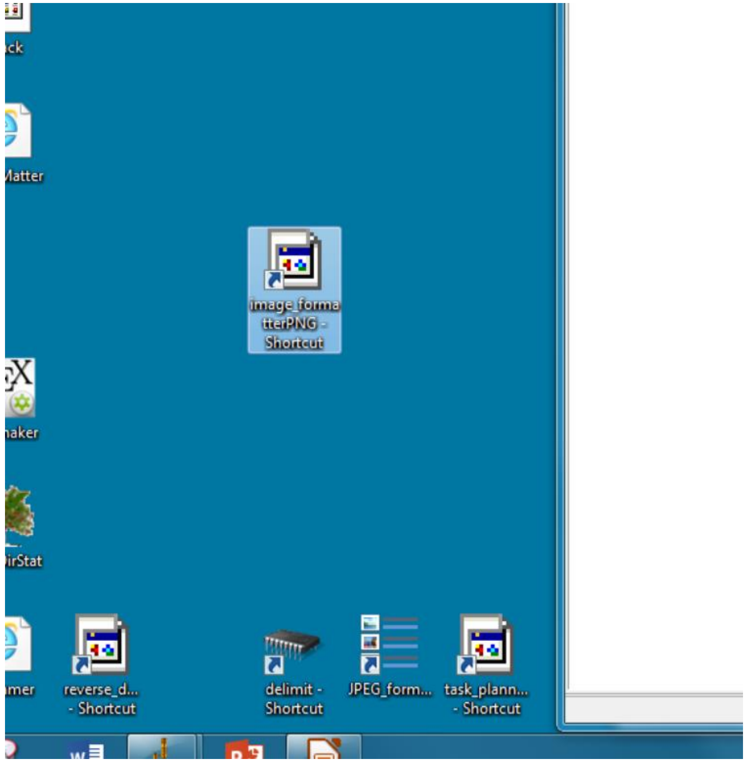
- **We will...**
  - introduce/remind you of basics concepts in programming today
  - give you some exposure to scientific programming
  - use this basis for learning bioinformatics throughout the rest of the course

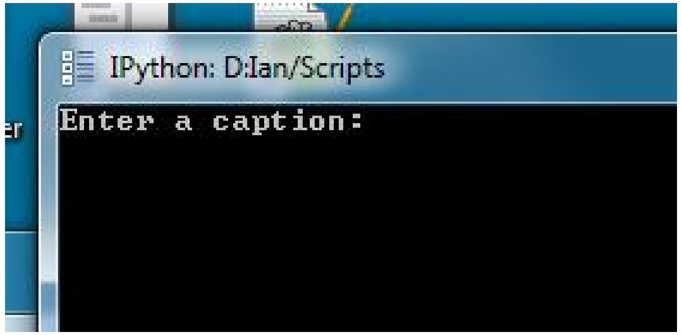
- **If you are new to programming**
  - spend extra time on the basics
  - ask us and your peers for help
  - research it independently

- **If you are already advanced**
  - use the tools we give you to experiment on your own
  - help your peers

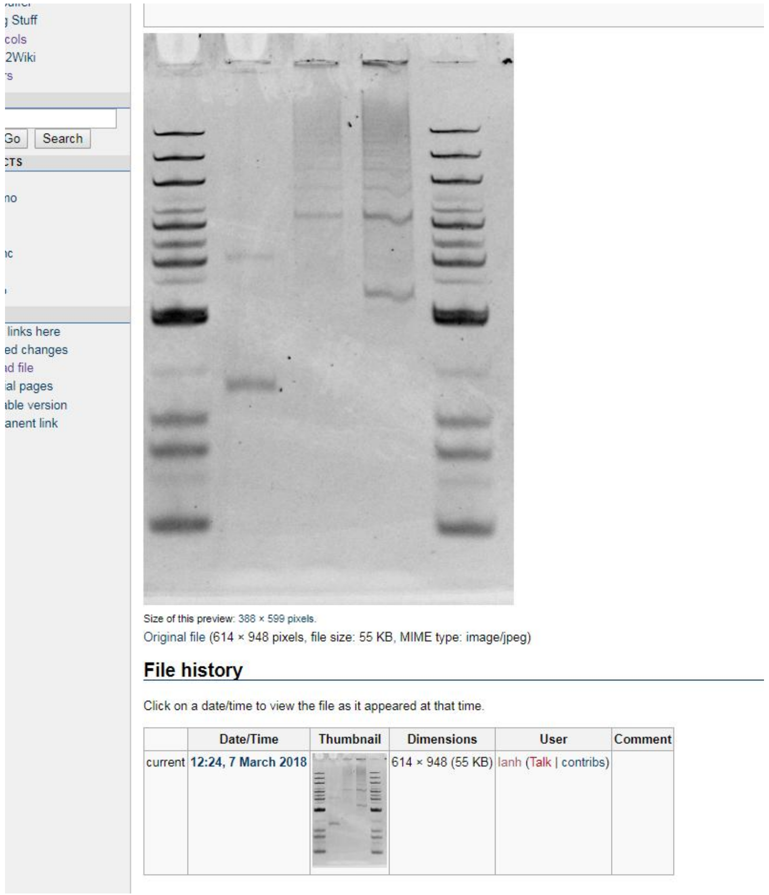
Uses for programming (specifically Python examples)

- **Scripting / file management**
  - programs that manage files, copying, creating folders, importing data from text files, sorting images....
  - Eliminate or reduce the cost of repetitive tasks





```
Enter a caption:
```



Size of this preview: 388 × 599 pixels.

Original file (614 × 948 pixels, file size: 55 KB, MIME type: image/jpeg)

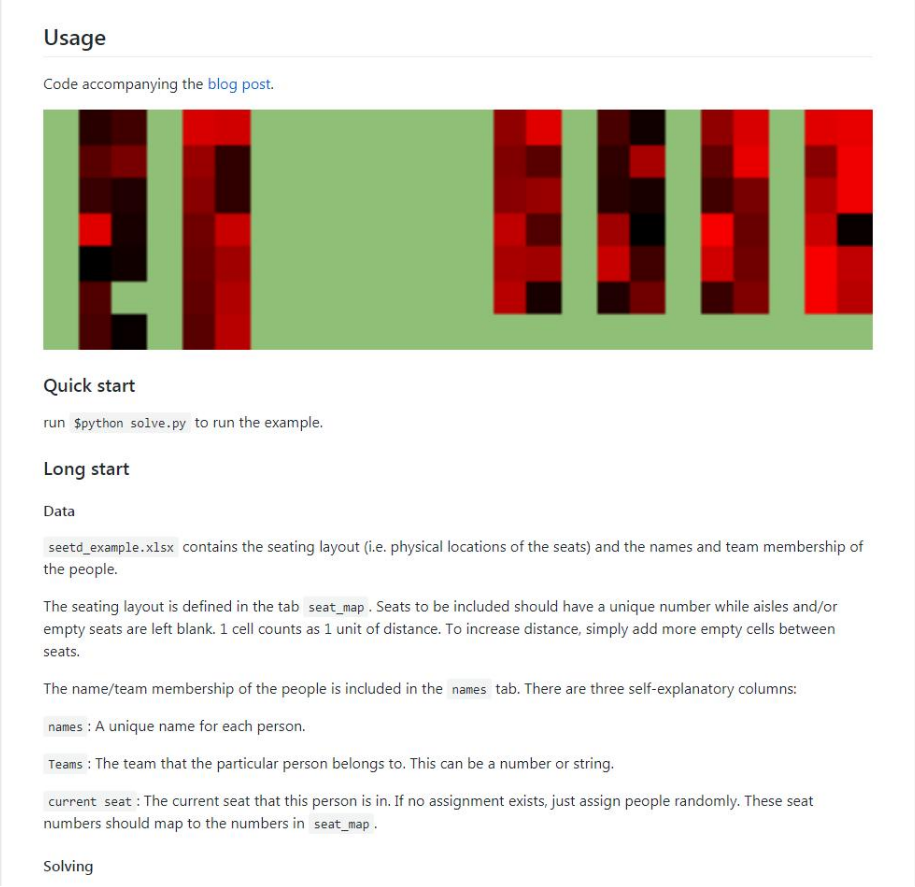
File history

Click on a date/time to view the file as it appeared at that time.

	Date/Time	Thumbnail	Dimensions	User	Comment
current	12:24, 7 March 2018		614 × 948 (55 KB)	Ianh (Talk | contribs)	

Uses for programming (specifically Python examples)

- **Make use of other people's code**
  - A seating preference optimizer
  - No executable download or web-based solution
  - Someone coded this algorithm in Python though
  - We can use it so long as we know how to run it



Usage

Code accompanying the blog post.

Quick start

run `$python solve.py` to run the example.

Long start

Data

`seetd_example.xlsx` contains the seating layout (i.e. physical locations of the seats) and the names and team membership of the people.

The seating layout is defined in the tab `seat_map`. Seats to be included should have a unique number while aisles and/or empty seats are left blank. 1 cell counts as 1 unit of distance. To increase distance, simply add more empty cells between seats.

The name/team membership of the people is included in the `names` tab. There are three self-explanatory columns:

`names` : A unique name for each person.

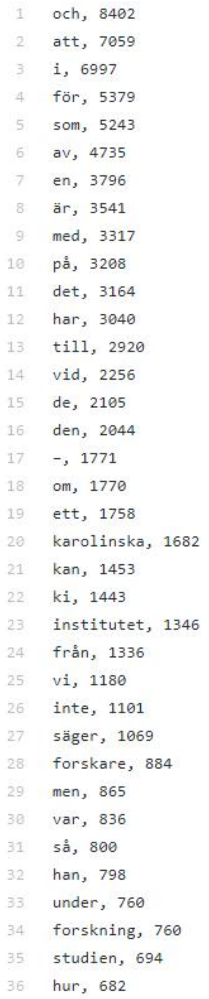
`Teams` : The team that the particular person belongs to. This can be a number or string.

`current seat` : The current seat that this person is in. If no assignment exists, just assign people randomly. These seat numbers should map to the numbers in `seat_map`.

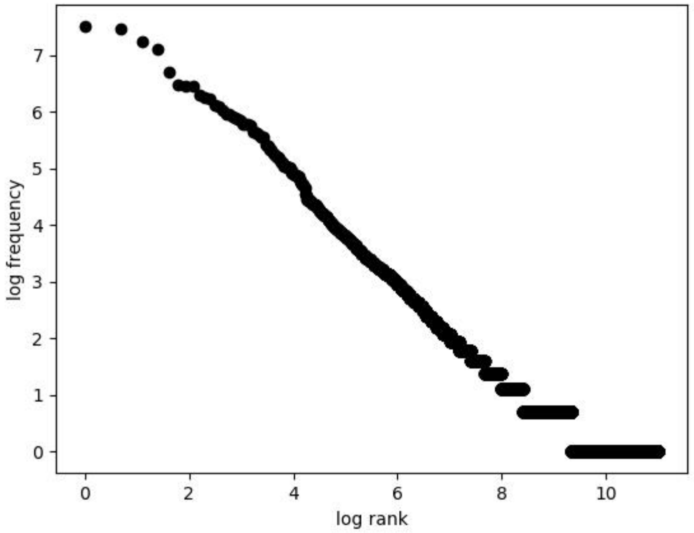
Solving

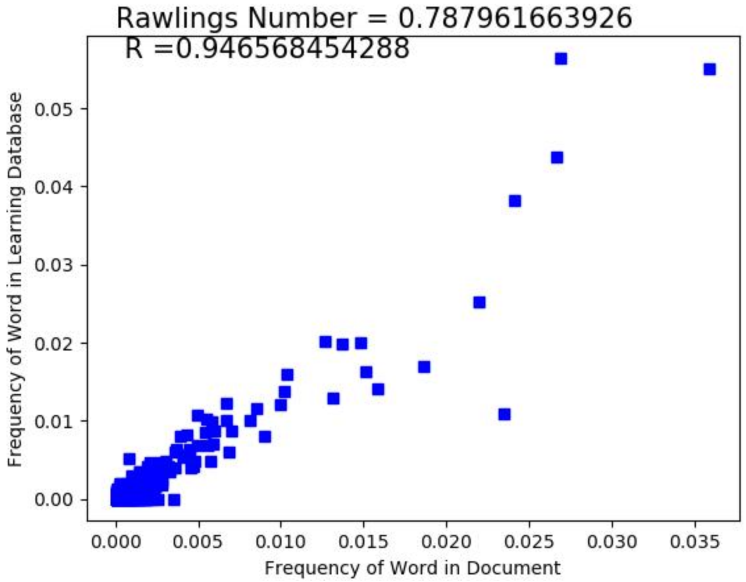
Uses for programming (specifically Python examples)

- **Manipulating and searching through text**
    - algorithm parses text and ranks words according to their frequency
    - useful for learning a new lan



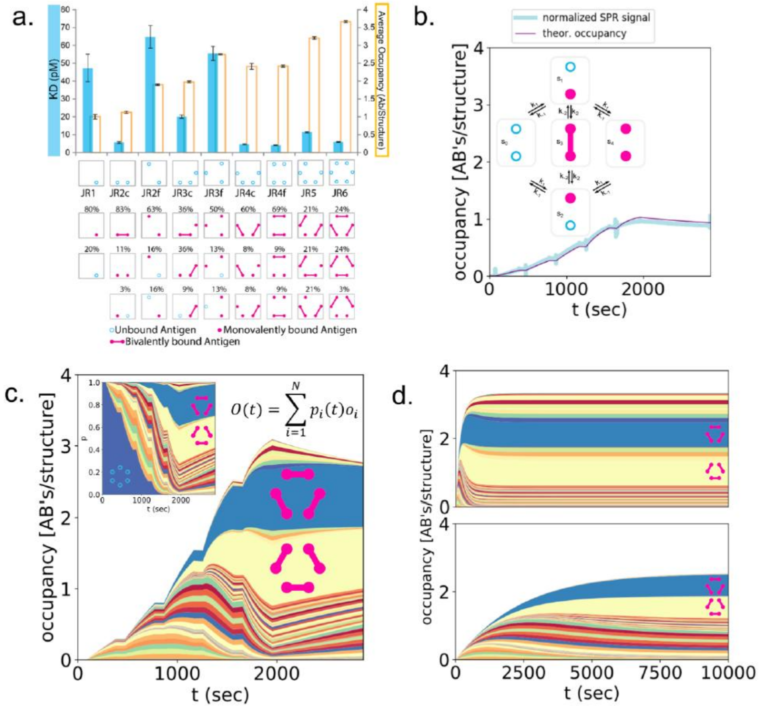
```
1   och, 8402
2   att, 7059
3   i, 6997
4   för, 5379
5   som, 5243
6   av, 4735
7   en, 3796
8   är, 3541
9   med, 3317
10  på, 3208
11  det, 3164
12  har, 3040
13  till, 2920
14  vid, 2256
15  de, 2105
16  den, 2044
17  -, 1771
18  om, 1770
19  ett, 1758
20  karolinska, 1682
21  kan, 1453
22  ki, 1443
23  institutet, 1346
24  från, 1336
25  vi, 1180
26  inte, 1101
27  säger, 1069
28  forskare, 884
29  men, 865
30  var, 836
31  så, 800
32  han, 798
33  under, 760
34  forskning, 760
35  studien, 694
36  hur, 682
```

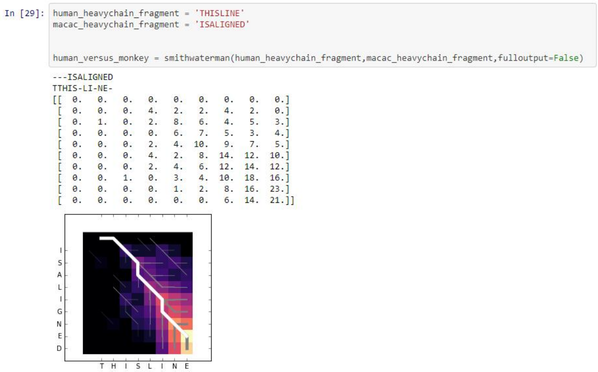




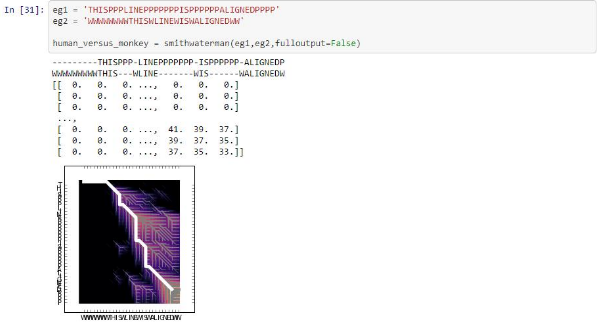
Uses for programming (specifically Python examples)

- **Applying and inventing creative data visualizations for science**

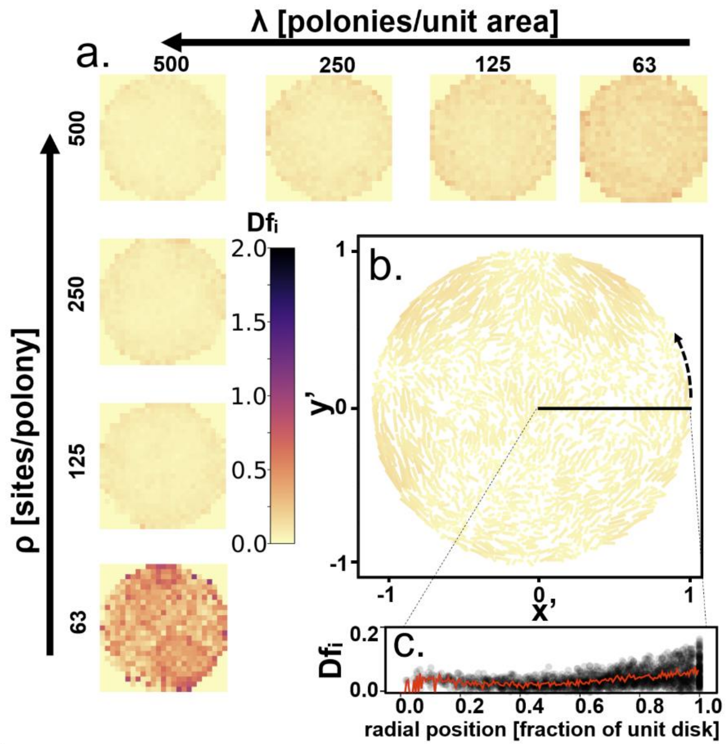




```
In [29]: human_heavychain_fragment = 'THISLINE'
         macac_heavychain_fragment = 'ISALIGNED'


         human_versus_monkey = smithwaterman(human_heavychain_fragment,macac_heavychain_fragment,fulloutput=False)

---ISALIGNED
TTHIS-LI-NE-
[[  0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   4.   2.   2.   4.   2.   0.]
 [  0.   1.   0.   2.   8.   6.   4.   5.   3.]
 [  0.   0.   0.   0.   6.   7.   5.   3.   4.]
 [  0.   0.   0.   2.   4.  10.   9.   7.   5.]
 [  0.   0.   0.   4.   2.   8.  14.  12.  10.]
 [  0.   0.   0.   2.   4.   6.  12.  14.  12.]
 [  0.   0.   1.   0.   3.   4.  10.  18.  16.]
 [  0.   0.   0.   0.   1.   2.   8.  16.  23.]
 [  0.   0.   0.   0.   0.   0.   6.  14.  21.]]
```

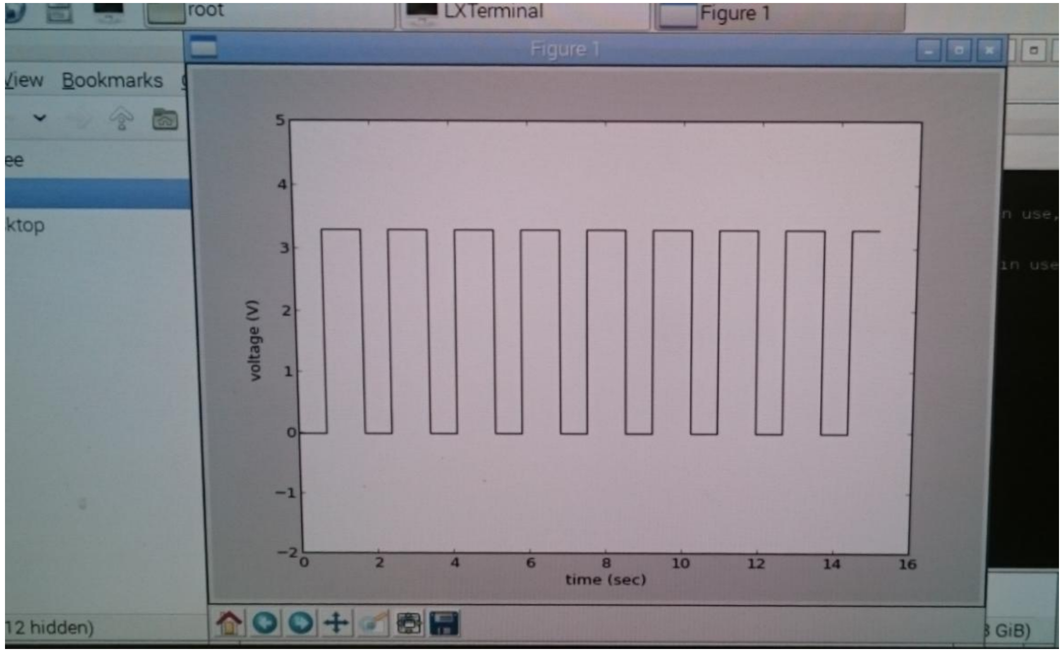


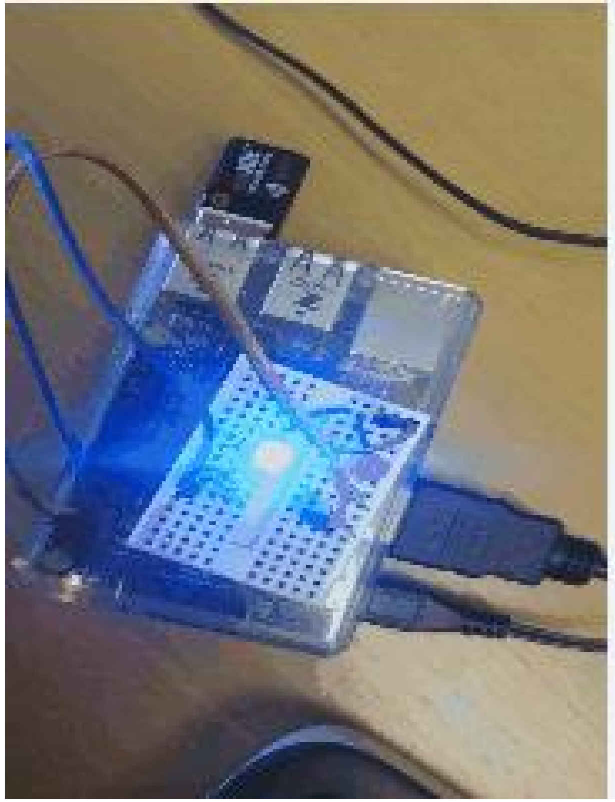
```
In [31]: eg1 = 'THISPPPLINEPPPPPPPISPPPPPPALIGNEDPPPP'
         eg2 = 'WWWWWWWWTHISWLINEWISWALIGNEDWW'

         human_versus_monkey = smithwaterman(eg1,eg2,fulloutput=False)

---------THISPPP-LINEPPPPPPP-ISPPPPPP-ALIGNEDP
WWWWWWWWTHIS---WLINE-------WIS------WALIGNEDW
[[  0.   0.   0. ...,   0.   0.   0.]
 [  0.   0.   0. ...,   0.   0.   0.]
 [  0.   0.   0. ...,   0.   0.   0.]
 ...,
 [  0.   0.   0. ...,  41.  39.  37.]
 [  0.   0.   0. ...,  39.  37.  35.]
 [  0.   0.   0. ...,  37.  35.  33.]]
```



Uses for programming (specifically Python examples)

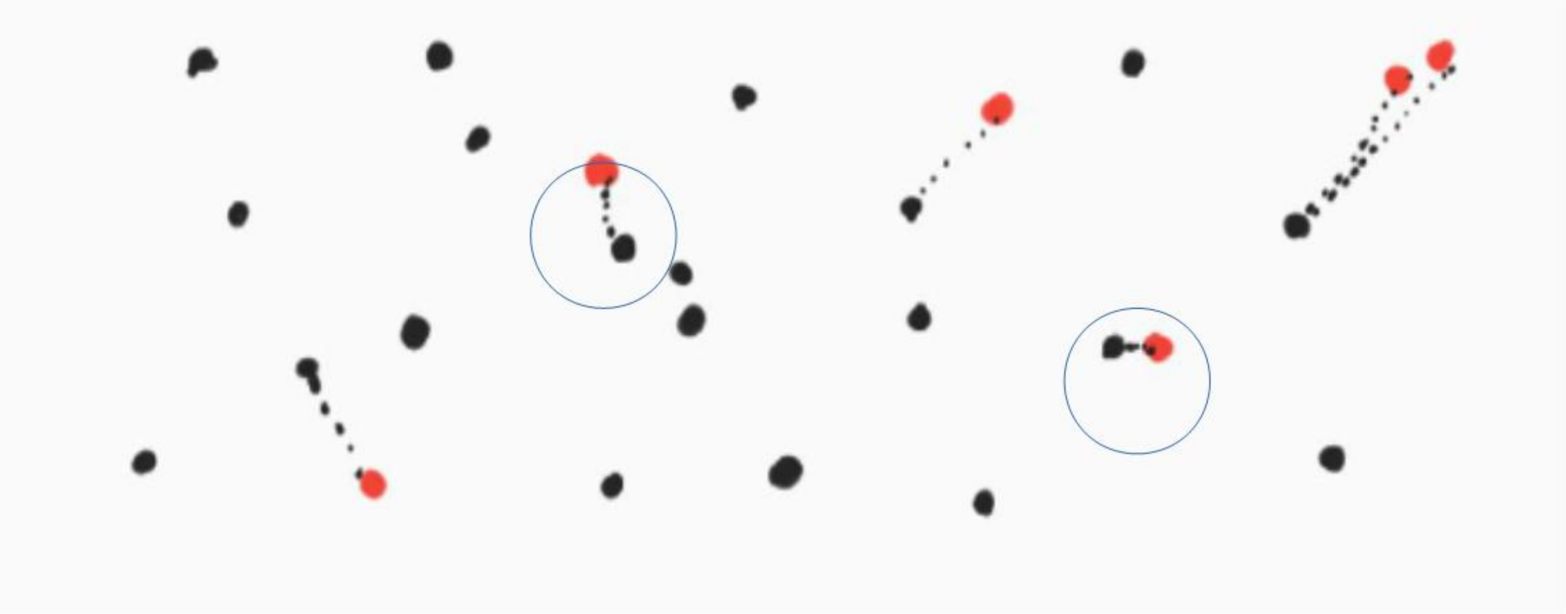
- **Controlling hardware**
  - an LED that flashes with a desired frequency to stimulate light-sensitive proteins





Uses for programming (specifically Python examples)

- **Simulating things especially when we don't know the math**
  - many scientific questions are easier to simulate than derive an analytical expression for
  - e.g. for a given density of randomly placed red dots and black dots, what is the fraction of pairs that land within distance x of each other?



Uses for programming (specifically Python examples)

- **Simulating things especially when we don't know the math**
  - many scientific questions are easier to simulate than derive an analytical expression for
  - e.g. for a given density of randomly placed red dots and black dots, what is the fraction of pairs that land within distance x of each other?

Mathematician's approach: derive what's known as the nearest neighbor distribution using calculus and probability theory

Imagine the bin of width dr of a histogram, what is its height for the nearest neighbor dist?

$$w(r)dr = \{P\ no\ point\ within\ r\}\{P\ point\ within\ r + dr\}$$

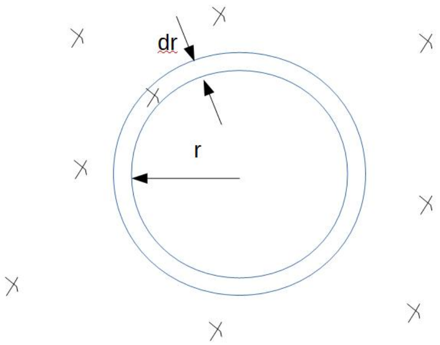
$$P\ no\ point\ within\ r = 1 - \int_0^r w(r)dr$$

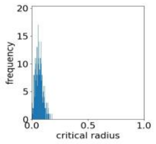
$$P\ point\ with\ r + dr = 2\pi r dr\lambda$$

$$w(r)dr = \left(1 - \int_0^r w(r)dr\right)2\pi r dr\lambda$$

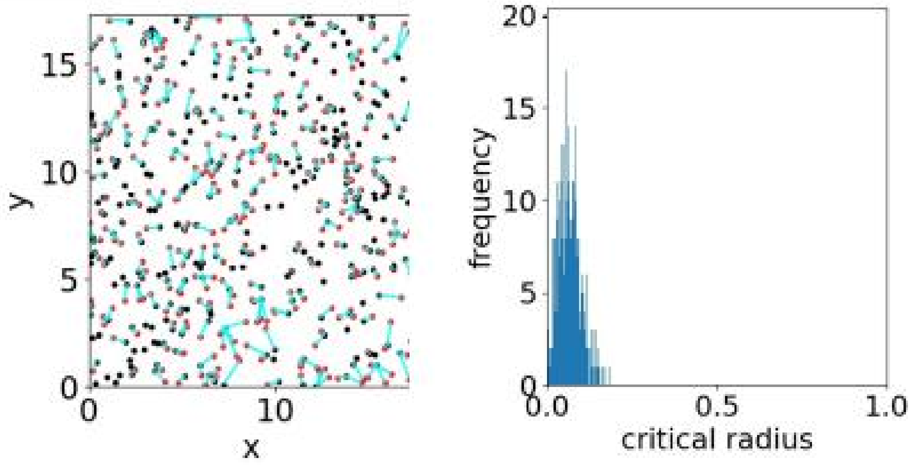
$$w(r) = 2\pi r\lambda e^{-\pi r^2\lambda}$$

$$CDF = \int_{-\infty}^{r_0} w(r)dr = 1 - e^{-\pi r^2\lambda}$$





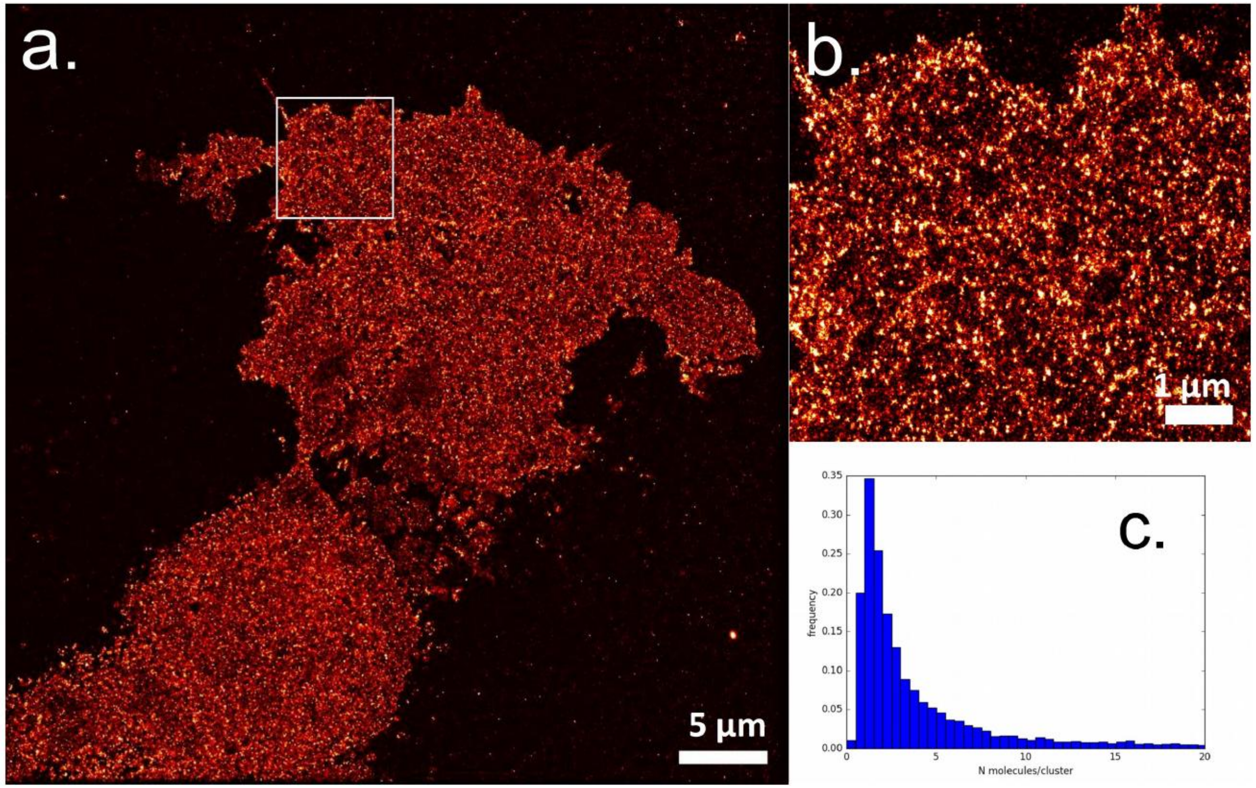
Programmer's approach: Generate two sets of coordinates and compute the distances between them



Uses for programming (specifically Python examples)

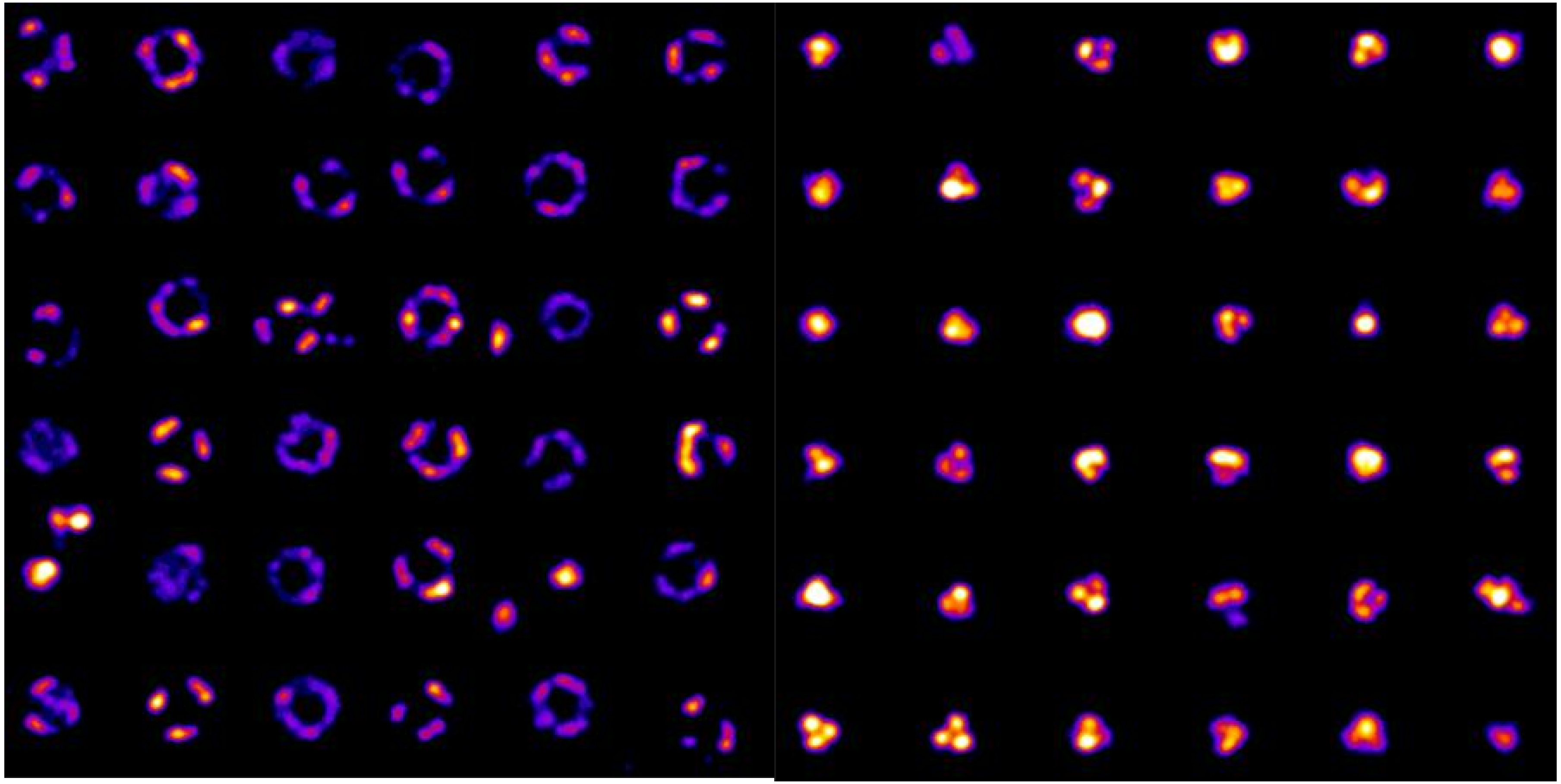
- **Process images!**
  - reconstructing super resolution microscopy data
  - analyzing the images to detect certain features automatically
  - remove human bias by having the machine do it

Surface receptors distributed at the nanoscale. (A) An SKBR3 breast cancer cell tagged with an oligonucleotide-affibody conjugate targeting HER2 receptors. DNA-PAINT was used to achieve approximately 20 nm resolution, revealing clusters of proteins rather than a uniform distribution. scale 5 µm. (B) 5x inset, scale bar 1 µm (C) Cluster size distribution determined via DBSCAN algorithm implemented in custom Python code.



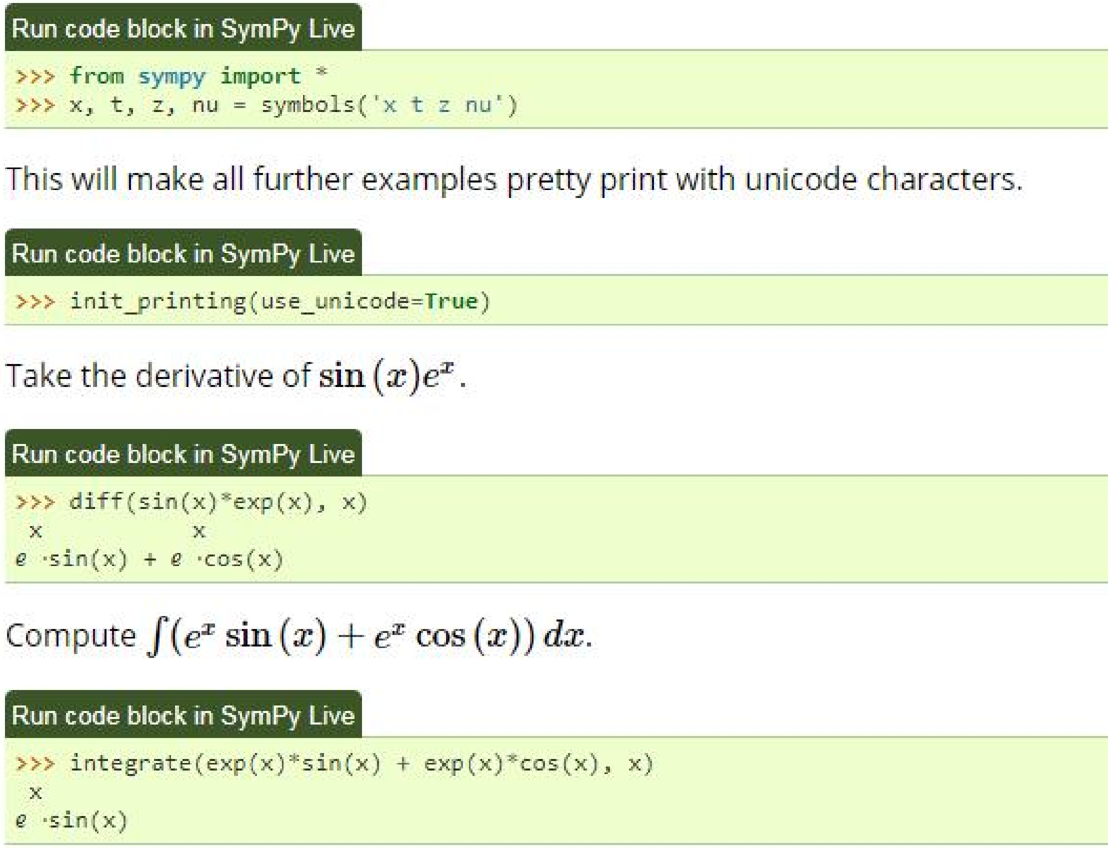
Uses for programming (specifically Python examples)

- **Machine learning/classification**
    - pick out structures in images and
    - group structures into 2D classes



Uses for programming (specifically Python examples)

- **Do symbolic math/algebra/calculus**
  - solving expressions like you would on paper
  - similar to Wolfram Alpha or Mathematica
  - free and integrated with the rest of Python



Run code block in SymPy Live

```
>>> from sympy import *
>>> x, t, z, nu = symbols('x t z nu')
```

This will make all further examples pretty print with unicode characters.

Run code block in SymPy Live

```
>>> init_printing(use_unicode=True)
```

Take the derivative of $\sin(x)e^x$.

Run code block in SymPy Live

```
>>> diff(sin(x)*exp(x), x)
 x           x
ℯ ⋅sin(x) + ℯ ⋅cos(x)
```

Compute $\int (e^x \sin(x) + e^x \cos(x))\, dx$.

Run code block in SymPy Live

```
>>> integrate(exp(x)*sin(x) + exp(x)*cos(x), x)
 x
ℯ ⋅sin(x)
```

Python compared to other languages

- **Python is an Interpreted language**
  - Commands are executed by an interpreter
  - Interpreter has subroutines already for translating new code into machine language
  - Means that time is spent on translation during the running
  - Python is thus slower as a result!
  - Syntax is easier to learn, code is more readable
- **Compiled languages (e.g. C++, C, Java)**
  - A step is taken before running a new program to convert the code into machine code
  - Ultimately leads to faster performance
  - Syntax is “closer to the machine” and thus more complex!
  - Useful for big software projects and under-the-hood applications
  - Most python libraries like numpy are written in precompiled code like C++
- **Python is a general language**
  - Some languages are optimized for certain tasks and can be worth using in certain contexts e.g. R, matlab, mathematica...
  - general languages have the advantage of being able to bring different specialties together

Industry usage

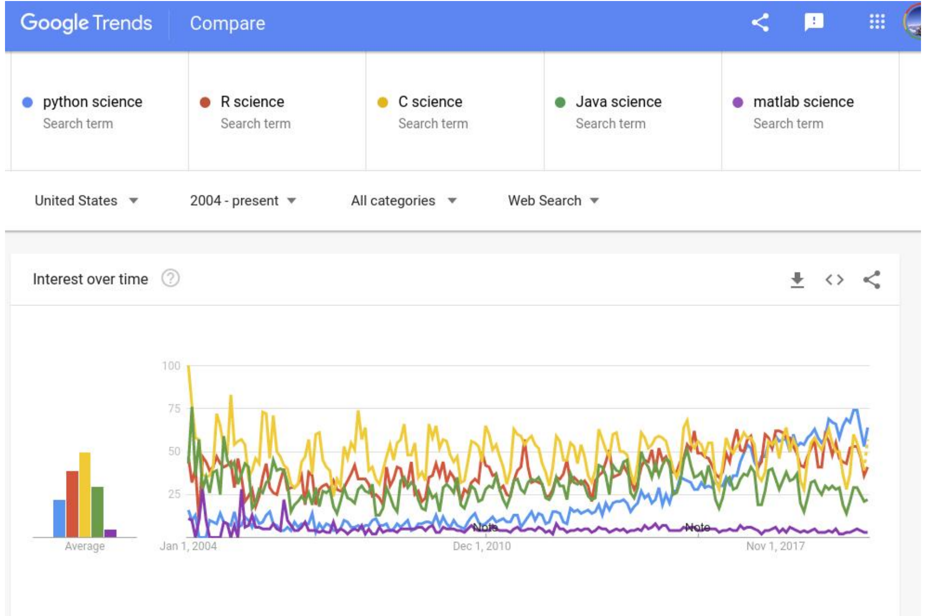
- **Google**
  - “Python where we can, C++ where we must”
  - an official server-side language along with C++, Java, and Go
  - Google’s very first web-crawling spider was first written in Java 1.0 and was so difficult that they rewrote it into Python. -Steven Levy “In the Plex”
- **Spotify**
  - uses a combination of Python and C++ for backend framework
  - uses Python for analytics - a module called Luigi
  - preferred because of the fast development pipeline
- **Reddit**
  - site was originally coded in Lisp - recoded into Python in 2005 shortly after launch
  - “There’s a library for everything. We’ve been learning a lot of these technologies and a lot of these architectures as we go. And, so, when I don’t understand connection pools, I can just find a library until I understand it better myself and write our own. Don’t understand web frameworks, so we’ll use someone else’s until we make our own…Python has an awesome crutch like that.” - Steve Huffman
- **Others big companies using Python**
  - Facebook, Quora, Dropbox, Netflix, Isntagram...

source: https://realpython.com/world-class-companies-using-python/#spotify

Prevalence in science

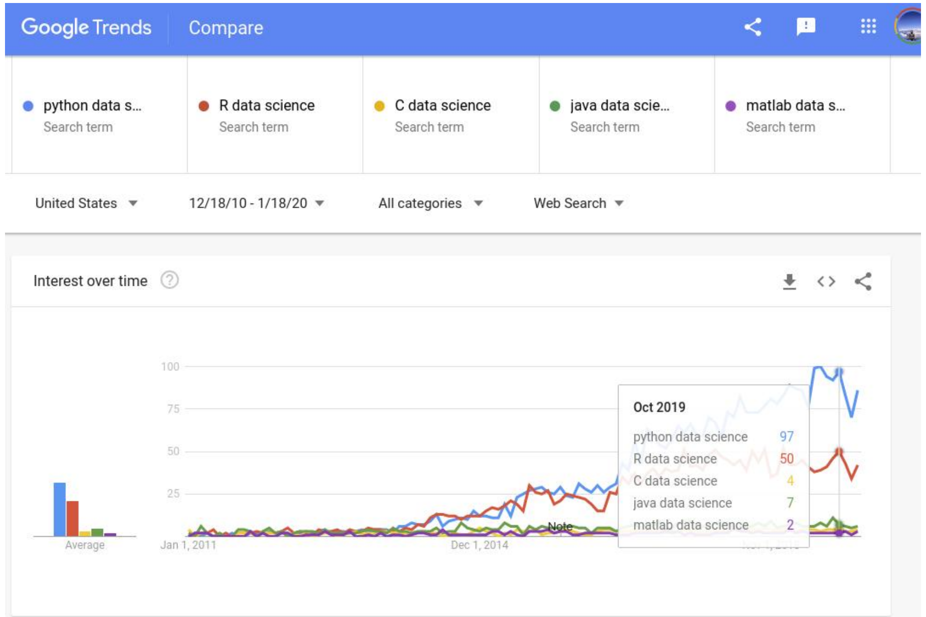
- **Fast prototyping pipeline is ideal for science**
  - less focus on end-product software for users
  - more focus on getting an answer, visualizing data, inventing new algorithms

- **Large and growing free opensource community**
  - more libraries due to large user base
  - more resources to get help
  - crowd-sourced maintenance rather than centralized maintenance by commercial developers (e.g. Matlab or MS Excel VBA)



● python science Search term
● R science Search term
● C science Search term
● Java science Search term
● matlab science Search term

United States ▾ 2004 - present ▾ All categories ▾ Web Search ▾



● python data s... Search term
● R data science Search term
● C data science Search term
● java data scie... Search term
● matlab data s... Search term

United States ▾ 12/18/10 - 1/18/20 ▾ All categories ▾ Web Search ▾

Oct 2019

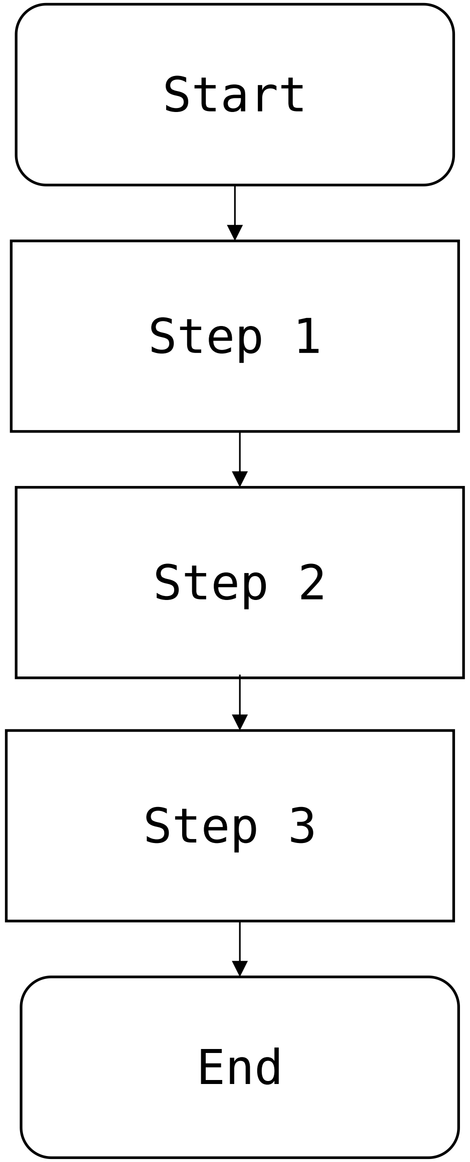
python data science	97
R data science	50
C data science	4
java data science	7
matlab data science	2

The Case for Learning Programming as Scientists/Engineers

- Freedom to build any tool that you need
- Professional caliber capability for free
- Socially active community of users and developers
- Easy to learn other languages once you know one
- A medium for learning (especially new math concepts)
- Understanding and reproducing other scientists' work
- Participate in our era - computing/information are the defininig features of today

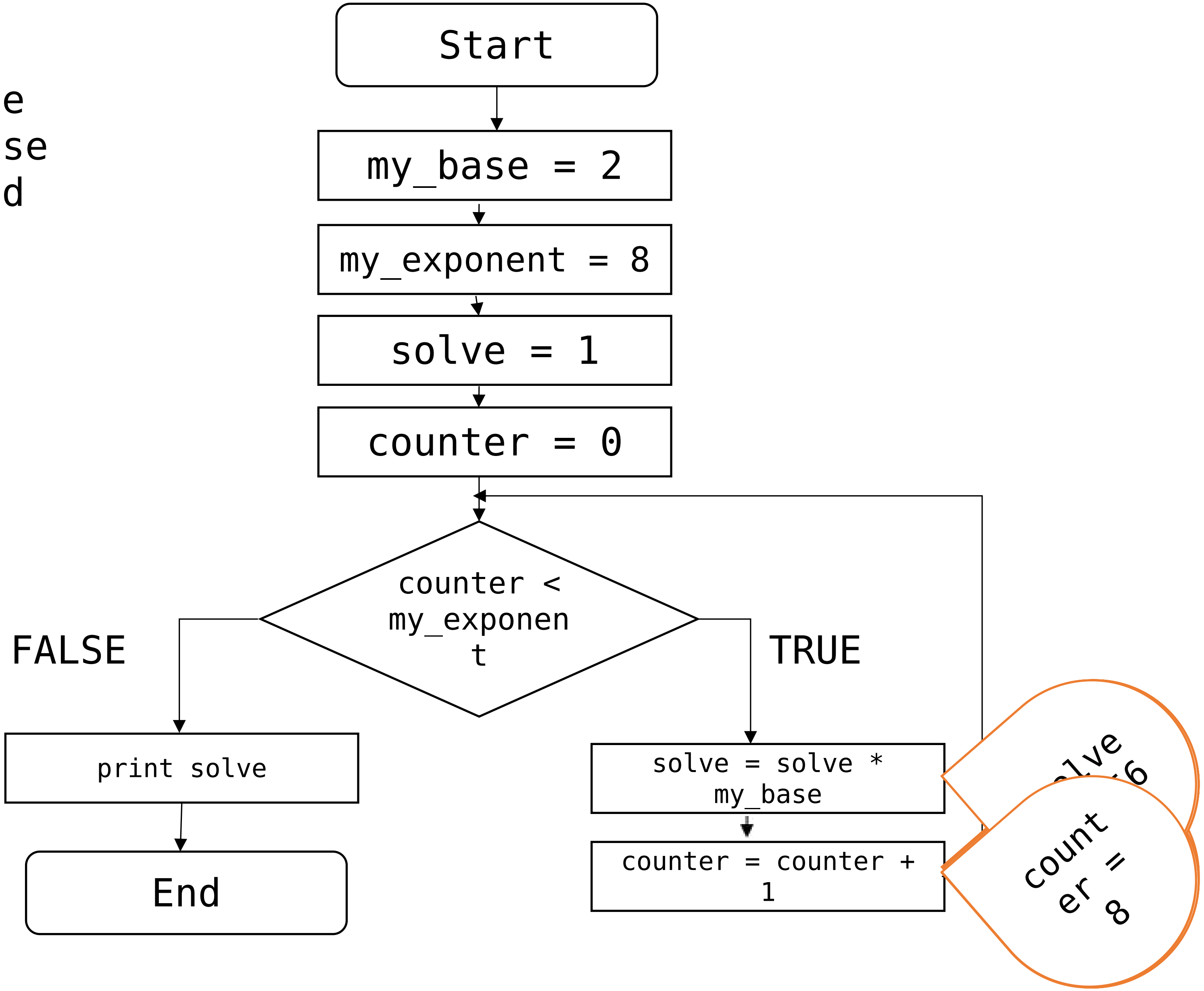
The anatomy of a program

- **An input or initial state**
- **A series of steps**
  - steps are carried out in order one after the other
  - each step modifies the state
- **An output or final state**



Majority of real programs have decisions and loops

write a program
that computes the
solution to a base
number (2) raised
to a power (8)



Start

my_base = 2

my_exponent = 8

solve = 1

counter = 0

counter <
my_exponen
t

FALSE

print solve

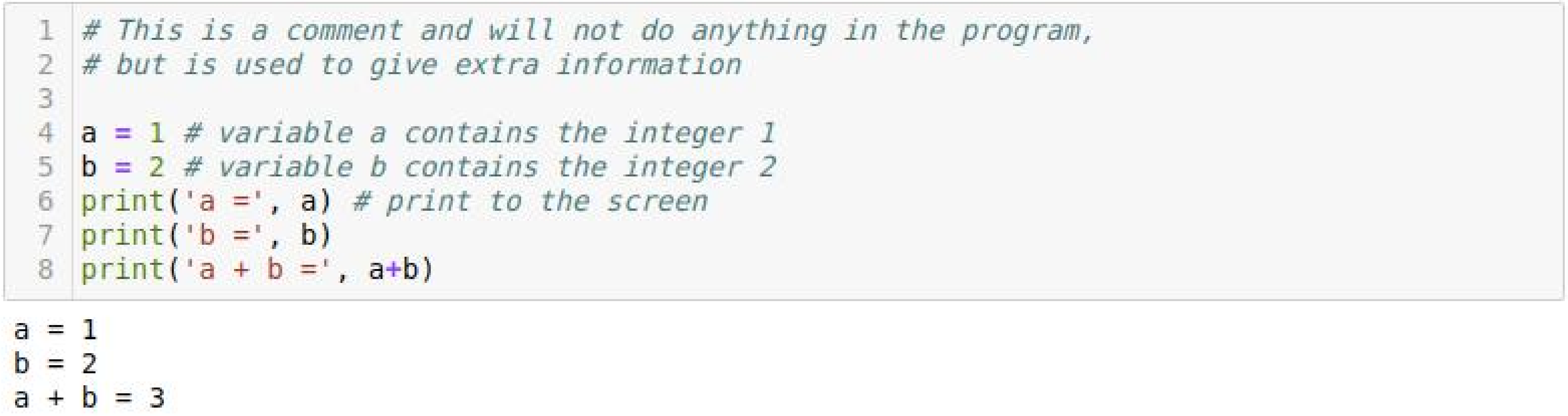
End

TRUE

solve = solve *
my_base

counter = counter +
1

Python variables get defined when you assign them a value

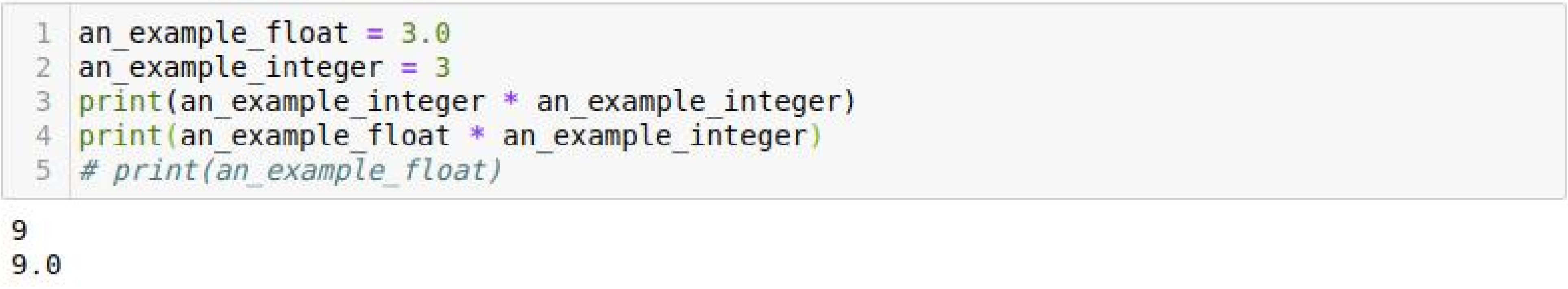


```python
# This is a comment and will not do anything in the program,
# but is used to give extra information

a = 1 # variable a contains the integer 1
b = 2 # variable b contains the integer 2
print('a =', a) # print to the screen
print('b =', b)
print('a + b =', a+b)
```

```
a = 1
b = 2
a + b = 3
```

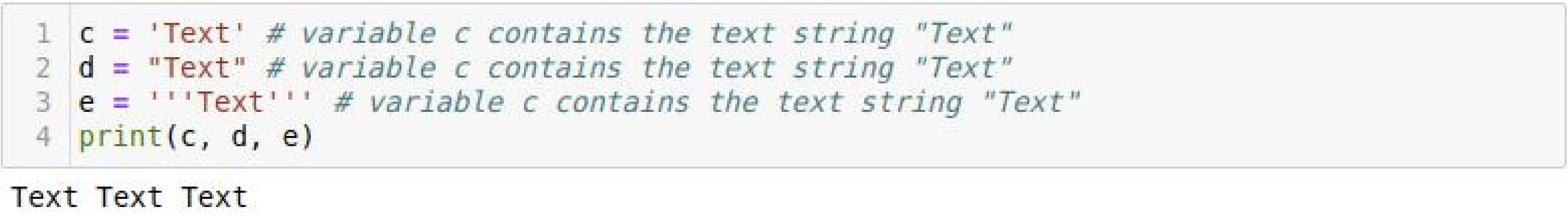
Variables can be defined using different data types

- **integers, floats, strings**



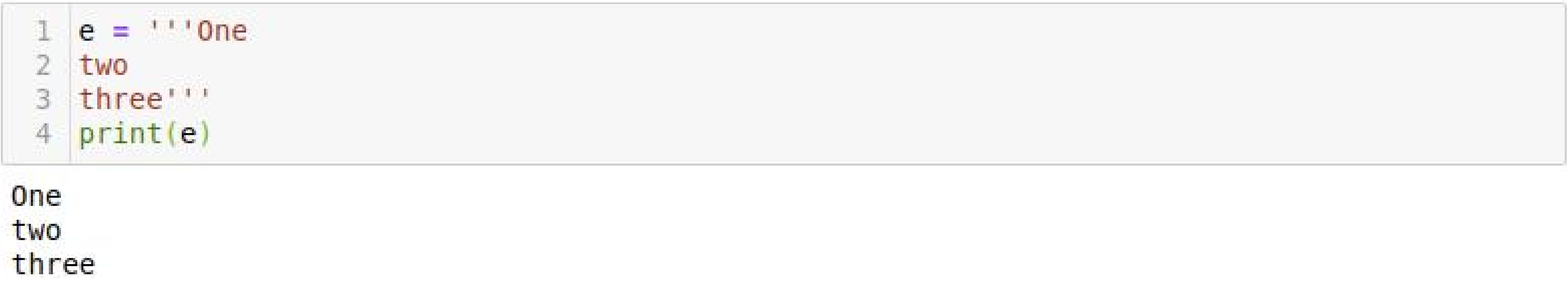
```python
an_example_float = 3.0
an_example_integer = 3
print(an_example_integer * an_example_integer)
print(an_example_float * an_example_integer)
# print(an_example_float)
```

```
9
9.0
```



```python
c = 'Text' # variable c contains the text string "Text"
d = "Text" # variable c contains the text string "Text"
e = '''Text''' # variable c contains the text string "Text"
print(c, d, e)
```

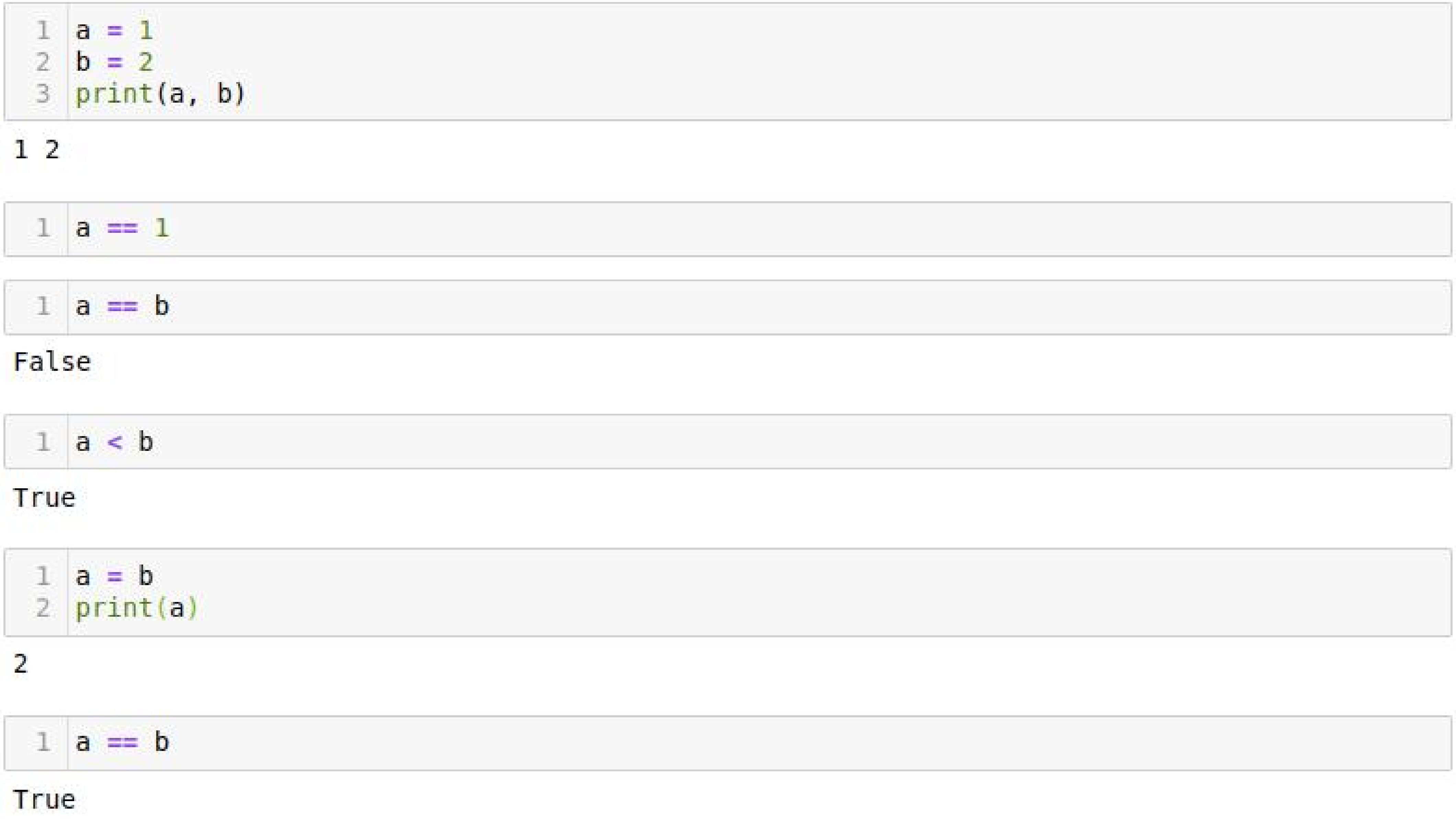
```
Text Text Text
```



```python
e = '''One
two
three'''
print(e)
```

```
One
two
three
```

Two kinds of equal signs: definition and evaluation



```
a = 1
b = 2
print(a, b)
```

```
1 2
```

```
a == 1
```

```
a == b
```

```
False
```

```
a < b
```

```
True
```

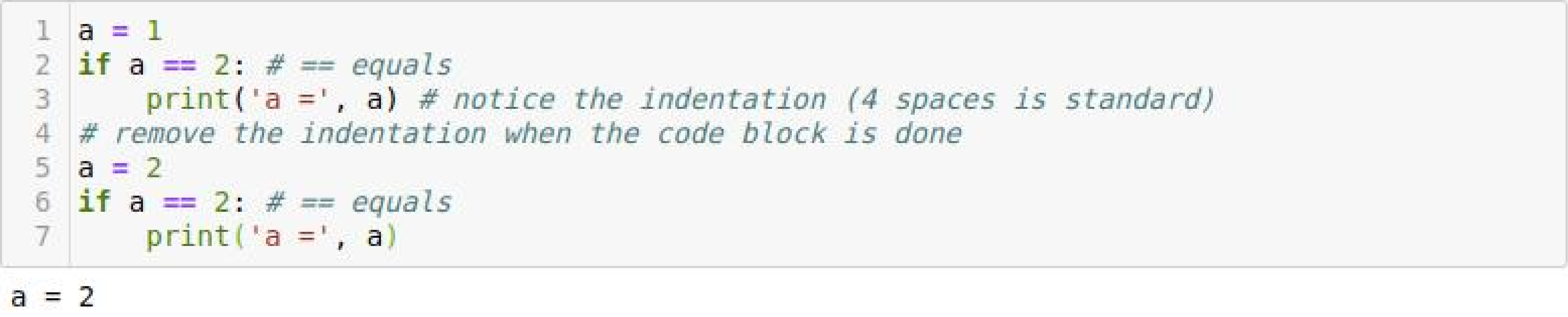
```
a = b
print(a)
```

```
2
```

```
a == b
```

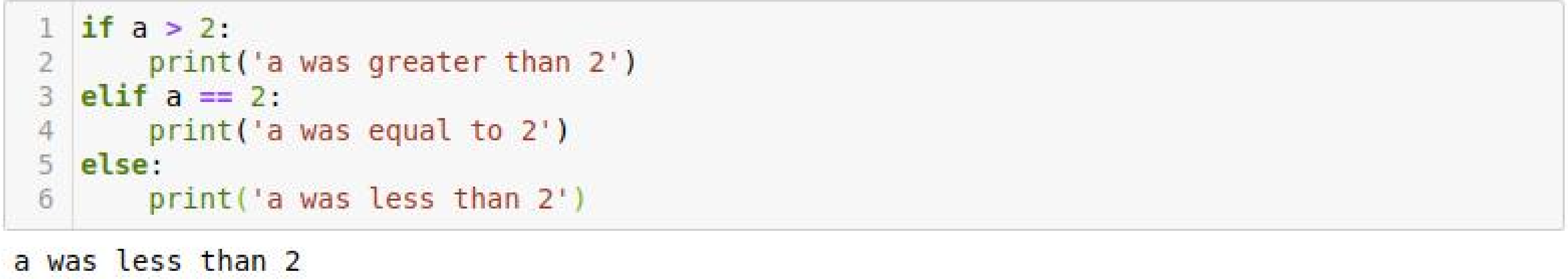
```
True
```

Conditional Statements (Decisions) and indentation syntax



```
a = 1
if a == 2: # == equals
    print('a =', a) # notice the indentation (4 spaces is standard)
# remove the indentation when the code block is done
a = 2
if a == 2: # == equals
    print('a =', a)
```

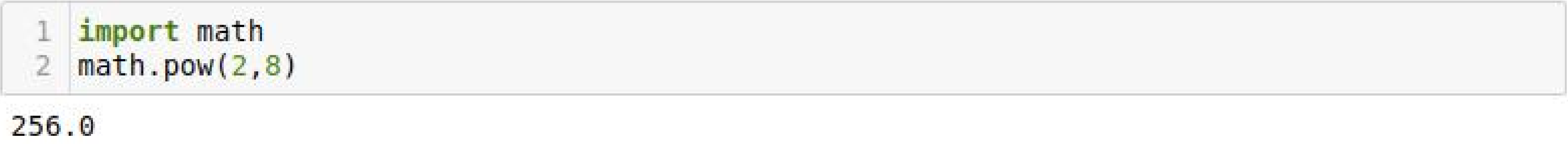
```
a = 2
```



```
if a > 2:
    print('a was greater than 2')
elif a == 2:
    print('a was equal to 2')
else:
    print('a was less than 2')
```

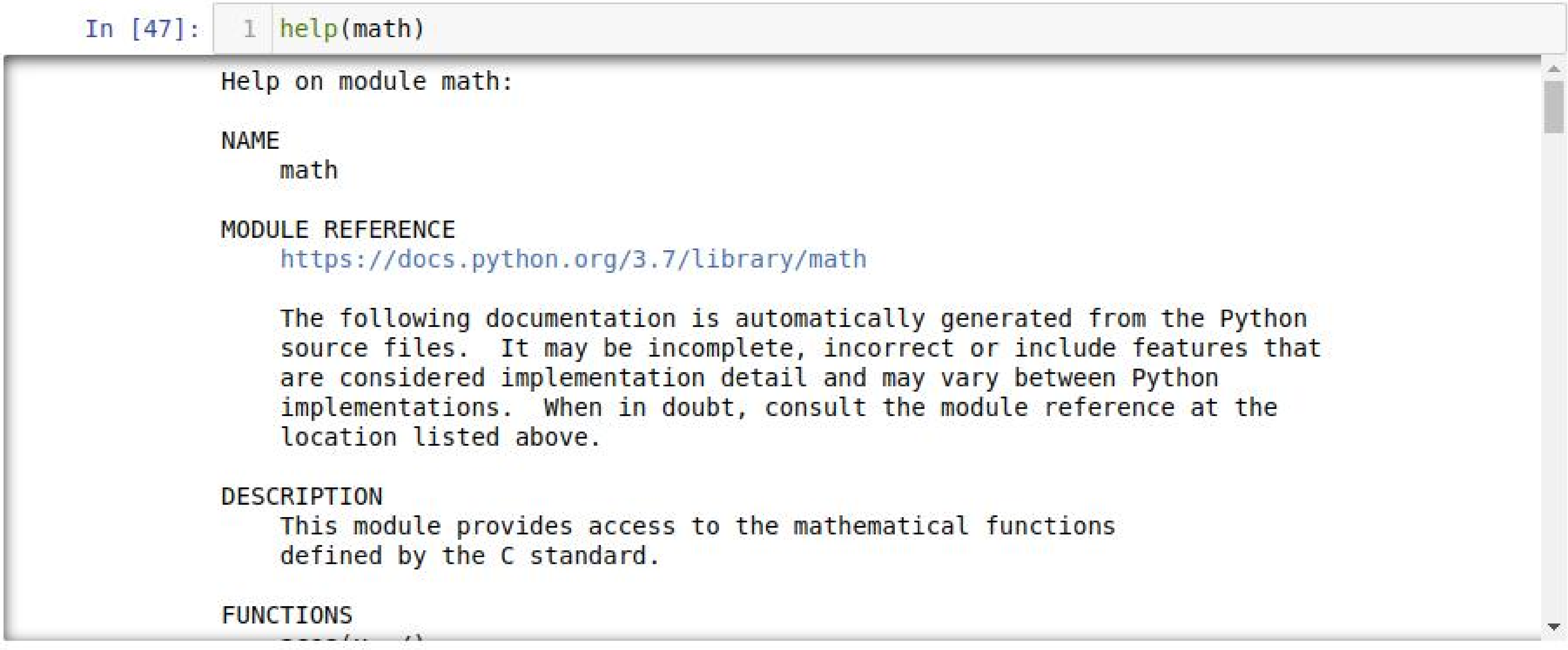
```
a was less than 2
```

Python modules, packages, and libraries



```python
import math
math.pow(2,8)
```

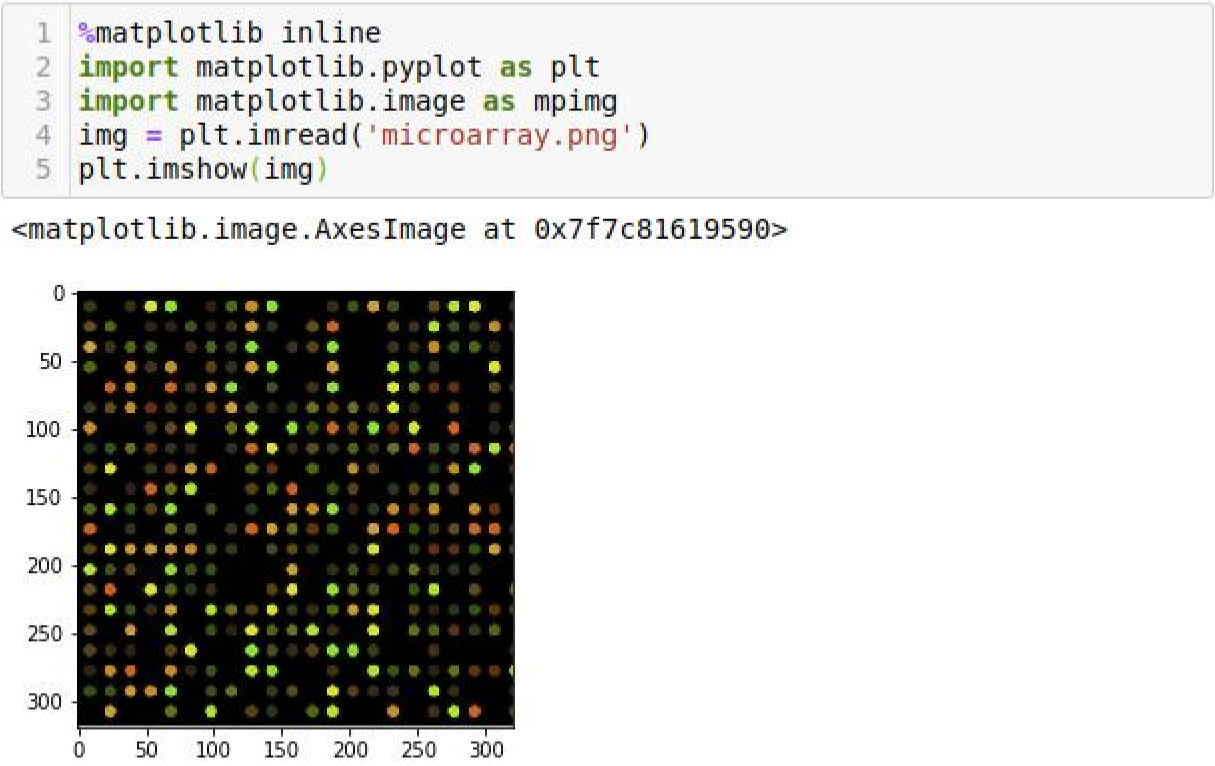
```
256.0
```



```python
help(math)
```

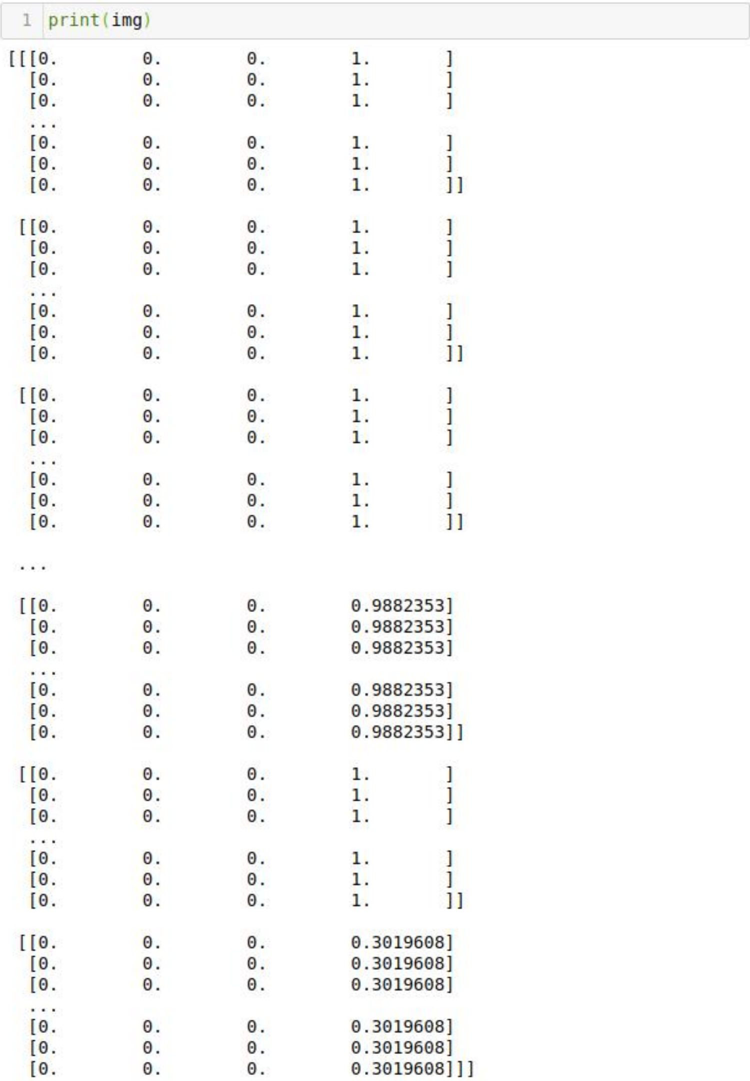
```
Help on module math:

NAME
    math

MODULE REFERENCE
    https://docs.python.org/3.7/library/math

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
```

Python modules, packages, and libraries



```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
img = plt.imread('microarray.png')
plt.imshow(img)
```

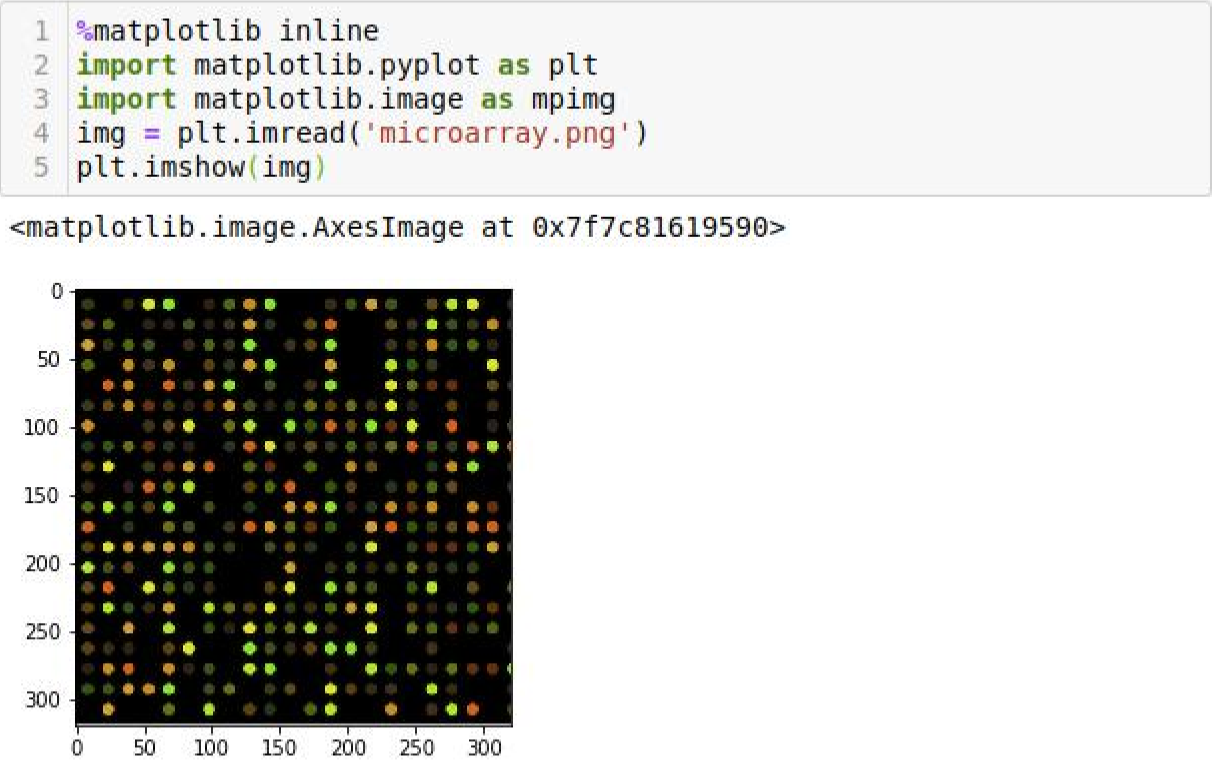
```
<matplotlib.image.AxesImage at 0x7f7c81619590>
```



```
print(img)
```

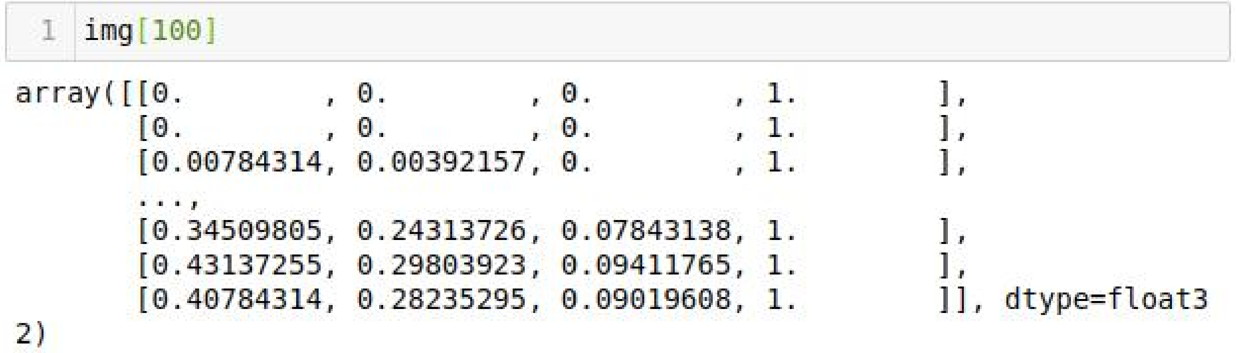
```
[[[0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  ...
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]]

 [[0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  ...
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]]

 [[0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  ...
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]]

 ...

 [[0.         0.         0.         0.9882353]
  [0.         0.         0.         0.9882353]
  [0.         0.         0.         0.9882353]
  ...
  [0.         0.         0.         0.9882353]
  [0.         0.         0.         0.9882353]
  [0.         0.         0.         0.9882353]]

 [[0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  ...
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]
  [0.         0.         0.         1.        ]]

 [[0.         0.         0.         0.3019608]
  [0.         0.         0.         0.3019608]
  [0.         0.         0.         0.3019608]
  ...
  [0.         0.         0.         0.3019608]
  [0.         0.         0.         0.3019608]
  [0.         0.         0.         0.3019608]]]
```

Python modules, packages, and libraries



```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
img = plt.imread('microarray.png')
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f7c81619590>
```



```
img[100]
```

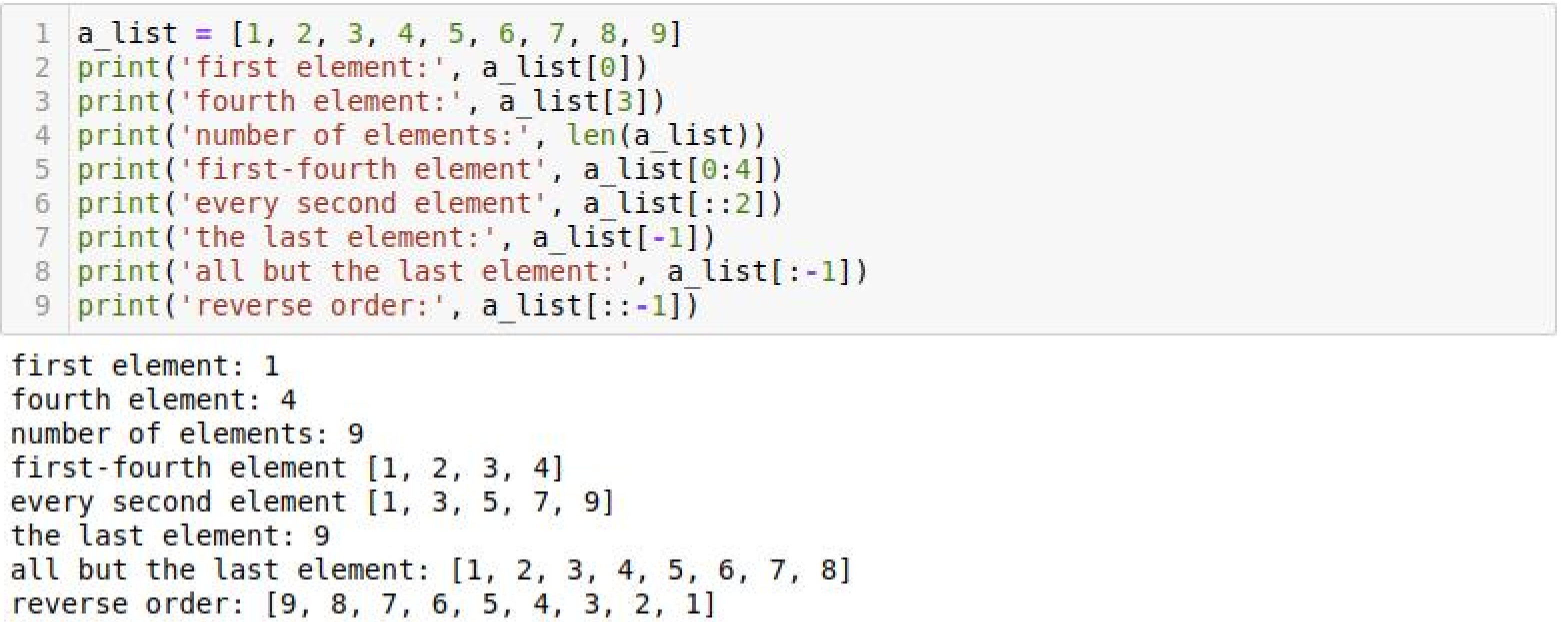
```
array([[0.        , 0.        , 0.        , 1.        ],
       [0.        , 0.        , 0.        , 1.        ],
       [0.00784314, 0.00392157, 0.        , 1.        ],
       ...,
       [0.34509805, 0.24313726, 0.07843138, 1.        ],
       [0.43137255, 0.29803923, 0.09411765, 1.        ],
       [0.40784314, 0.28235295, 0.09019608, 1.        ]], dtype=float3
2)
```

Python modules, packages, and libraries

cool or useful libraries to know about:

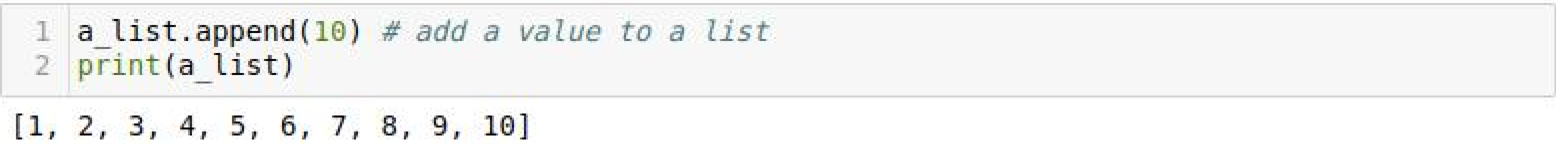
- numpy
  - essential for all numerical problems, plotting, data management
- scipy
  - lots of statistics, machine learning, and useful mathematical functions
  - implement them first, understand them second - great way to learn new math
- networkx
  - library for generating and visualizing networks/graphs
- biopython
  - library for dealing with biological sequence data
- matplotlib
  - essential for dealing with images, plotting, making figures for publications, animations...
- random
  - functions for generating random numbers - very handy for simulation
- os
  - short for “operating system” - very handy for manipulating files - loading them, writing them, copying and pasting etc

Lists



```python
a_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print('first element:', a_list[0])
print('fourth element:', a_list[3])
print('number of elements:', len(a_list))
print('first-fourth element', a_list[0:4])
print('every second element', a_list[::2])
print('the last element:', a_list[-1])
print('all but the last element:', a_list[:-1])
print('reverse order:', a_list[::-1])
```

```
first element: 1
fourth element: 4
number of elements: 9
first-fourth element [1, 2, 3, 4]
every second element [1, 3, 5, 7, 9]
the last element: 9
all but the last element: [1, 2, 3, 4, 5, 6, 7, 8]
reverse order: [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

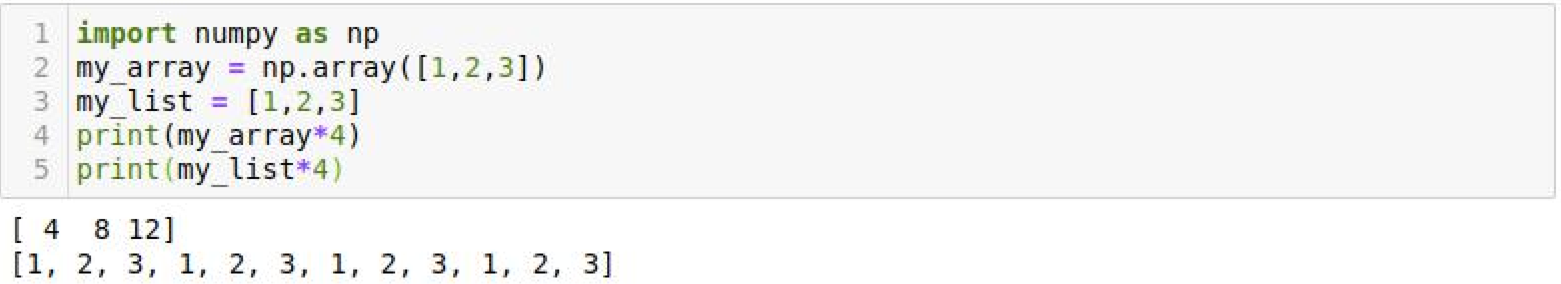


```python
a_list.append(10) # add a value to a list
print(a_list)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

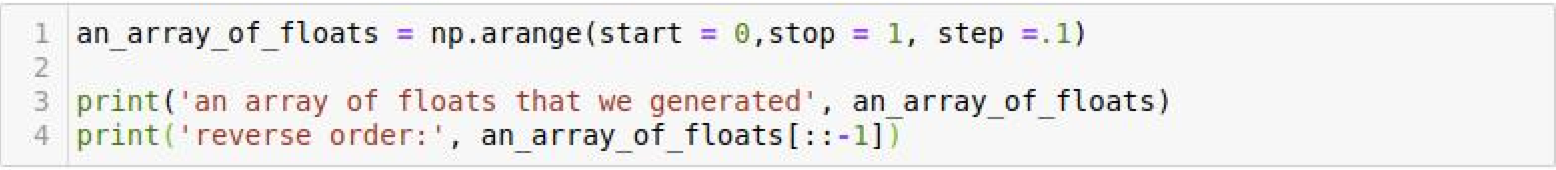
Numpy - a library for arrays and matrices

- numpy for numerical/mathematical operations, linear algebra, matrix operations
- lists for organization, looping
- a lot of overlap and conversion between them



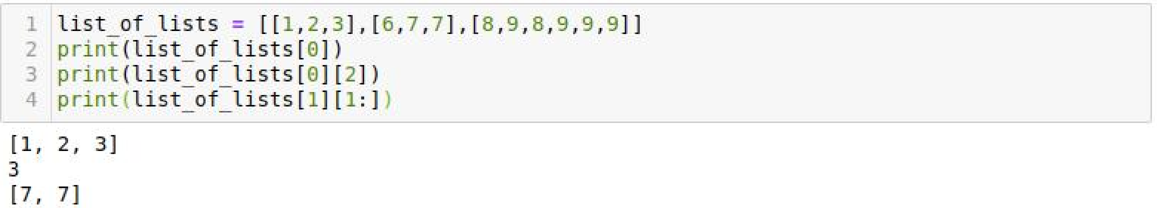
```python
import numpy as np
my_array = np.array([1,2,3])
my_list = [1,2,3]
print(my_array*4)
print(my_list*4)
```

```
[ 4  8 12]
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```



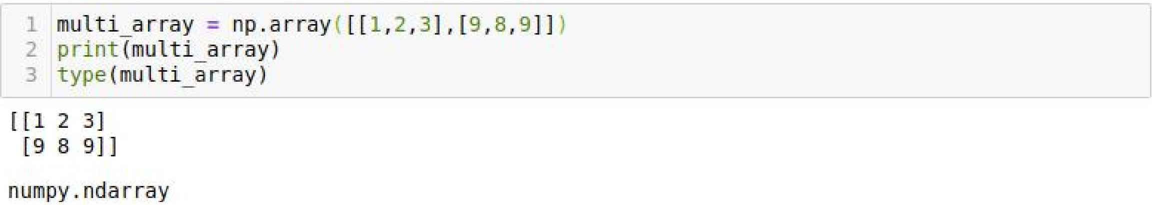
```python
an_array_of_floats = np.arange(start = 0,stop = 1, step =.1)

print('an array of floats that we generated', an_array_of_floats)
print('reverse order:', an_array_of_floats[::-1])
```

Multidimensional arrays and lists



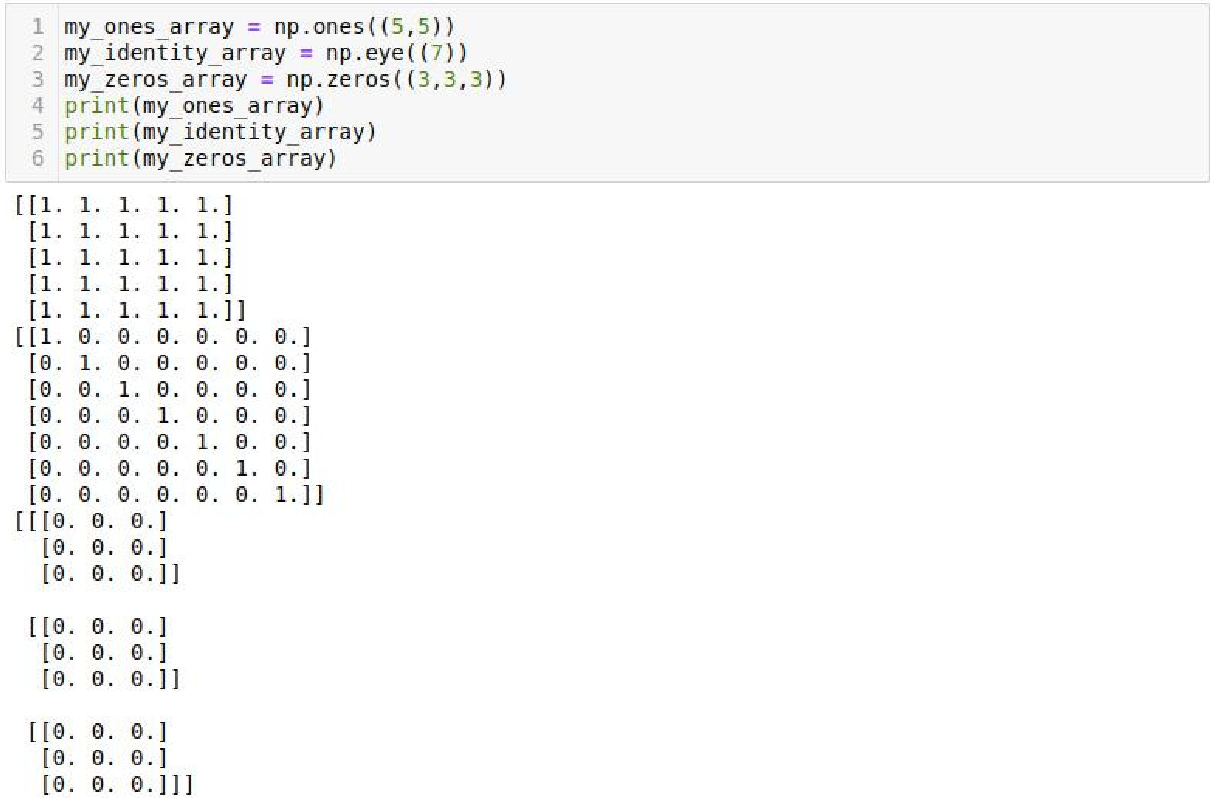
```
list_of_lists = [[1,2,3],[6,7,7],[8,9,8,9,9,9]]
print(list_of_lists[0])
print(list_of_lists[0][2])
print(list_of_lists[1][1:])
```

```
[1, 2, 3]
3
[7, 7]
```



```
multi_array = np.array([[1,2,3],[9,8,9]])
print(multi_array)
type(multi_array)
```

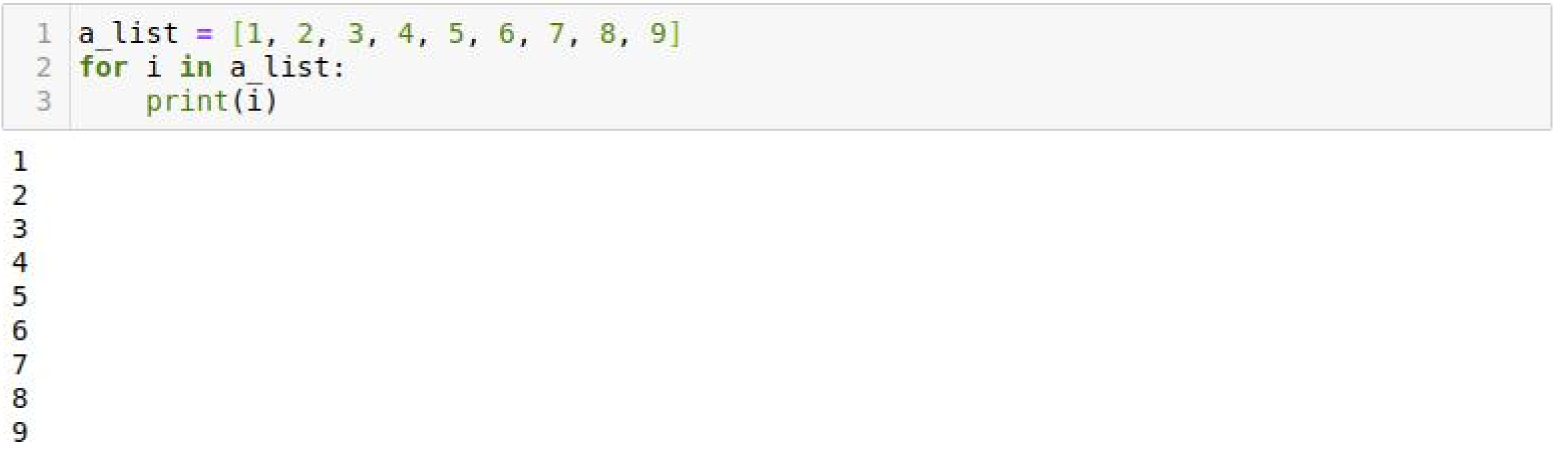
```
[[1 2 3]
 [9 8 9]]

numpy.ndarray
```



```
my_ones_array = np.ones((5,5))
my_identity_array = np.eye((7))
my_zeros_array = np.zeros((3,3,3))
print(my_ones_array)
print(my_identity_array)
print(my_zeros_array)
```

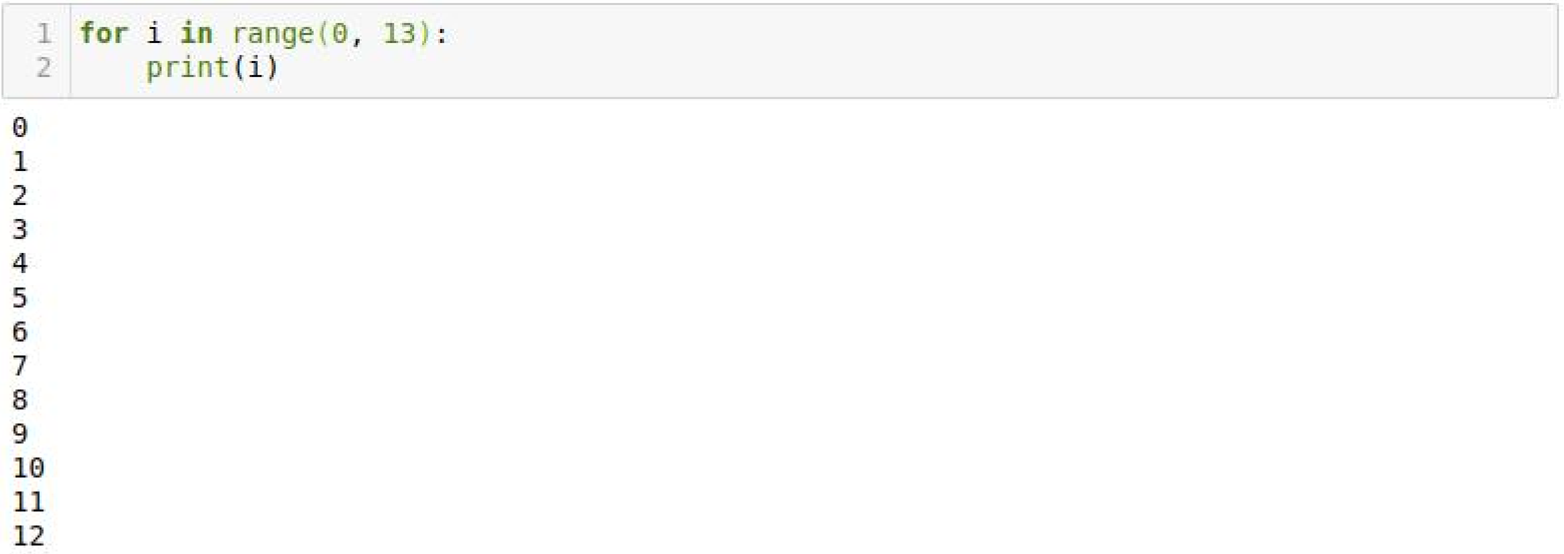
```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
[[1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1.]]
[[[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]

 [[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]

 [[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]]
```

for Loops



```python
a_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
for i in a_list:
    print(i)
```

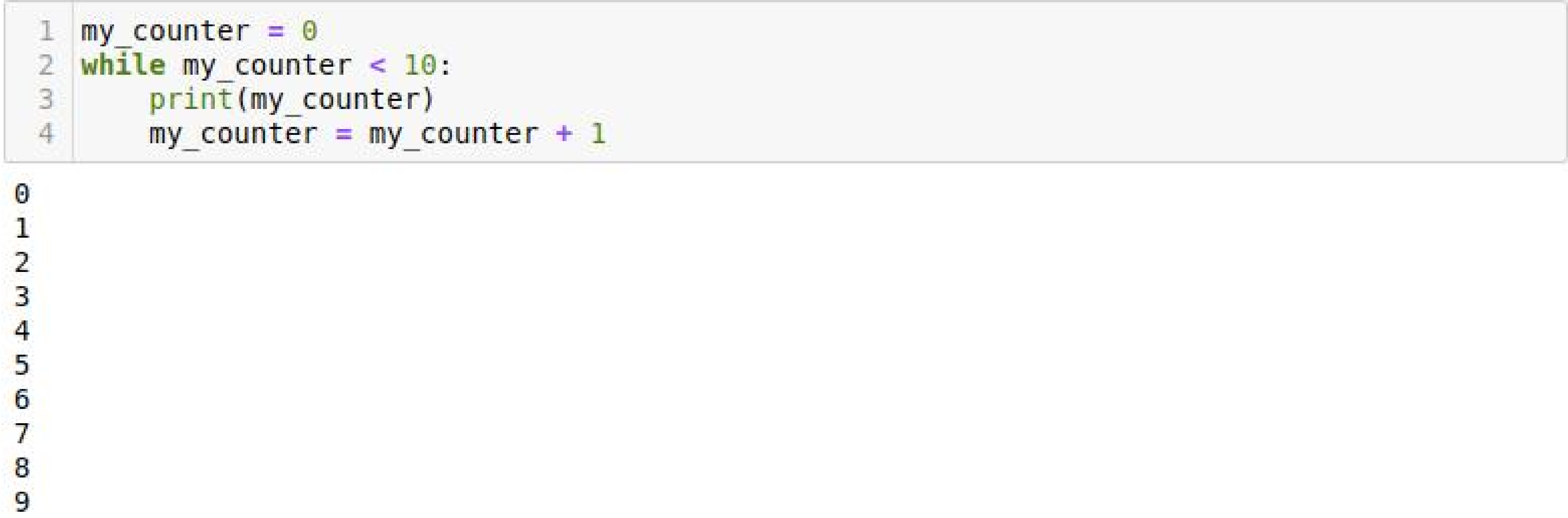
```
1
2
3
4
5
6
7
8
9
```



```python
for i in range(0, 13):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
```

while Loops

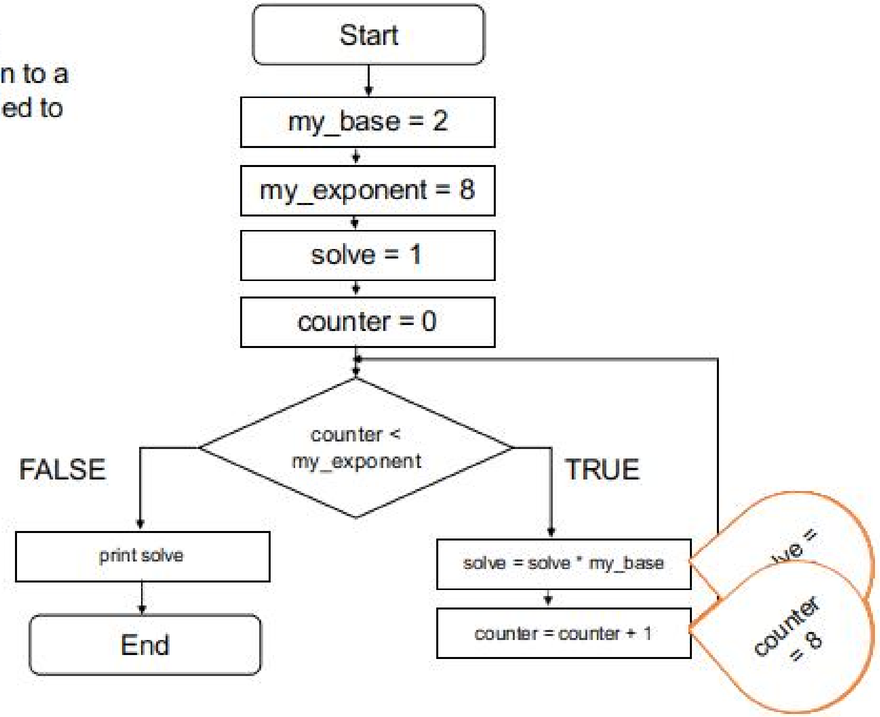


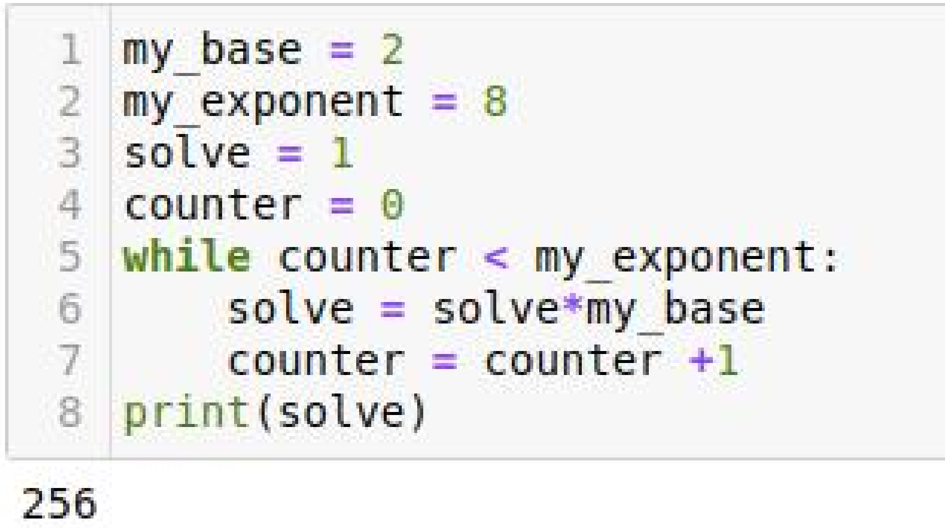
```python
my_counter = 0
while my_counter < 10:
    print(my_counter)
    my_counter = my_counter + 1
```

```
0
1
2
3
4
5
6
7
8
9
```

Implementing our exponentiator

write a program that computes the solution to a base number (2) raised to a power (8)

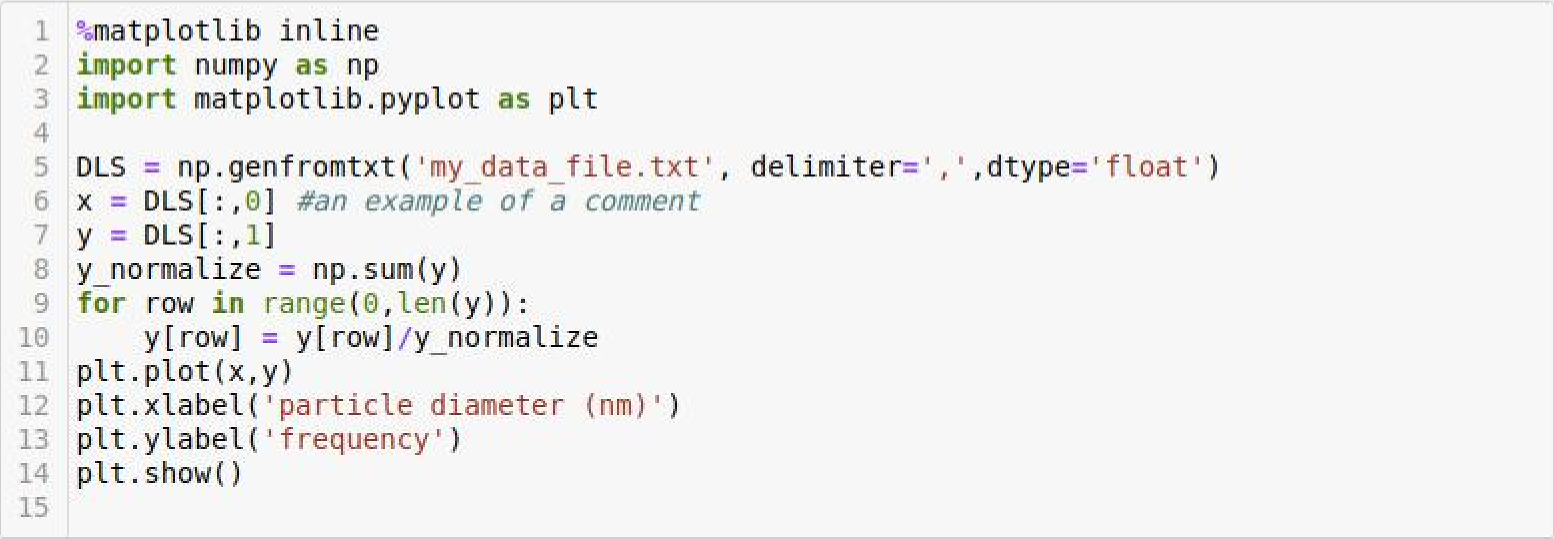




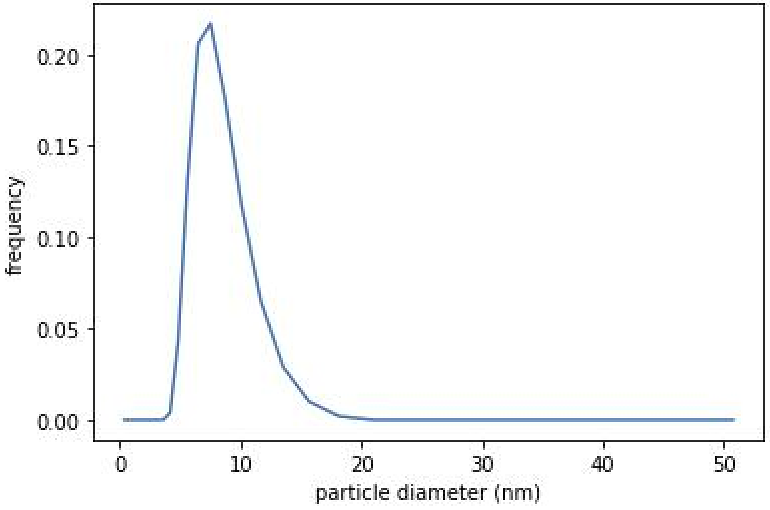
```
my_base = 2
my_exponent = 8
solve = 1
counter = 0
while counter < my_exponent:
     solve = solve*my_base
     counter = counter +1
print(solve)
```

```
256
```

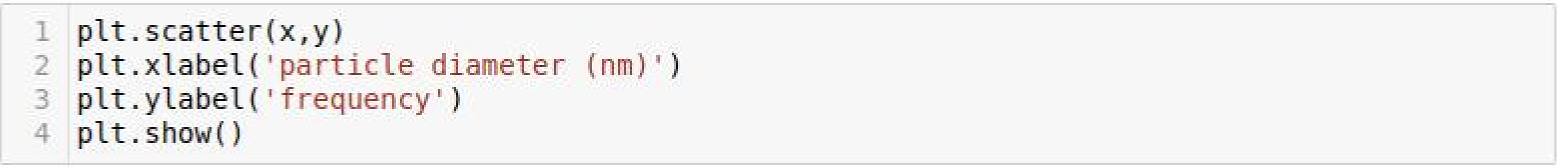
Scientific programming - data and plotting



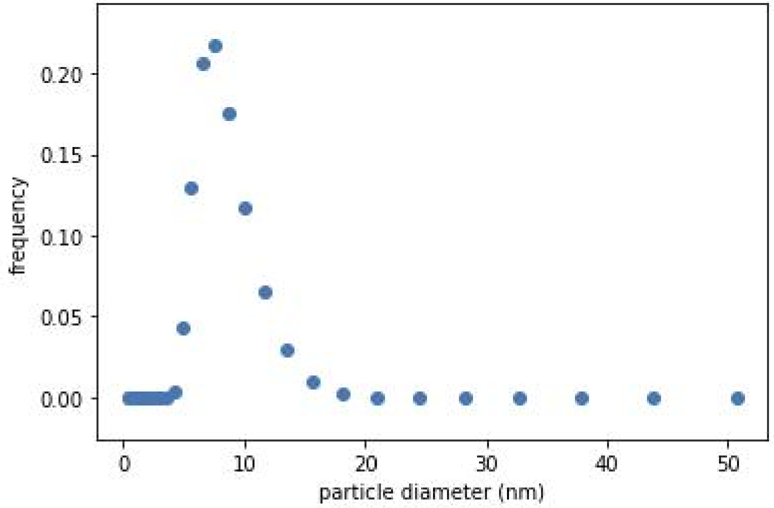
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

DLS = np.genfromtxt('my_data_file.txt', delimiter=',',dtype='float')
x = DLS[:,0] #an example of a comment
y = DLS[:,1]
y_normalize = np.sum(y)
for row in range(0,len(y)):
    y[row] = y[row]/y_normalize
plt.plot(x,y)
plt.xlabel('particle diameter (nm)')
plt.ylabel('frequency')
plt.show()

```



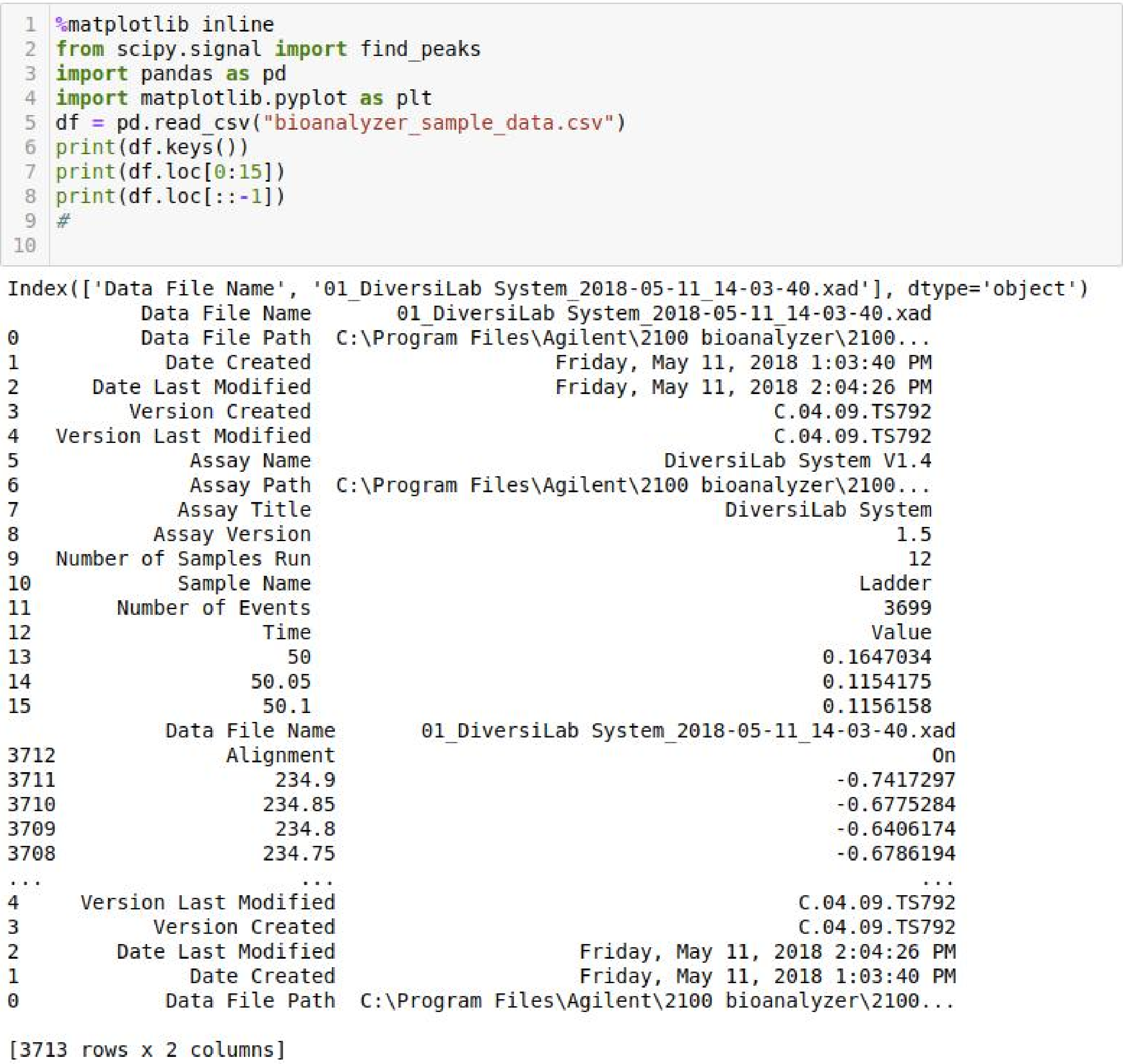
Scientific programming - data and plotting



```
plt.scatter(x,y)
plt.xlabel('particle diameter (nm)')
plt.ylabel('frequency')
plt.show()
```



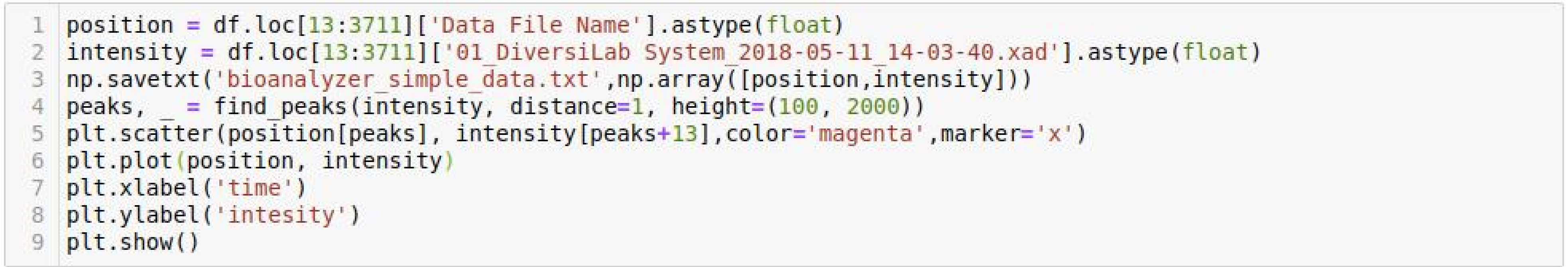
Scientific programming - data and plotting



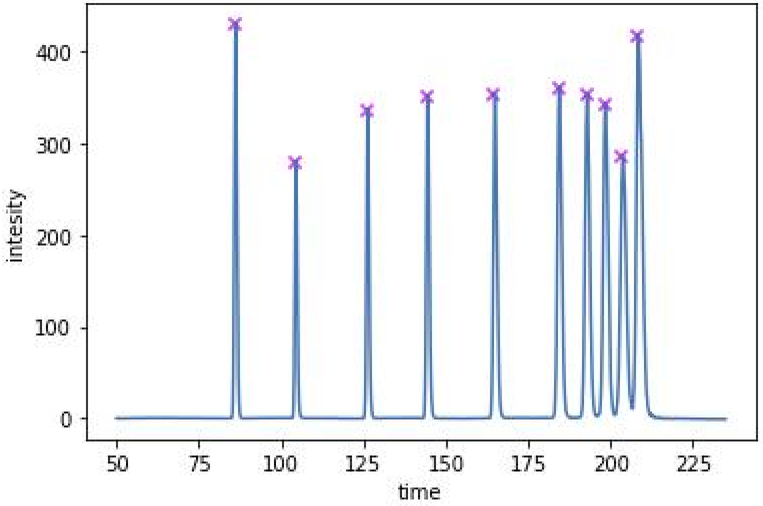
```python
%matplotlib inline
from scipy.signal import find_peaks
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("bioanalyzer_sample_data.csv")
print(df.keys())
print(df.loc[0:15])
print(df.loc[::-1])
#

```

```
Index(['Data File Name', '01_DiversiLab System_2018-05-11_14-03-40.xad'], dtype='object')
           Data File Name       01_DiversiLab System_2018-05-11_14-03-40.xad
0          Data File Path  C:\Program Files\Agilent\2100 bioanalyzer\2100...
1            Date Created                    Friday, May 11, 2018 1:03:40 PM
2      Date Last Modified                    Friday, May 11, 2018 2:04:26 PM
3         Version Created                                      C.04.09.TS792
4   Version Last Modified                                      C.04.09.TS792
5              Assay Name                             DiversiLab System V1.4
6              Assay Path  C:\Program Files\Agilent\2100 bioanalyzer\2100...
7             Assay Title                                  DiversiLab System
8           Assay Version                                                1.5
9   Number of Samples Run                                                 12
10            Sample Name                                             Ladder
11       Number of Events                                               3699
12                   Time                                              Value
13                     50                                          0.1647034
14                  50.05                                          0.1154175
15                   50.1                                          0.1156158
             Data File Name       01_DiversiLab System_2018-05-11_14-03-40.xad
3712              Alignment                                                 On
3711                  234.9                                         -0.7417297
3710                 234.85                                         -0.6775284
3709                  234.8                                         -0.6406174
3708                 234.75                                         -0.6786194
...                     ...                                                ...
4     Version Last Modified                                      C.04.09.TS792
3           Version Created                                      C.04.09.TS792
2        Date Last Modified                    Friday, May 11, 2018 2:04:26 PM
1              Date Created                    Friday, May 11, 2018 1:03:40 PM
0            Data File Path  C:\Program Files\Agilent\2100 bioanalyzer\2100...

[3713 rows x 2 columns]
```

Scientific programming - data and plotting

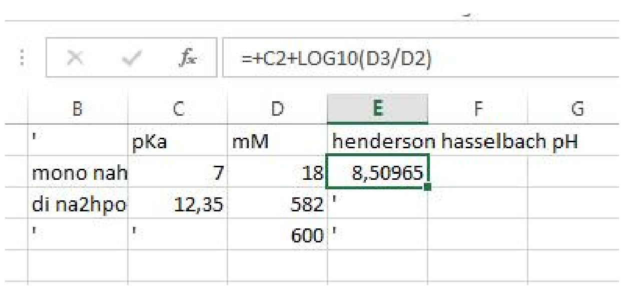


```python
position = df.loc[13:3711]['Data File Name'].astype(float)
intensity = df.loc[13:3711]['01_DiversiLab System_2018-05-11_14-03-40.xad'].astype(float)
np.savetxt('bioanalyzer_simple_data.txt',np.array([position,intensity]))
peaks, _ = find_peaks(intensity, distance=1, height=(100, 2000))
plt.scatter(position[peaks], intensity[peaks+13],color='magenta',marker='x')
plt.plot(position, intensity)
plt.xlabel('time')
plt.ylabel('intesity')
plt.show()
```



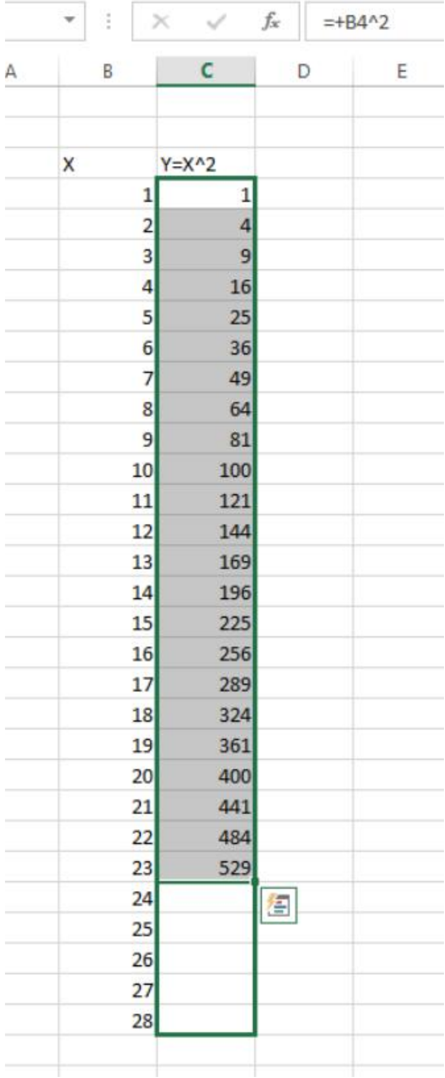
Knowing where to start - tips for visualizing programs

- arrays and lists are like columns and rows in spreadsheets
- for loops are like the "drag" function in spreadsheets



=+C2+LOG10(D3/D2)

B	C	D	E	F	G
'	pKa	mM	henderson hasselbach pH		
mono nah	7	18	8,50965		
di na2hpo	12,35	582	'		
'	'	600	'		

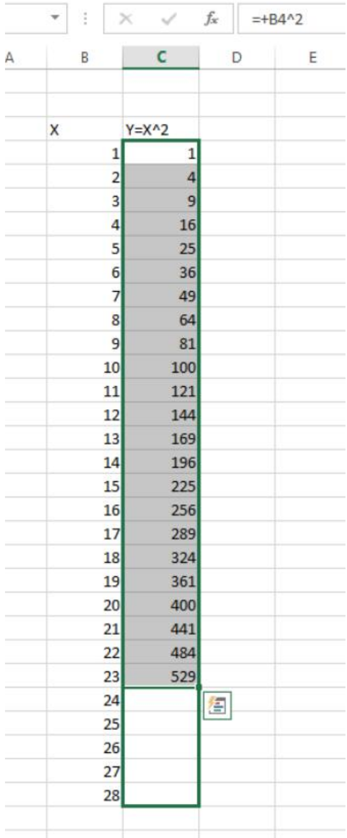


=+B4^2

A	B	C	D	E
	X	Y=X^2		
	1	1		
	2	4		
	3	9		
	4	16		
	5	25		
	6	36		
	7	49		
	8	64		
	9	81		
	10	100		
	11	121		
	12	144		
	13	169		
	14	196		
	15	225		
	16	256		
	17	289		
	18	324		
	19	361		
	20	400		
	21	441		
	22	484		
	23	529		
	24			
	25			
	26			
	27			
	28			

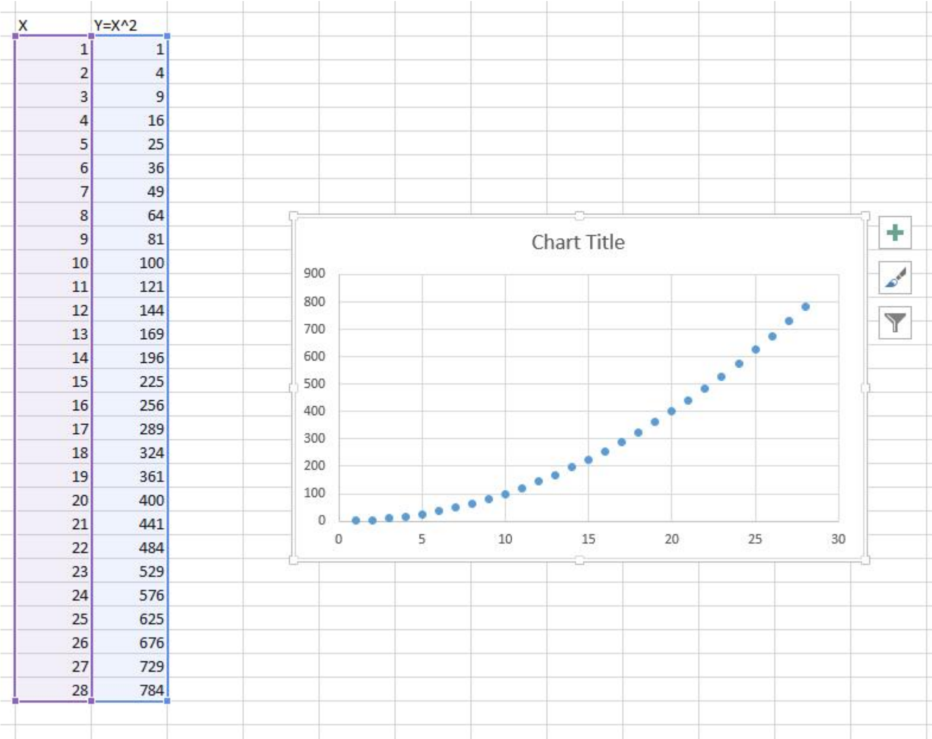
Knowing where to start - tips for visualizing programs

- arrays and lists are like columns and rows in spreadsheets
- for loops are like the "drag" function in spreadsheets



=+B4^2

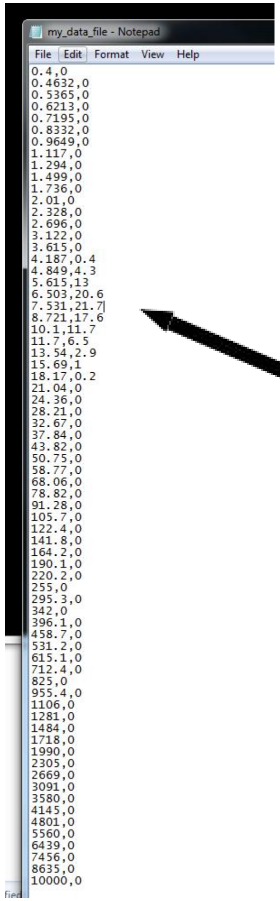
X	Y=X^2
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100
11	121
12	144
13	169
14	196
15	225
16	256
17	289
18	324
19	361
20	400
21	441
22	484
23	529
24	
25	
26	
27	
28	



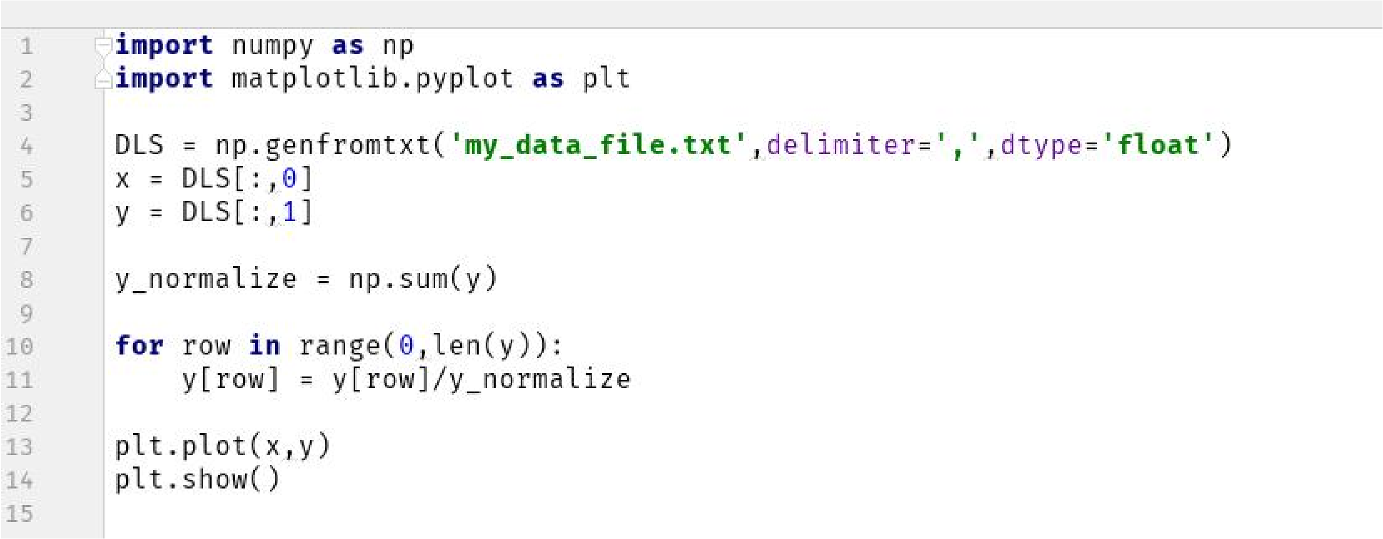
X	Y=X^2
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100
11	121
12	144
13	169
14	196
15	225
16	256
17	289
18	324
19	361
20	400
21	441
22	484
23	529
24	576
25	625
26	676
27	729
28	784

Knowing where to start - tips for visualizing programs

- arrays and lists are like columns and rows in spreadsheets
- for loops are like the “drag” function in spreadsheets



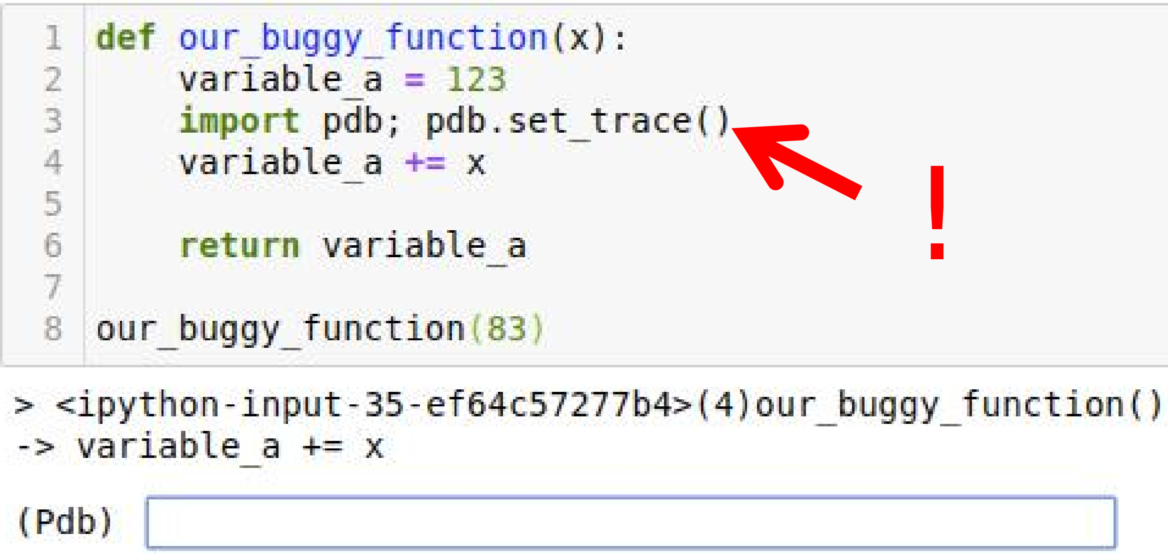
```
0.4,0
0.4632,0
0.5365,0
0.6213,0
0.7195,0
0.8332,0
0.9649,0
1.117,0
1.294,0
1.499,0
1.736,0
2.01,0
2.328,0
2.696,0
3.122,0
3.615,0
4.187,0.4
4.849,4.3
5.615,13
6.503,20.6
7.531,21.7
8.721,17.6
10.1,11.7
11.7,6.5
13.54,2.9
15.69,1
18.17,0.2
21.04,0
24.36,0
28.21,0
32.67,0
37.84,0
43.82,0
50.75,0
58.77,0
68.06,0
78.82,0
91.28,0
105.7,0
122.4,0
141.8,0
164.2,0
190.1,0
220.2,0
255,0
295.3,0
342,0
396.1,0
458.7,0
531.2,0
615.1,0
712.4,0
825,0
955.4,0
1106,0
1281,0
1484,0
1718,0
1990,0
2305,0
2669,0
3091,0
3580,0
4145,0
4801,0
5560,0
6439,0
7456,0
8635,0
10000,0
```



```python
import numpy as np
import matplotlib.pyplot as plt

DLS = np.genfromtxt('my_data_file.txt',delimiter=',',dtype='float')
x = DLS[:,0]
y = DLS[:,1]

y_normalize = np.sum(y)

for row in range(0,len(y)):
    y[row] = y[row]/y_normalize

plt.plot(x,y)
plt.show()

```

Debugging by exploring from “within” a program

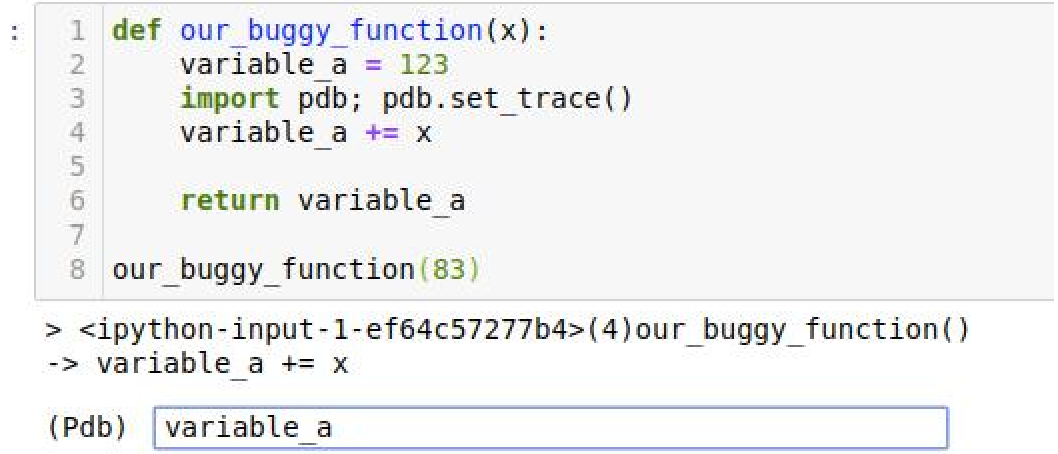
- use a set_trace() command to explore a program at a specific line



```python
def our_buggy_function(x):
    variable_a = 123
    import pdb; pdb.set_trace()
    variable_a += x

    return variable_a

our_buggy_function(83)
```

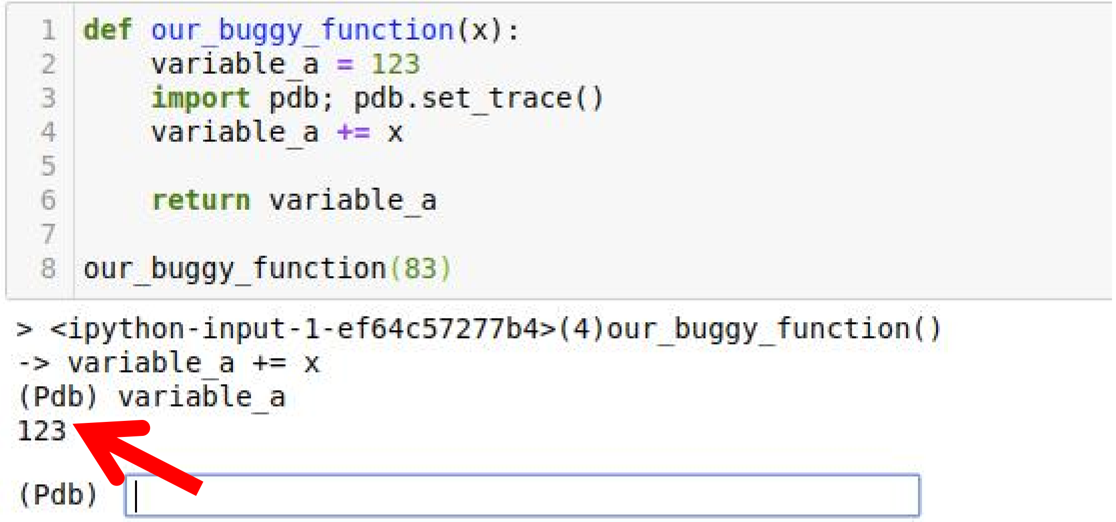
!

```
> <ipython-input-35-ef64c57277b4>(4)our_buggy_function()
-> variable_a += x

(Pdb)
```



```python
def our_buggy_function(x):
    variable_a = 123
    import pdb; pdb.set_trace()
    variable_a += x

    return variable_a

our_buggy_function(83)
```

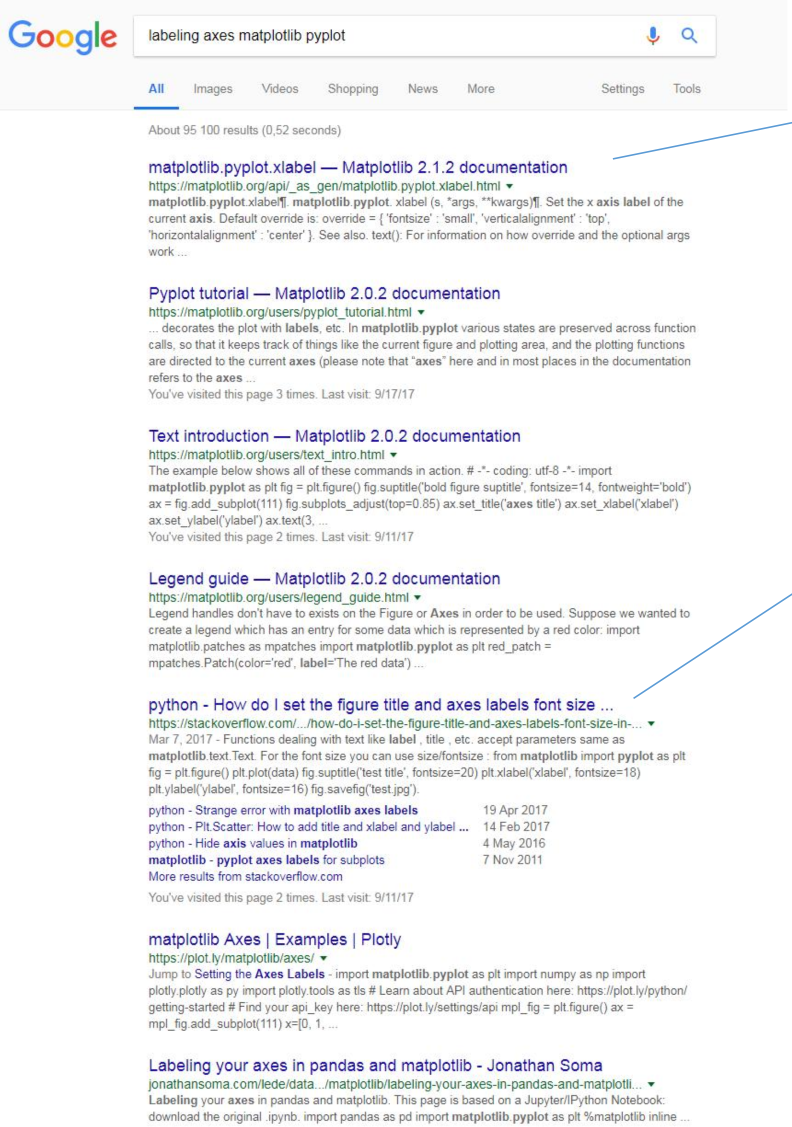
```
> <ipython-input-1-ef64c57277b4>(4)our_buggy_function()
-> variable_a += x

(Pdb) variable_a
```



```python
def our_buggy_function(x):
    variable_a = 123
    import pdb; pdb.set_trace()
    variable_a += x

    return variable_a

our_buggy_function(83)
```

```
> <ipython-input-1-ef64c57277b4>(4)our_buggy_function()
-> variable_a += x
(Pdb) variable_a
123

(Pdb) |
```

The important art of Googling



About 95 100 results (0,52 seconds)

matplotlib.pyplot.xlabel — Matplotlib 2.1.2 documentation
https://matplotlib.org/api/_as_gen/matplotlib.pyplot.xlabel.html ▾
matplotlib.**pyplot**.xlabel¶. **matplotlib**.**pyplot**. xlabel (s, *args, **kwargs)¶. Set the x **axis label** of the current **axis**. Default override is: override = { 'fontsize' : 'small', 'verticalalignment' : 'top', 'horizontalalignment' : 'center' }. See also. text(): For information on how override and the optional args work ...

Pyplot tutorial — Matplotlib 2.0.2 documentation
https://matplotlib.org/users/pyplot_tutorial.html ▾
... decorates the plot with **labels**, etc. In **matplotlib**.**pyplot** various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current **axes** (please note that "**axes**" here and in most places in the documentation refers to the **axes** ...
You've visited this page 3 times. Last visit: 9/17/17

Text introduction — Matplotlib 2.0.2 documentation
https://matplotlib.org/users/text_intro.html ▾
The example below shows all of these commands in action. # -*- coding: utf-8 -*- import **matplotlib**.**pyplot** as plt fig = plt.figure() fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold') ax = fig.add_subplot(111) fig.subplots_adjust(top=0.85) ax.set_title('**axes** title') ax.set_xlabel('xlabel') ax.set_ylabel('ylabel') ax.text(3, ...
You've visited this page 2 times. Last visit: 9/11/17

Legend guide — Matplotlib 2.0.2 documentation
https://matplotlib.org/users/legend_guide.html ▾
Legend handles don't have to exists on the Figure or **Axes** in order to be used. Suppose we wanted to create a legend which has an entry for some data which is represented by a red color: import matplotlib.patches as mpatches import **matplotlib**.**pyplot** as plt red_patch = mpatches.Patch(color='red', **label**='The red data') ...

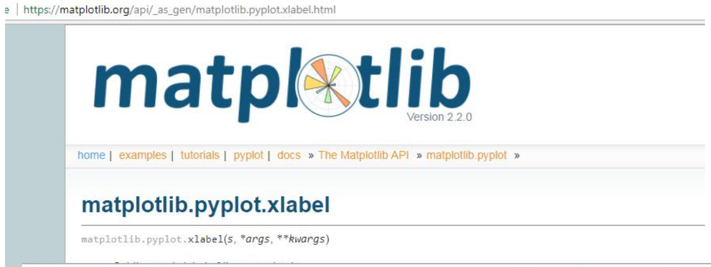
python - How do I set the figure title and axes labels font size ...
https://stackoverflow.com/.../how-do-i-set-the-figure-title-and-axes-labels-font-size-in-... ▾
Mar 7, 2017 - Functions dealing with text like **label** , title , etc. accept parameters same as **matplotlib**.text.Text. For the font size you can use size/fontsize : from **matplotlib** import **pyplot** as plt fig = plt.figure() plt.plot(data) fig.suptitle('test title', fontsize=20) plt.xlabel('xlabel', fontsize=18) plt.ylabel('ylabel', fontsize=16) fig.savefig('test.jpg').

python - Strange error with **matplotlib axes labels**	19 Apr 2017
python - Plt.Scatter: How to add title and xlabel and ylabel ...	14 Feb 2017
python - Hide **axis** values in **matplotlib**	4 May 2016
matplotlib - **pyplot axes labels** for subplots	7 Nov 2011

More results from stackoverflow.com
You've visited this page 2 times. Last visit: 9/11/17

matplotlib Axes | Examples | Plotly
https://plot.ly/matplotlib/axes/ ▾
Jump to Setting the **Axes Labels** - import **matplotlib**.**pyplot** as plt import numpy as np import plotly.plotly as py import plotly.tools as tls # Learn about API authentication here: https://plot.ly/python/getting-started # Find your api_key here: https://plot.ly/settings/api mpl_fig = plt.figure() ax = mpl_fig.add_subplot(111) x=[0, 1, ...

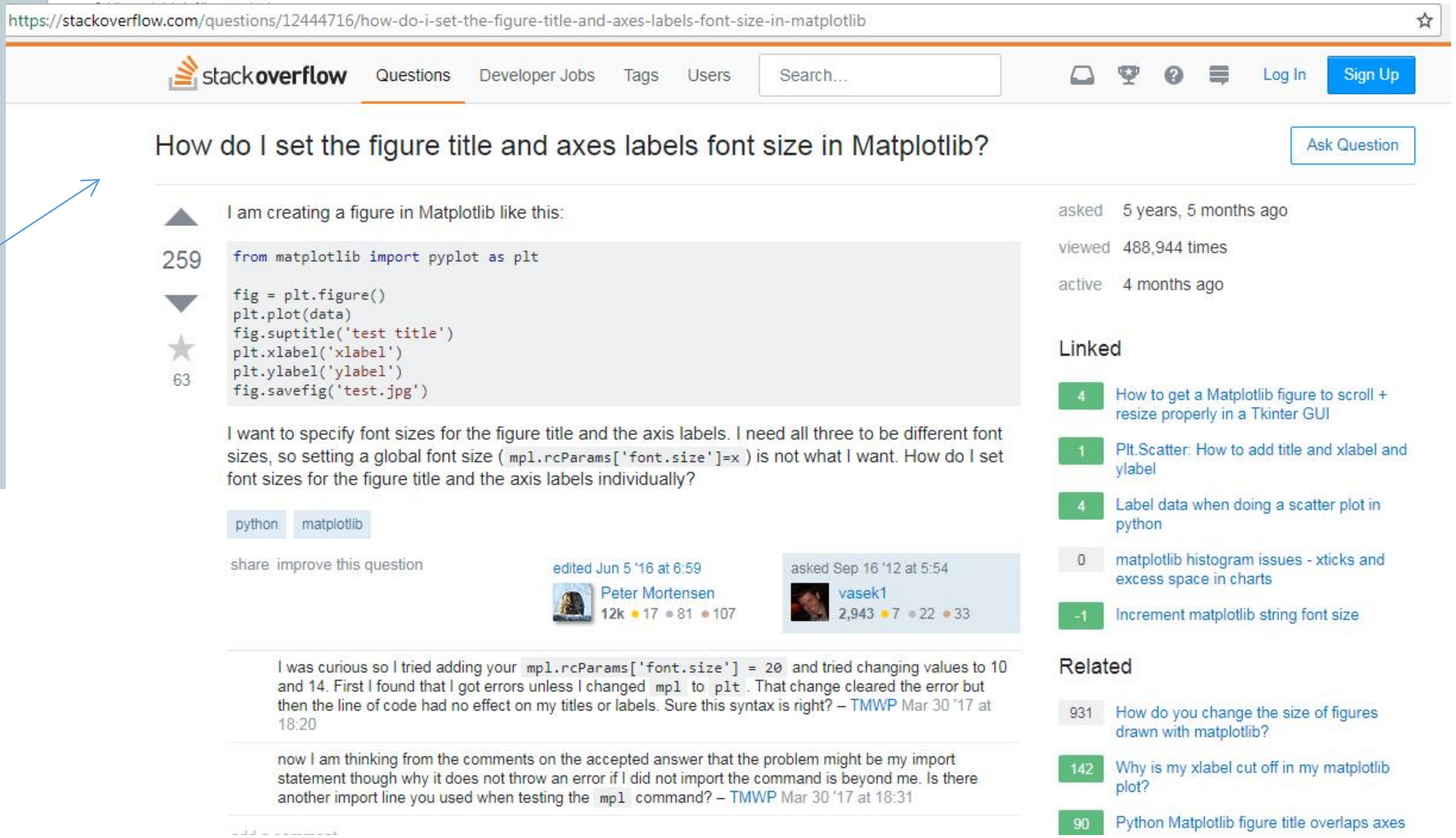
Labeling your axes in pandas and matplotlib - Jonathan Soma
jonathansoma.com/lede/data.../matplotlib/labeling-your-axes-in-pandas-and-matplotli... ▾
Labeling your **axes** in pandas and matplotlib. This page is based on a Jupyter/IPython Notebook: download the original .ipynb. import pandas as pd import **matplotlib**.**pyplot** as plt %matplotlib inline ...


matplotlib
Version 2.2.0
home | examples | tutorials | pyplot | docs » The Matplotlib API » matplotlib.pyplot »

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.xlabel.html

matplotlib.pyplot.xlabel

matplotlib.pyplot.xlabel(*s*, **args*, ***kwargs*)



https://stackoverflow.com/questions/12444716/how-do-i-set-the-figure-title-and-axes-labels-font-size-in-matplotlib

How do I set the figure title and axes labels font size in Matplotlib?

I am creating a figure in Matplotlib like this:

```
from matplotlib import pyplot as plt

fig = plt.figure()
plt.plot(data)
fig.suptitle('test title')
plt.xlabel('xlabel')
plt.ylabel('ylabel')
fig.savefig('test.jpg')
```

I want to specify font sizes for the figure title and the axis labels. I need all three to be different font sizes, so setting a global font size (`mpl.rcParams['font.size']=x`) is not what I want. How do I set font sizes for the figure title and the axis labels individually?

python matplotlib

share improve this question

edited Jun 5 '16 at 6:59
Peter Mortensen
12k ● 17 ● 81 ● 107

asked Sep 16 '12 at 5:54
vasek1
2,943 ● 7 ● 22 ● 33

I was curious so I tried adding your `mpl.rcParams['font.size'] = 20` and tried changing values to 10 and 14. First I found that I got errors unless I changed `mpl` to `plt`. That change cleared the error but then the line of code had no effect on my titles or labels. Sure this syntax is right? – TMWP Mar 30 '17 at 18:20

now I am thinking from the comments on the accepted answer that the problem might be my import statement though why it does not throw an error if I did not import the command is beyond me. Is there another import line you used when testing the `mpl` command? – TMWP Mar 30 '17 at 18:31

asked 5 years, 5 months ago
viewed 488,944 times
active 4 months ago

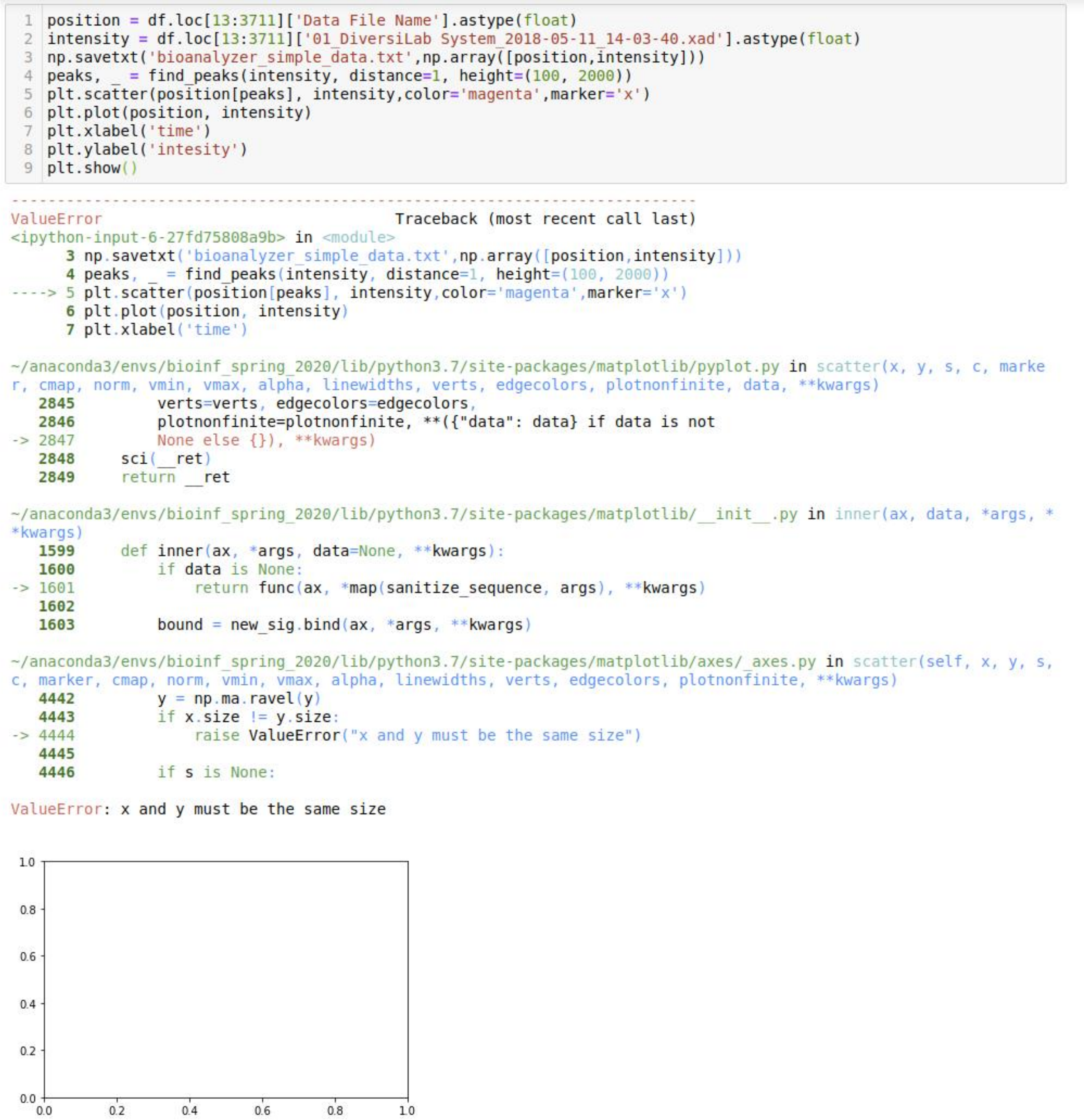
Linked

4	How to get a Matplotlib figure to scroll + resize properly in a Tkinter GUI
1	Plt.Scatter: How to add title and xlabel and ylabel
4	Label data when doing a scatter plot in python
0	matplotlib histogram issues - xticks and excess space in charts
-1	Increment matplotlib string font size

Related

931	How do you change the size of figures drawn with matplotlib?
142	Why is my xlabel cut off in my matplotlib plot?
90	Python Matplotlib figure title overlaps axes

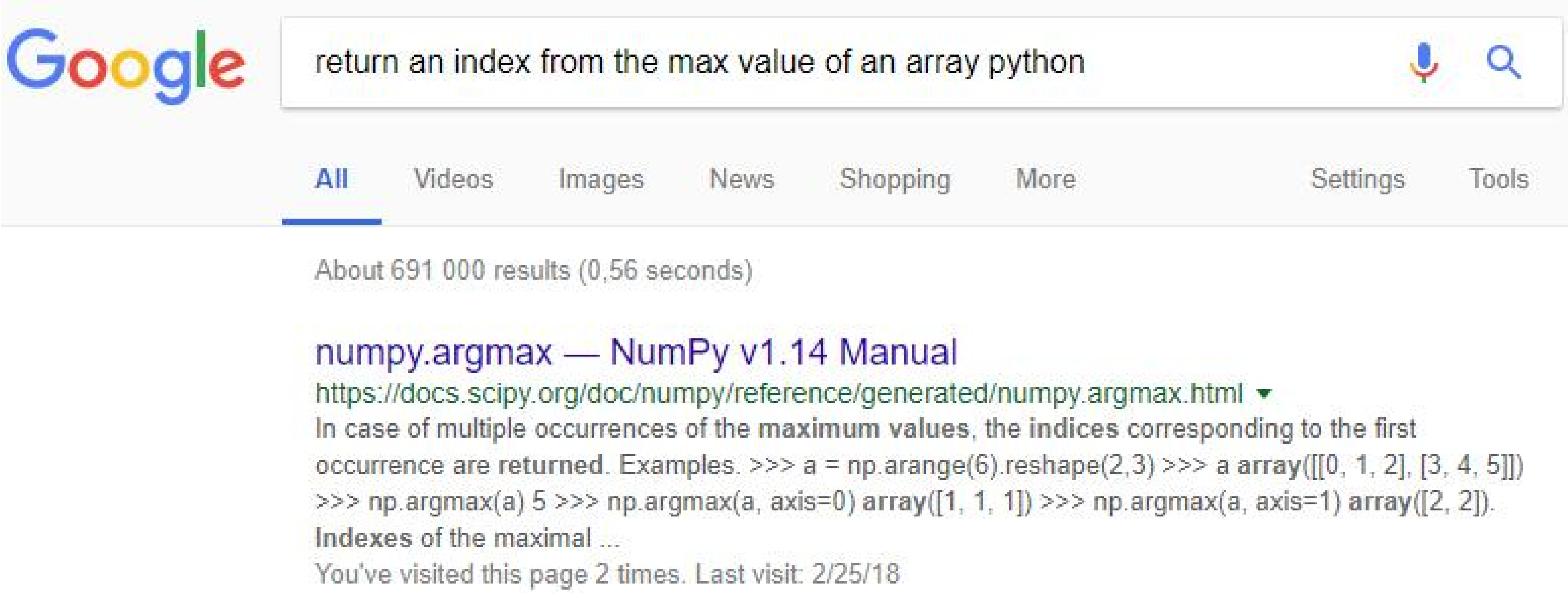
Use text returned from errors to identify location and type of error



```
position = df.loc[13:3711]['Data File Name'].astype(float)
intensity = df.loc[13:3711]['01_DiversiLab System_2018-05-11_14-03-40.xad'].astype(float)
np.savetxt('bioanalyzer_simple_data.txt',np.array([position,intensity]))
peaks, _ = find_peaks(intensity, distance=1, height=(100, 2000))
plt.scatter(position[peaks], intensity,color='magenta',marker='x')
plt.plot(position, intensity)
plt.xlabel('time')
plt.ylabel('intesity')
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-6-27fd75808a9b> in <module>
      3 np.savetxt('bioanalyzer_simple_data.txt',np.array([position,intensity]))
      4 peaks, _ = find_peaks(intensity, distance=1, height=(100, 2000))
----> 5 plt.scatter(position[peaks], intensity,color='magenta',marker='x')
      6 plt.plot(position, intensity)
      7 plt.xlabel('time')

~/anaconda3/envs/bioinf_spring_2020/lib/python3.7/site-packages/matplotlib/pyplot.py in scatter(x, y, s, c, marke
r, cmap, norm, vmin, vmax, alpha, linewidths, verts, edgecolors, plotnonfinite, data, **kwargs)
   2845         verts=verts, edgecolors=edgecolors,
   2846         plotnonfinite=plotnonfinite, **({"data": data} if data is not
-> 2847         None else {}), **kwargs)
   2848     sci(__ret)
   2849     return __ret

~/anaconda3/envs/bioinf_spring_2020/lib/python3.7/site-packages/matplotlib/__init__.py in inner(ax, data, *args, *
*kwargs)
   1599     def inner(ax, *args, data=None, **kwargs):
   1600         if data is None:
-> 1601             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1602
   1603         bound = new_sig.bind(ax, *args, **kwargs)

~/anaconda3/envs/bioinf_spring_2020/lib/python3.7/site-packages/matplotlib/axes/_axes.py in scatter(self, x, y, s,
c, marker, cmap, norm, vmin, vmax, alpha, linewidths, verts, edgecolors, plotnonfinite, **kwargs)
   4442         y = np.ma.ravel(y)
   4443         if x.size != y.size:
-> 4444             raise ValueError("x and y must be the same size")
   4445
   4446         if s is None:

ValueError: x and y must be the same size
```

Googling gets easier as you learn vocabulary



return an index from the max value of an array python

About 691 000 results (0,56 seconds)

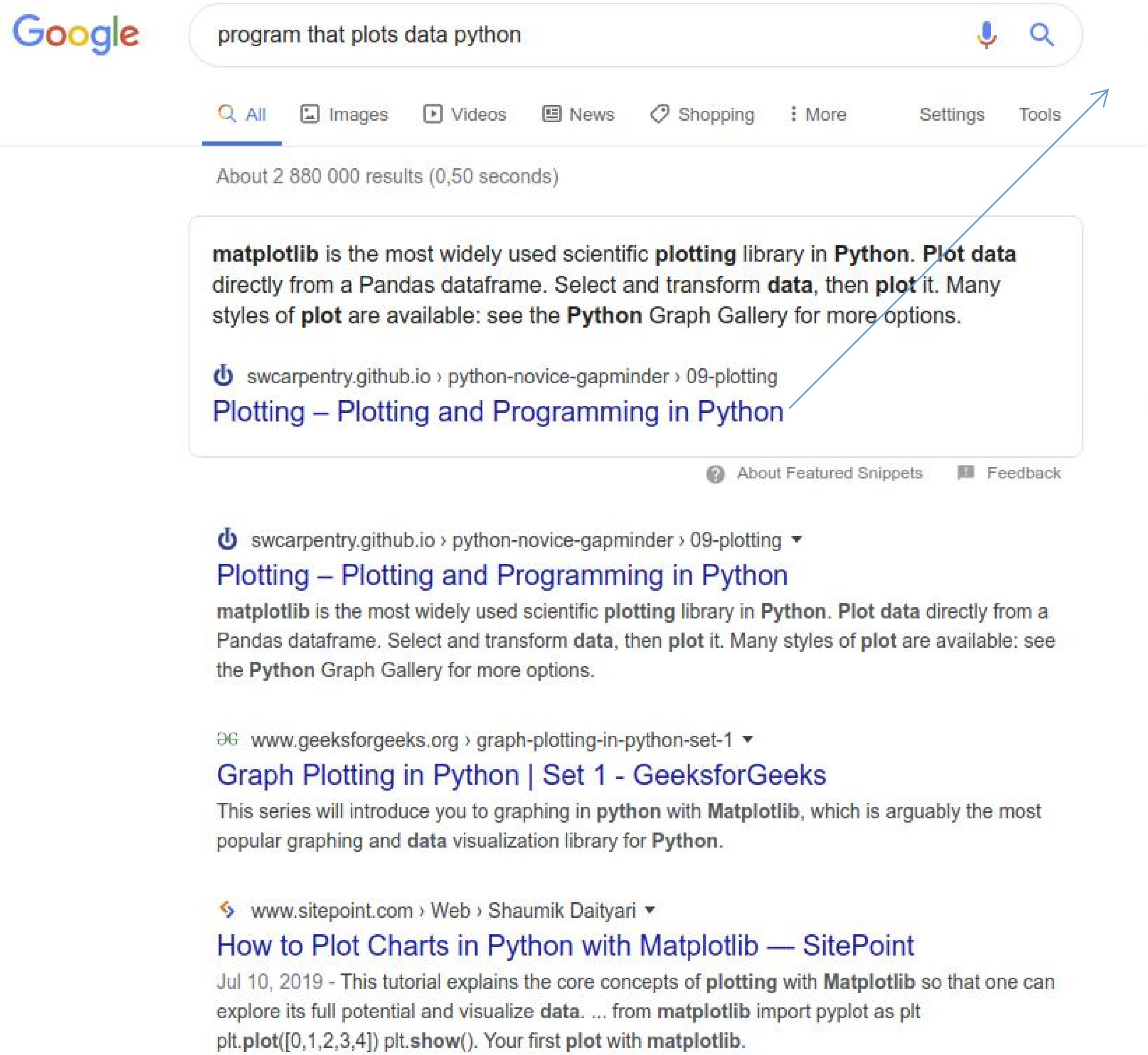
numpy.argmax — NumPy v1.14 Manual

https://docs.scipy.org/doc/numpy/reference/generated/numpy.argmax.html ▾

In case of multiple occurrences of the **maximum values**, the **indices** corresponding to the first occurrence are **returned**. Examples. >>> a = np.arange(6).reshape(2,3) >>> a **array**([[0, 1, 2], [3, 4, 5]]) >>> np.argmax(a) 5 >>> np.argmax(a, axis=0) **array**([1, 1, 1]) >>> np.argmax(a, axis=1) **array**([2, 2]). **Indexes** of the maximal ...

You've visited this page 2 times. Last visit: 2/25/18

Borrow sample code and modify it



Google

program that plots data python

About 2 880 000 results (0,50 seconds)

matplotlib is the most widely used scientific **plotting** library in **Python**. **Plot data** directly from a Pandas dataframe. Select and transform **data**, then **plot** it. Many styles of **plot** are available: see the **Python** Graph Gallery for more options.

swcarpentry.github.io › python-novice-gapminder › 09-plotting

Plotting – Plotting and Programming in Python

swcarpentry.github.io › python-novice-gapminder › 09-plotting ▾

Plotting – Plotting and Programming in Python

matplotlib is the most widely used scientific **plotting** library in **Python**. **Plot data** directly from a Pandas dataframe. Select and transform **data**, then **plot** it. Many styles of **plot** are available: see the **Python** Graph Gallery for more options.

www.geeksforgeeks.org › graph-plotting-in-python-set-1 ▾

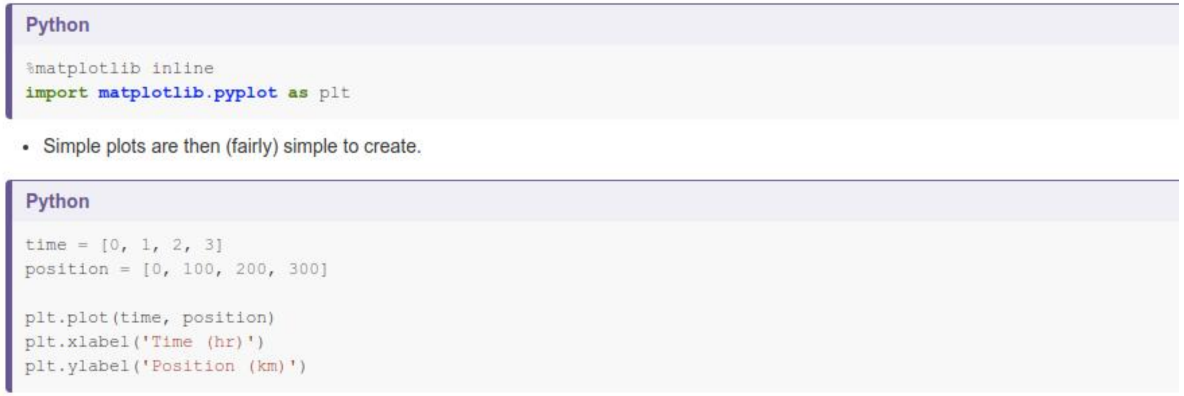
Graph Plotting in Python | Set 1 - GeeksforGeeks

This series will introduce you to graphing in **python** with **Matplotlib**, which is arguably the most popular graphing and **data** visualization library for **Python**.

www.sitepoint.com › Web › Shaumik Daityari ▾

How to Plot Charts in Python with Matplotlib — SitePoint

Jul 10, 2019 - This tutorial explains the core concepts of **plotting** with **Matplotlib** so that one can explore its full potential and visualize **data**. ... from **matplotlib** import pyplot as plt plt.**plot**([0,1,2,3,4]) plt.**show**(). Your first **plot** with **matplotlib**.



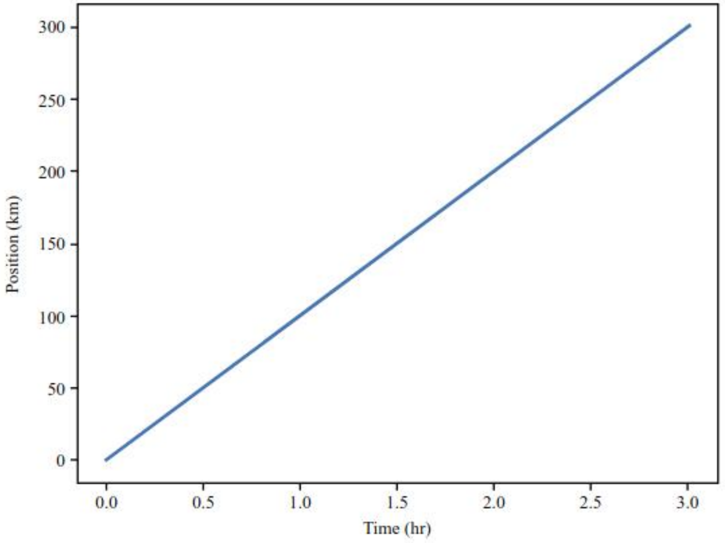
Python

```
%matplotlib inline
import matplotlib.pyplot as plt
```

- Simple plots are then (fairly) simple to create.

Python

```
time = [0, 1, 2, 3]
position = [0, 100, 200, 300]

plt.plot(time, position)
plt.xlabel('Time (hr)')
plt.ylabel('Position (km)')
```



Tips for getting started on your own

- download and install a distribution of Python
  - anaconda is a good one (free, comes with many scientific programming libraries)
- download and install a program editor (for writing and saving code)
  - Spyder - a good, free editing application that we will use in our exercises

Tip: come up with your own project - something you care about

- take a spreadsheet and convert it into Python
- pick something from a math or science textbook and implement it in Python
  - networks
  - machine learning
  - bioinformatics :)
- pick a boring/repetitive task that you have to do often and automate it
- make something visual with matplotlib
  - data visualization
  - animation
  - plot a nice mathematical function

Independent learning resources

- youtube
  - thousands of tutorials on everything from basics to specific libraries
- MOOCs - massive online open course
  - Coursera
  - EDX
- forums - ask questions and get answers from other programmers
  - stack overflow
  - reddit