# Introduction to Python

Presented by GradQuant

Steph DeMora

## Acknowledgement:

Slides adopted by **Preston Carman, Steven Jacobs, Rohith Mohan, Heran Bhakta…**

Based on: **Introduction to Python and Programming** by **Michael Ernst** (UW CSE 190p, Summer 2012)

## Who should attend?

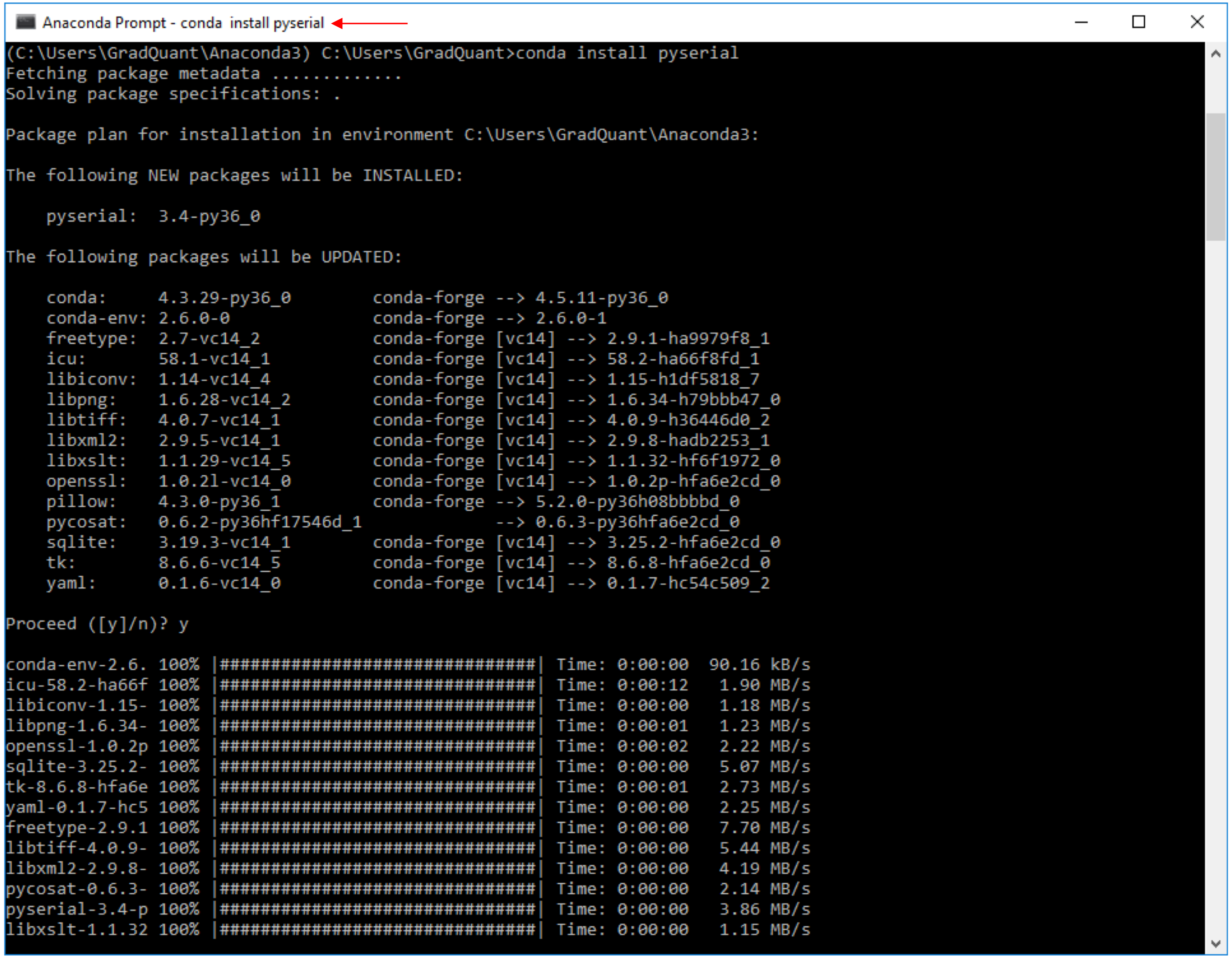
NO PROGRAMMING EXPERIENCE

NEVER USED PYTHON

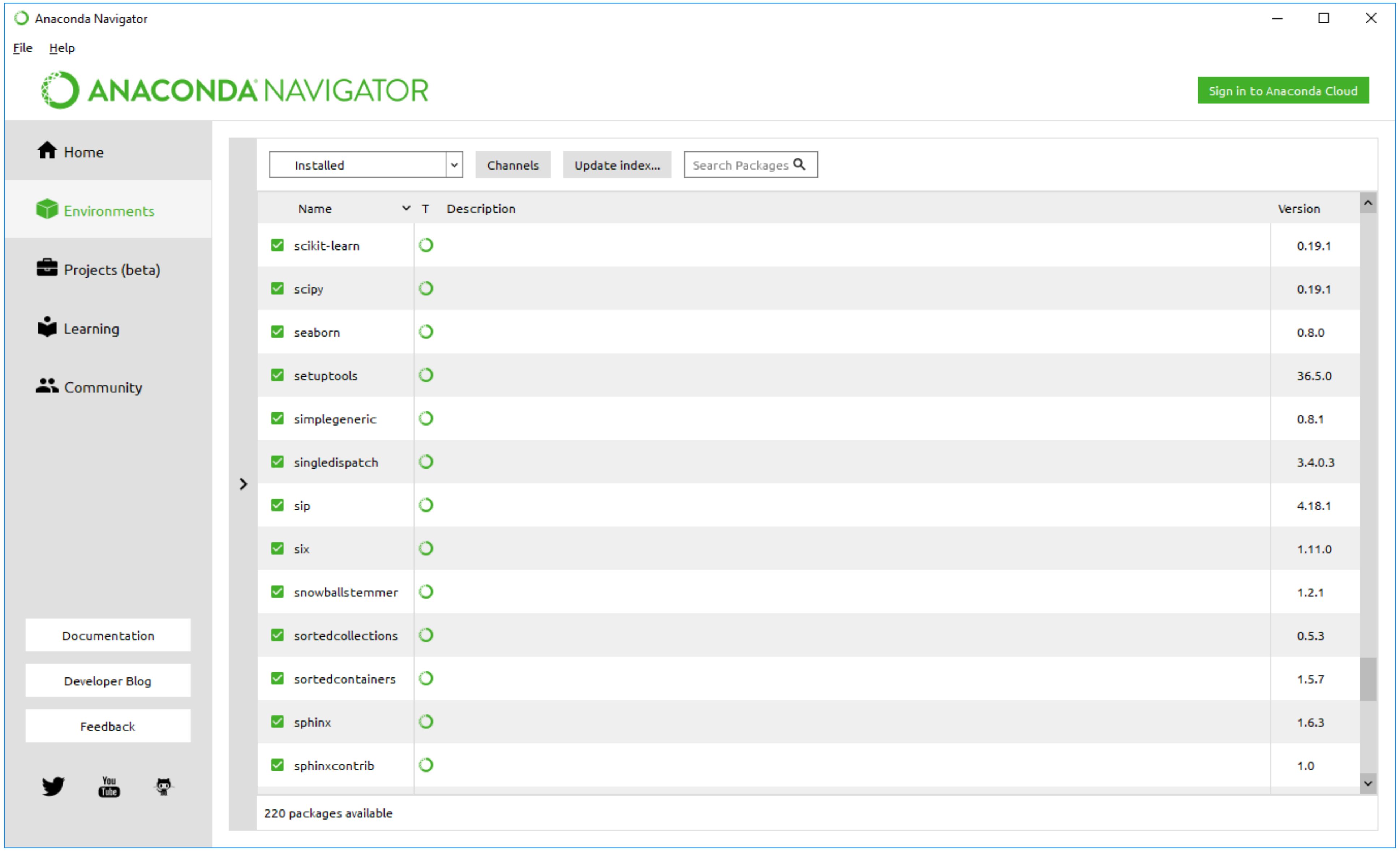
## Goals for this workshop

- Introduce Python programming concepts
  - Distributions and packages (anaconda)
- Review Python syntax
- Review available development tools
- Create a Python script!

## Anaconda

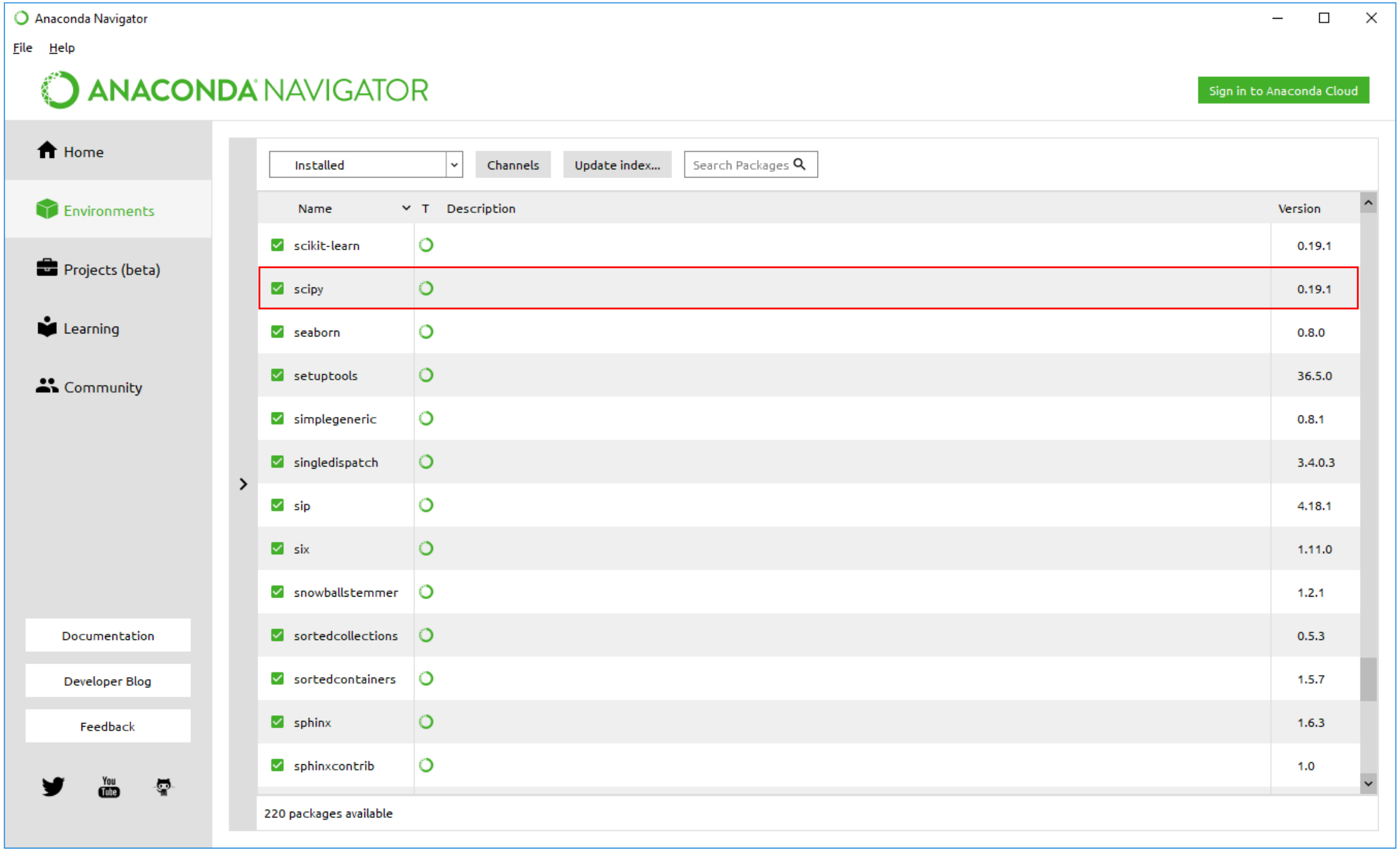


```
(C:\Users\GradQuant\Anaconda3) C:\Users\GradQuant>conda install pyserial
Fetching package metadata .............
Solving package specifications: .

Package plan for installation in environment C:\Users\GradQuant\Anaconda3:

The following NEW packages will be INSTALLED:

    pyserial:  3.4-py36_0

The following packages will be UPDATED:

    conda:     4.3.29-py36_0        conda-forge --> 4.5.11-py36_0
    conda-env: 2.6.0-0              conda-forge --> 2.6.0-1
    freetype:  2.7-vc14_2           conda-forge [vc14] --> 2.9.1-ha9979f8_1
    icu:       58.1-vc14_1          conda-forge [vc14] --> 58.2-ha66f8fd_1
    libiconv:  1.14-vc14_4          conda-forge [vc14] --> 1.15-h1df5818_7
    libpng:    1.6.28-vc14_2        conda-forge [vc14] --> 1.6.34-h79bbb47_0
    libtiff:   4.0.7-vc14_1         conda-forge [vc14] --> 4.0.9-h36446d0_2
    libxml2:   2.9.5-vc14_1         conda-forge [vc14] --> 2.9.8-hadb2253_1
    libxslt:   1.1.29-vc14_5        conda-forge [vc14] --> 1.1.32-hf6f1972_0
    openssl:   1.0.2l-vc14_0        conda-forge [vc14] --> 1.0.2p-hfa6e2cd_0
    pillow:    4.3.0-py36_1         conda-forge --> 5.2.0-py36h08bbbbd_0
    pycosat:   0.6.2-py36hf17546d_1             --> 0.6.3-py36hfa6e2cd_0
    sqlite:    3.19.3-vc14_1        conda-forge [vc14] --> 3.25.2-hfa6e2cd_0
    tk:        8.6.6-vc14_5         conda-forge [vc14] --> 8.6.8-hfa6e2cd_0
    yaml:      0.1.6-vc14_0         conda-forge [vc14] --> 0.1.7-hc54c509_2

Proceed ([y]/n)? y

conda-env-2.6. 100% |###############################| Time: 0:00:00  90.16 kB/s
icu-58.2-ha66f 100% |###############################| Time: 0:00:12   1.90 MB/s
libiconv-1.15- 100% |###############################| Time: 0:00:00   1.18 MB/s
libpng-1.6.34- 100% |###############################| Time: 0:00:01   1.23 MB/s
openssl-1.0.2p 100% |###############################| Time: 0:00:02   2.22 MB/s
sqlite-3.25.2- 100% |###############################| Time: 0:00:00   5.07 MB/s
tk-8.6.8-hfa6e 100% |###############################| Time: 0:00:01   2.73 MB/s
yaml-0.1.7-hc5 100% |###############################| Time: 0:00:00   2.25 MB/s
freetype-2.9.1 100% |###############################| Time: 0:00:00   7.70 MB/s
libtiff-4.0.9- 100% |###############################| Time: 0:00:00   5.44 MB/s
libxml2-2.9.8- 100% |###############################| Time: 0:00:00   4.19 MB/s
pycosat-0.6.3- 100% |###############################| Time: 0:00:00   2.14 MB/s
pyserial-3.4-p 100% |###############################| Time: 0:00:00   3.86 MB/s
libxslt-1.1.32 100% |###############################| Time: 0:00:00   1.15 MB/s
```



| Name | T | Description | Version |
|---|---|---|---|
| ✓ scikit-learn | ○ | | 0.19.1 |
| ✓ scipy | ○ | | 0.19.1 |
| ✓ seaborn | ○ | | 0.8.0 |
| ✓ setuptools | ○ | | 36.5.0 |
| ✓ simplegeneric | ○ | | 0.8.1 |
| ✓ singledispatch | ○ | | 3.4.0.3 |
| ✓ sip | ○ | | 4.18.1 |
| ✓ six | ○ | | 1.11.0 |
| ✓ snowballstemmer | ○ | | 1.2.1 |
| ✓ sortedcollections | ○ | | 0.5.3 |
| ✓ sortedcontainers | ○ | | 1.5.7 |
| ✓ sphinx | ○ | | 1.6.3 |
| ✓ sphinxcontrib | ○ | | 1.0 |

220 packages available

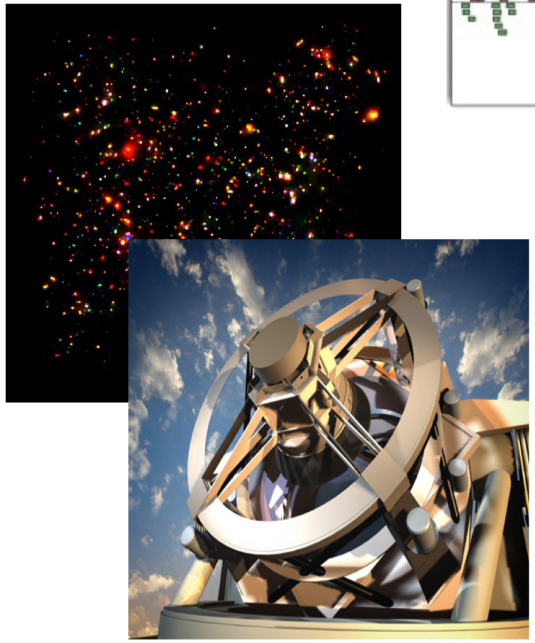


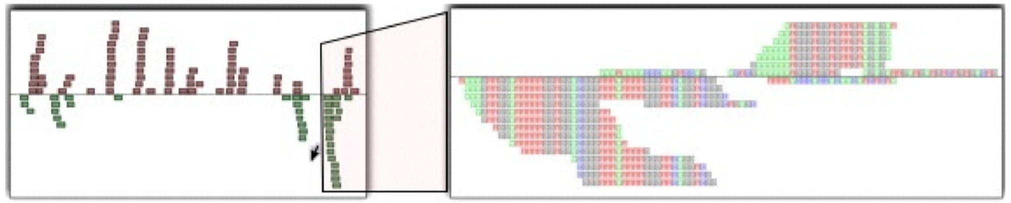
| Name | T | Description | Version |
|---|---|---|---|
| ✓ scikit-learn | ○ | | 0.19.1 |
| ✓ scipy | ○ | | 0.19.1 |
| ✓ seaborn | ○ | | 0.8.0 |
| ✓ setuptools | ○ | | 36.5.0 |
| ✓ simplegeneric | ○ | | 0.8.1 |
| ✓ singledispatch | ○ | | 3.4.0.3 |
| ✓ sip | ○ | | 4.18.1 |
| ✓ six | ○ | | 1.11.0 |
| ✓ snowballstemmer | ○ | | 1.2.1 |
| ✓ sortedcollections | ○ | | 0.5.3 |
| ✓ sortedcontainers | ○ | | 1.5.7 |
| ✓ sphinx | ○ | | 1.6.3 |
| ✓ sphinxcontrib | ○ | | 1.0 |

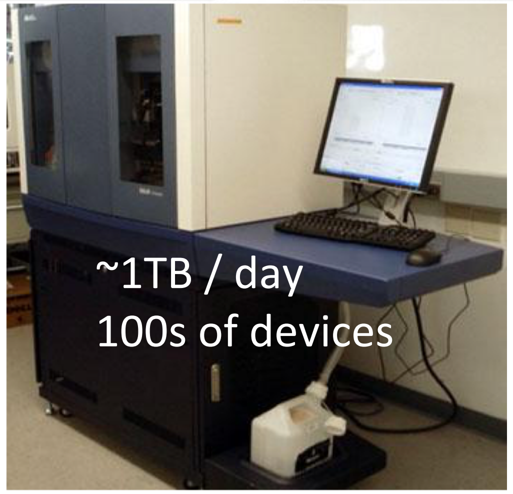
220 packages available

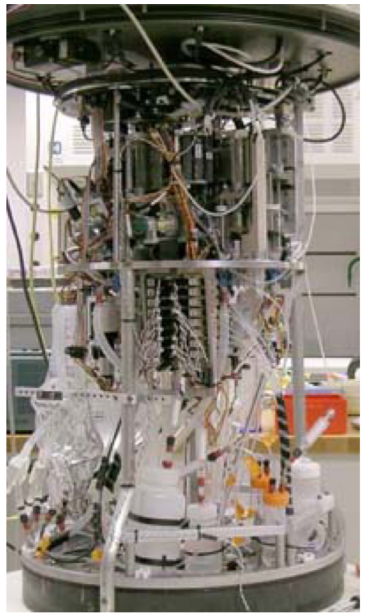
# All of science is reducing to computational data manipulation

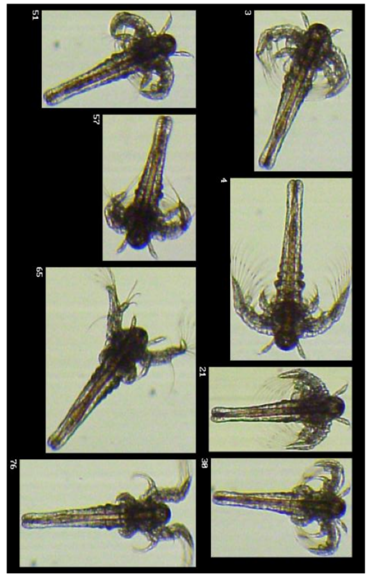
- Astronomy: High-resolution, high-frequency sky surveys (SDSS, LSST, PanSTARRS)
- Biology: lab automation, high-throughput sequencing,
- Oceanography: high-resolution models, cheap sensors, satellites



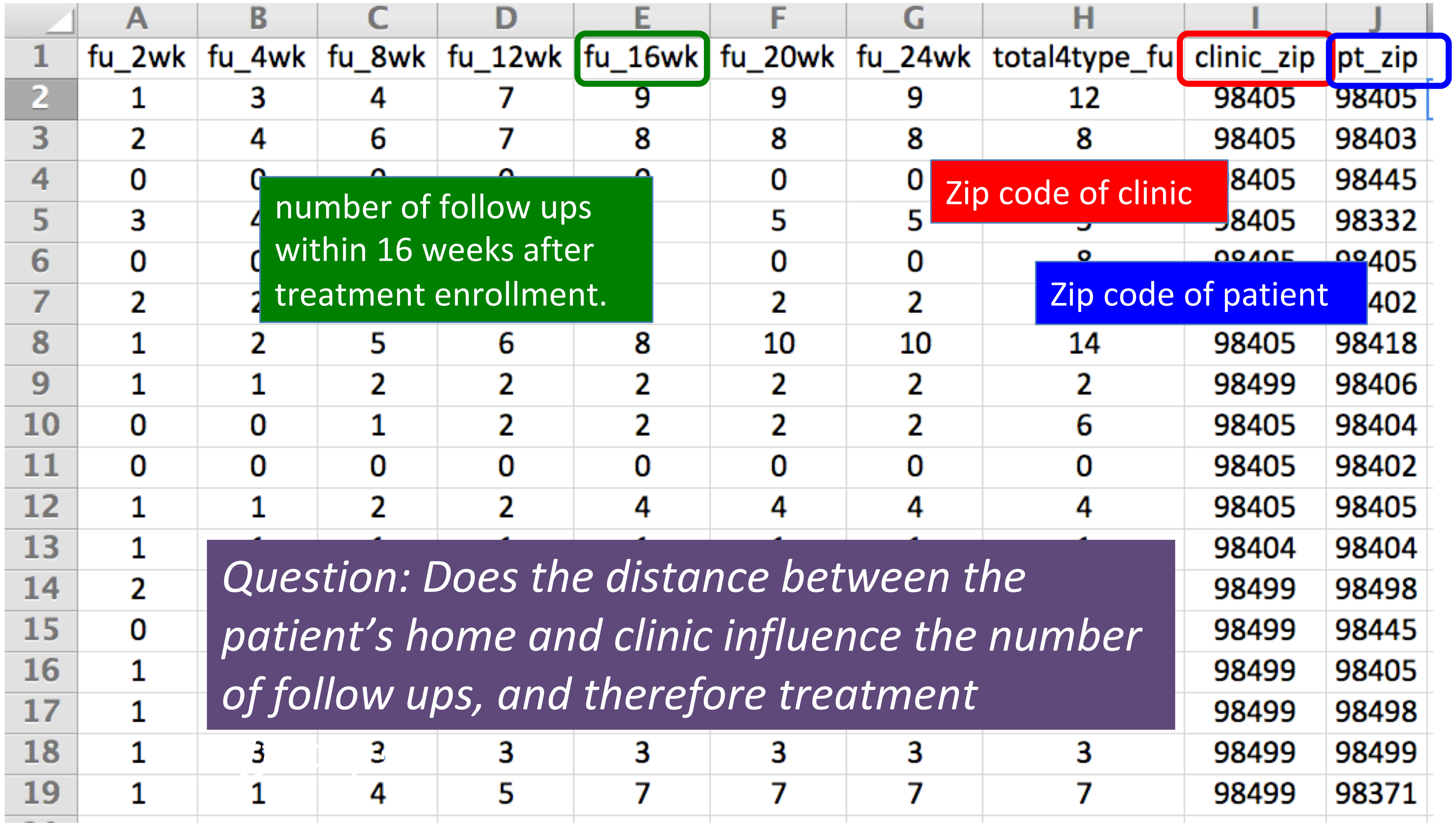








# Example: Assessing treatment efficacy



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | fu_2wk | fu_4wk | fu_8wk | fu_12wk | fu_16wk | fu_20wk | fu_24wk | total4type_fu | clinic_zip | pt_zip |
| 2 | 1 | 3 | 4 | 7 | 9 | 9 | 9 | 12 | 98405 | 98405 |
| 3 | 2 | 4 | 6 | 7 | 8 | 8 | 8 | 8 | 98405 | 98403 |
| 4 | 0 | 0 | [illegible] | [illegible] | [illegible] | 0 | 0 | [illegible] | [illegible] | 98445 |
| 5 | 3 | 4 | [illegible] | [illegible] | [illegible] | 5 | 5 | [illegible] | [illegible] | 98332 |
| 6 | 0 | 0 | [illegible] | [illegible] | [illegible] | 0 | 0 | 8 | 98405 | 98405 |
| 7 | 2 | 2 | [illegible] | [illegible] | [illegible] | 2 | 2 | [illegible] | [illegible] | [illegible] |
| 8 | 1 | 2 | 5 | 6 | 8 | 10 | 10 | 14 | 98405 | 98418 |
| 9 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 98499 | 98406 |
| 10 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 6 | 98405 | 98404 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 98405 | 98402 |
| 12 | 1 | 1 | 2 | 2 | 4 | 4 | 4 | 4 | 98405 | 98405 |
| 13 | 1 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 98404 | 98404 |
| 14 | 2 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 98499 | 98498 |
| 15 | 0 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 98499 | 98445 |
| 16 | 1 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 98499 | 98405 |
| 17 | 1 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 98499 | 98498 |
| 18 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 98499 | 98499 |
| 19 | 1 | 1 | 4 | 5 | 7 | 7 | 7 | 7 | 98499 | 98371 |

number of follow ups within 16 weeks after treatment enrollment.

Zip code of clinic

Zip code of patient

*Question: Does the distance between the patient's home and clinic influence the number of follow ups, and therefore treatment*

## Assessing treatment efficacy

# This program reads an Excel spreadsheet whose penultimate

# and antepenultimate columns are zip codes.

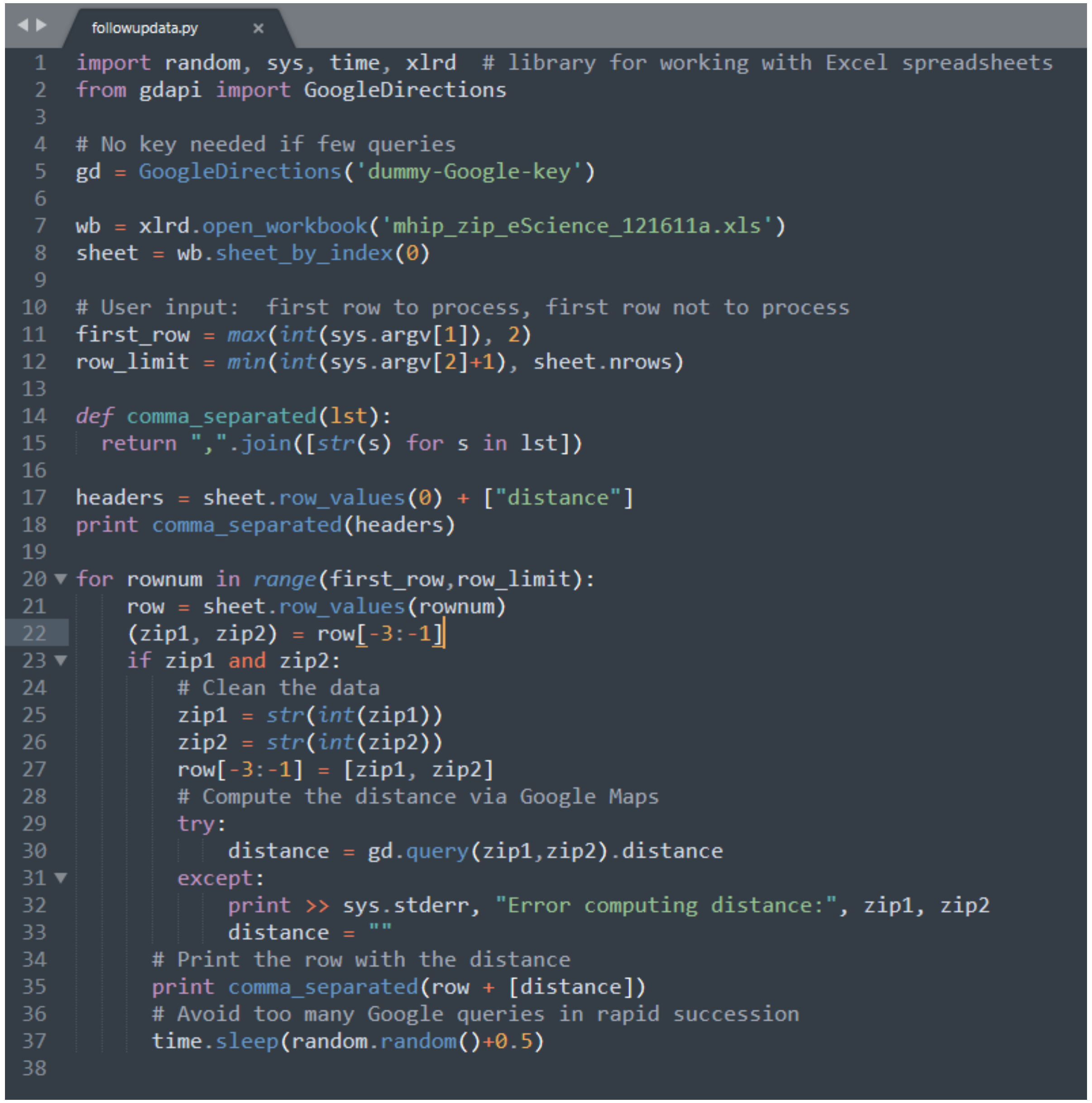
# It adds a new last column for the distance between those zip

# codes, and outputs in CSV (comma-separated values) format.

# Call the program with two numeric values: the first and last

# row to include.

# The output contains the column headers and those rows.

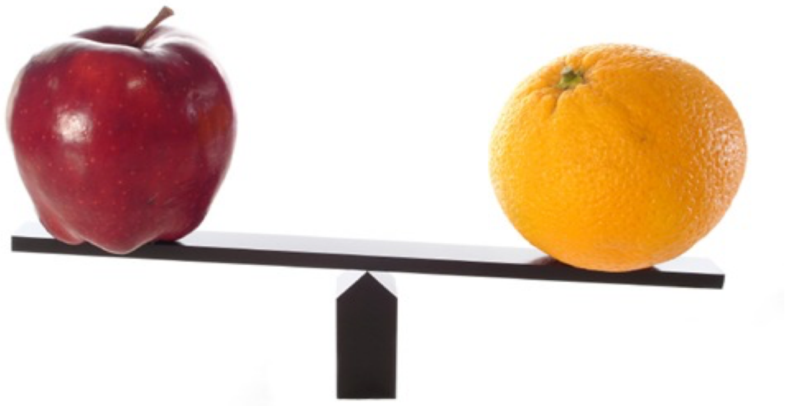

followupdata.py

```python
import random, sys, time, xlrd  # library for working with Excel spreadsheets
from gdapi import GoogleDirections

# No key needed if few queries
gd = GoogleDirections('dummy-Google-key')

wb = xlrd.open_workbook('mhip_zip_eScience_121611a.xls')
sheet = wb.sheet_by_index(0)

# User input:  first row to process, first row not to process
first_row = max(int(sys.argv[1]), 2)
row_limit = min(int(sys.argv[2]+1), sheet.nrows)

def comma_separated(lst):
  return ",".join([str(s) for s in lst])

headers = sheet.row_values(0) + ["distance"]
print comma_separated(headers)

for rownum in range(first_row,row_limit):
    row = sheet.row_values(rownum)
    (zip1, zip2) = row[-3:-1]
    if zip1 and zip2:
        # Clean the data
        zip1 = str(int(zip1))
        zip2 = str(int(zip2))
        row[-3:-1] = [zip1, zip2]
        # Compute the distance via Google Maps
        try:
            distance = gd.query(zip1,zip2).distance
        except:
            print >> sys.stderr, "Error computing distance:", zip1, zip2
            distance = ""
      # Print the row with the distance
      print comma_separated(row + [distance])
      # Avoid too many Google queries in rapid succession
      time.sleep(random.random()+0.5)

```

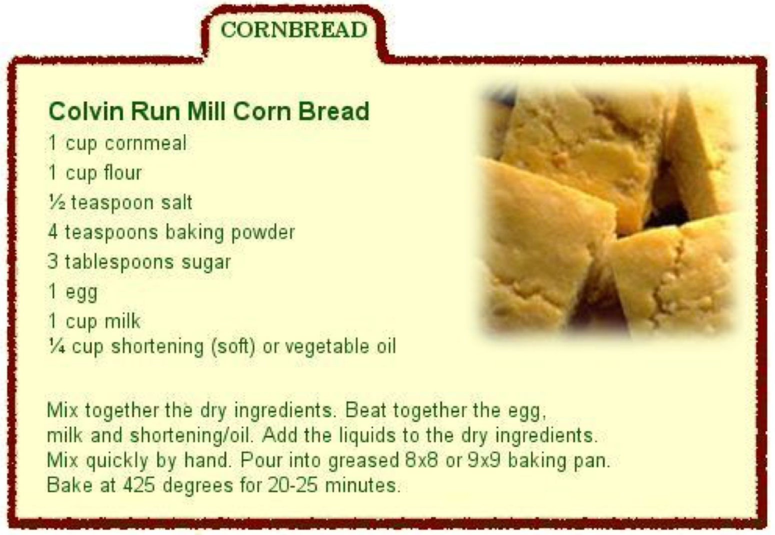
1. A variable contains a value

2. Python performs operations

3. Different types act differently



4. A program is a recipe



CORNBREAD

Colvin Run Mill Corn Bread

1 cup cornmeal
1 cup flour
½ teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
¼ cup shortening (soft) or vegetable oil

Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.

## Don’t panic!

This workshop is for people who have never  programmed

(If you have programmed, you don’t need to be here.)

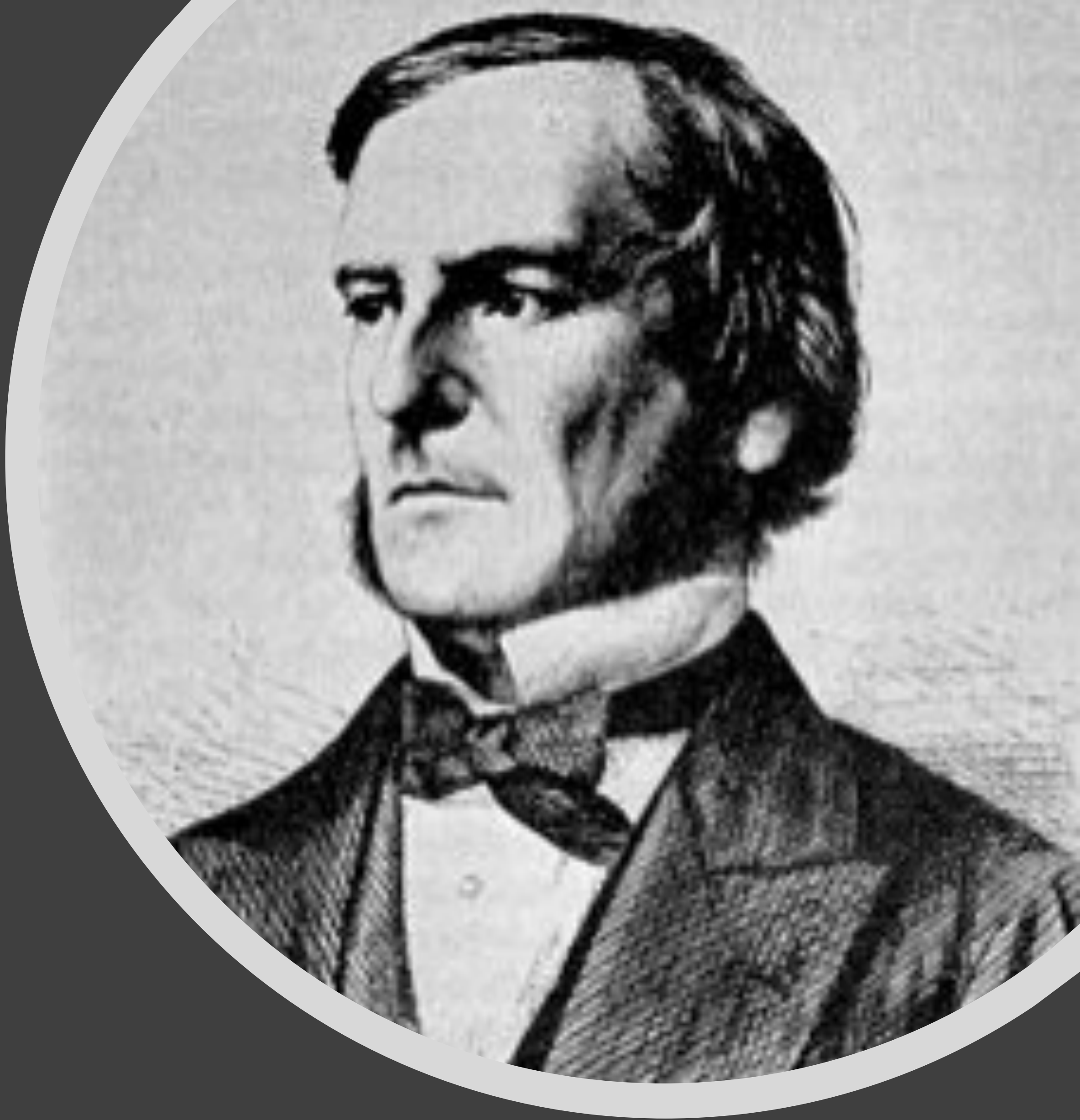
Ask questions!

- This is the best way to learn

## 1. A variable contains a value

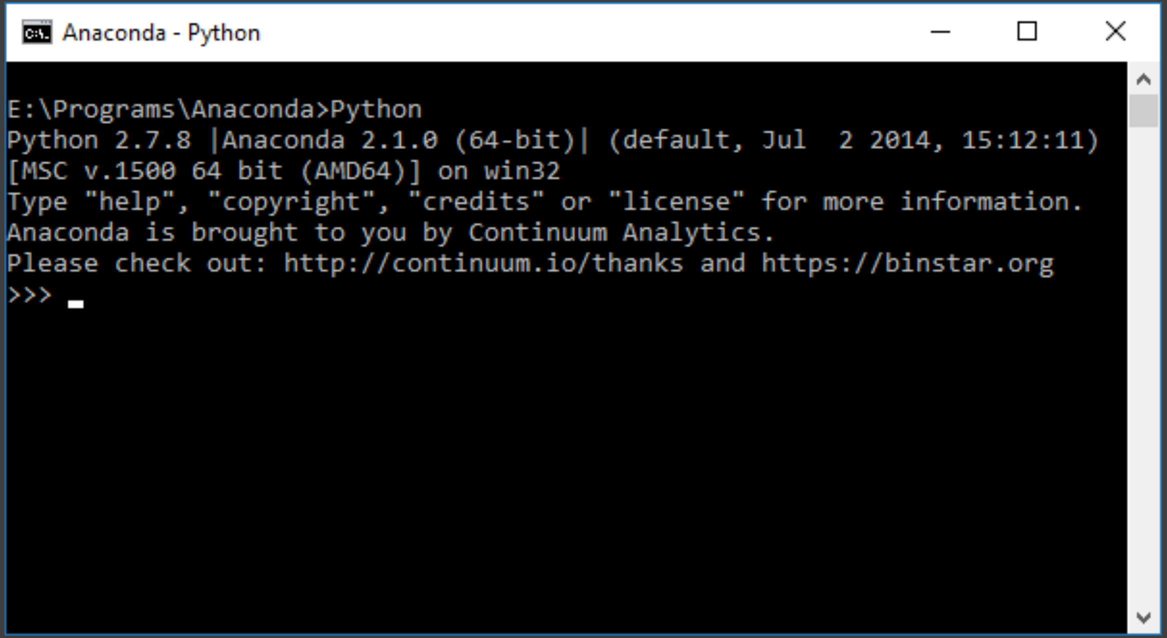
## 4 Basic types of values

- Integers (**int**): **-22**, **0**, **44**
  - No decimal points
- Real numbers (**float**, for “floating point”): **2.718**, **3.14159**
- Strings (**str**): **“Steph is the coolest!”**
- Truth values (**bool**, for “Boolean”): **TRUE**, **FALSE**


George Boole

## The Python Interpreter

- **Type Python to start running python**
- **Python prompts with '>>>'.**
- **To exit Python:** CTRL-D or type **exit()**



```
E:\Programs\Anaconda>Python
Python 2.7.8 |Anaconda 2.1.0 (64-bit)| (default, Jul  2 2014, 15:12:11)
[MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
>>>
```

## You type expressions. Python computes their values.

- 5
- 3+4
- 44/2
- 2**3 (what is a **?)
- 3*4+5*6
    - If precedence is unclear, use parentheses
- (72 – 32) / 9.0 * 5

## Important: Integers vs. Floats

- An operation on Integers will return an Integer
- An operation on Floats will return a Float

- What will each of these return?
  - 12 / 4
  - 13 / 4
  - 13.0 / 4.0
  - 13 / 4.0
  - Modulo operator (for Integers)
  - 13 % 4
  - 12 % 4

# Expressions

**Expression**: A data value or set of operations to compute a value.

|  |  |
|---|---|
| Examples: | `1 + 4 * 3` |
|  | `42` |

Arithmetic operators we will use:

|  |  |
|---|---|
| `+ - * /` | addition, subtraction/negation, multiplication, division |
| `%` | modulus, a.k.a. remainder |
| `**` | exponentiation |

**Precedence**: Order in which operations are computed.

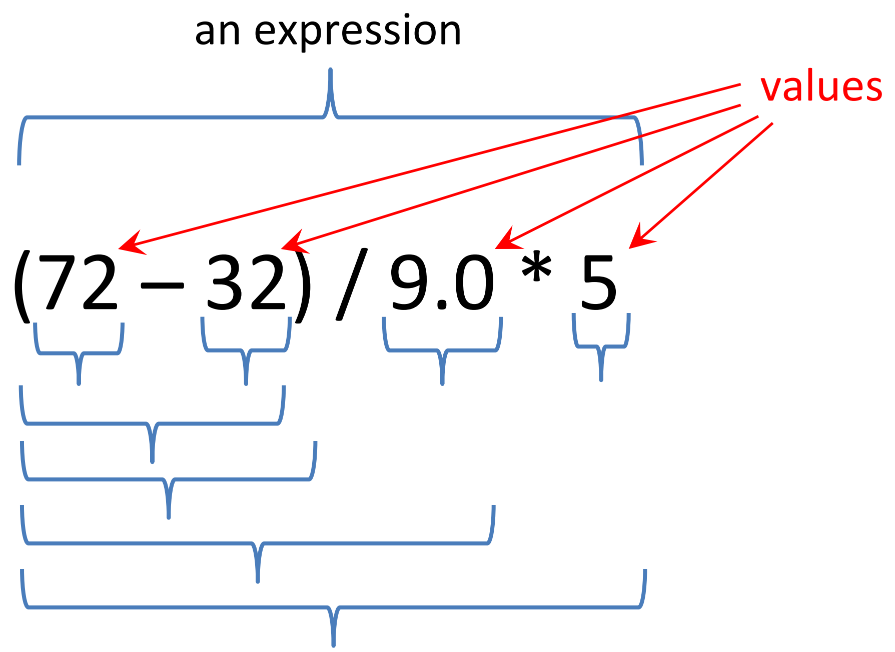
|  |  |
|---|---|
| `* / % **` | have a higher precedence than `+ -` |
| `1 + 3 * 4` | is `13` |

Parentheses can be used to force a certain order of evaluation.

|  |  |
|---|---|
| `(1 + 3) * 4` | is `16` |

# An expression is evaluated from inside out

- How many expressions are in this Python code?



(72 – 32) / 9.0 * 5

40 / 9.0 * 5

4.44 * 5

22.2

# Assignment

Now we have expressions that return values

How do we store these values?

Variables

Assignment Operator

X = 5

NOT an equality!

In Python, equality is represented as ==

## Variables hold values

- To assign a variable, use "*variableName* = *expression*"
  - **pi = 3.14**
  - **pi**
  - **Lost = 4815162342**
  - **Lost**
  - **22 = x** # Error! Why?
- Not all variable names are permitted

## Naming Rules

- Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.
- bob Bob _bob _2_bob_ bob_2 BOB
- 
- There are some reserved words:
    - and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, **import**, in, is, lambda, not, or, pass, print, raise, return, try, while

## Changing existing variables (“re-binding” or “re-assigning”)

- “=” in an assignment is *not* a promise of eternal equality
- Evaluating an expression gives a new (copy of a) number, rather than changing an existing one

```
x = 2 - 1
x
y = x
y
x = 5
x
y
```

## How an assignment is executed

1. Evaluate the right-hand side to a value
2. Store that value in the variable

```
x = 2
print x
y = x + 1
print y
x = 5
print y
z = x + 1
print x
print y
print z
```

State of the computer:

x: 5

y: 3

z: 6

Printed output:

```
2
3
3
5
3
6
```

To visualize a program's execution: [http://people.csail.mit.edu/pgbovine/python/tutor.html](http://people.csail.mit.edu/pgbovine/python/tutor.html)

## 2. Python performs operations

## Arithmetic Operations (already seen)

- 22 * 10
- 22 / 10
- 22.0 / 10
- 3 ** 2
- (5 +6) * (4 -3)

- x = 3
- y = x + 2
- z = x + y

- **What about this?**
- z = 2
- z - 5
- z

## More operations: Conditionals (return **`TRUE/FALSE`**)

```
22 > 4
22 < 4
22 == 4
x = 100                 # Assignment, not conditional!
x == 200
x == 100
22 = 4                  # Error!
x >= 5
not True
not (x >= 200)
3<4 and 7<6
4<3 or 5<6

temp = 72
is_liquid = temp > 32 and temp < 212
```

# More operations: “strings”

A string represents text, can use single or double quotations

```
“Python” or 'Python'
myName = “Steph"
```

Operations:

- Length:

```
len(myName)
```

- Concatenation:

```
“Steph" + ‘DeMora’                          #What will this do?
```

- More advanced: Containment/searching:

```
‘eph' in myName                             #What do these return?
“v" in myName
```

## Mathematical Operations

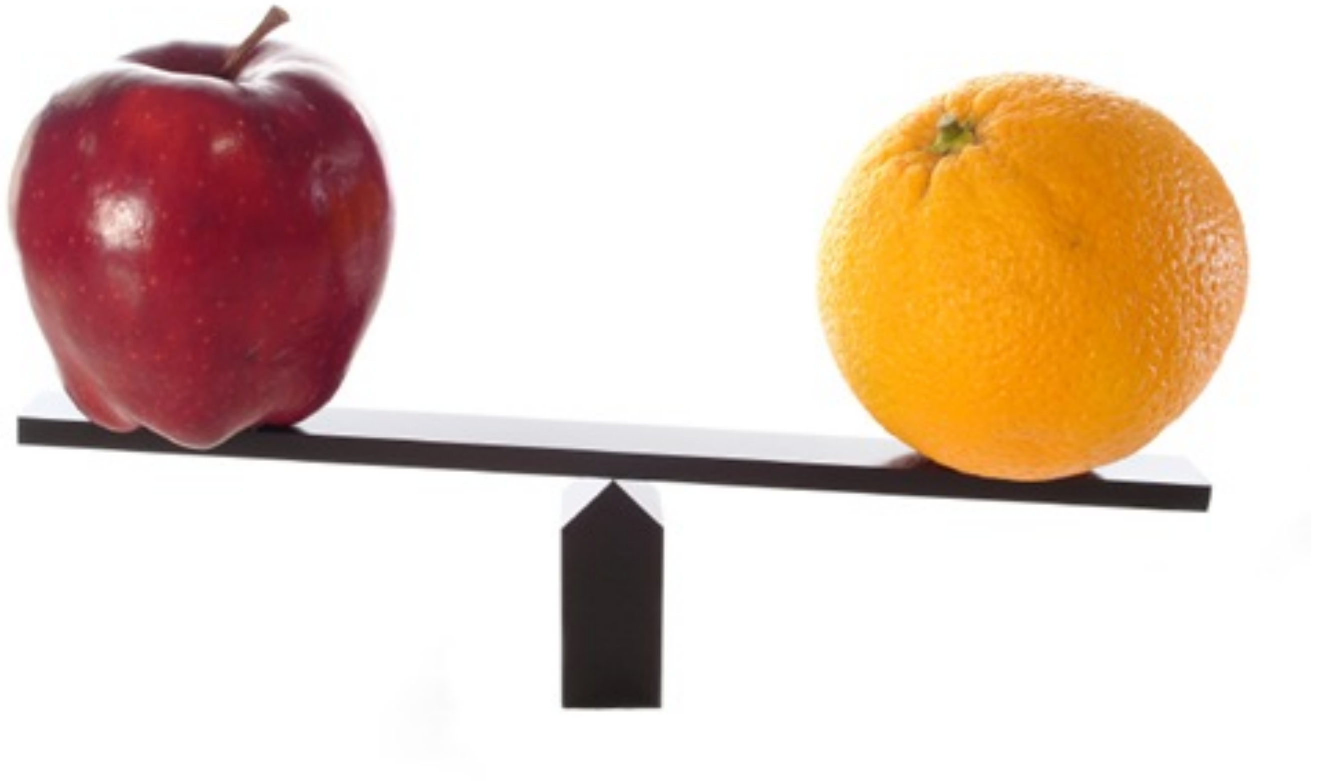
- Python has useful commands for performing calculations.

| Command name | Description |
|---|---|
| `abs(`***value***`)` | absolute value |
| `ceil(`***value***`)` | rounds up |
| `cos(`***value***`)` | cosine, in radians |
| `floor(`***value***`)` | rounds down |
| `log(`***value***`)` | logarithm, base *e* |
| `log10(`***value***`)` | logarithm, base 10 |
| `max(`***value1***`,` ***value2***`)` | larger of two values |
| `min(`***value1***`,` ***value2***`)` | smaller of two values |
| `round(`***value***`)` | nearest whole number |
| `sin(`***value***`)` | sine, in radians |
| `sqrt(`***value***`)` | square root |

| Constant | Description |
|---|---|
| `ea` | 2.7182818... |
| `pi` | 3.1415926... |

- To use many of these commands, you must write the following at the top of your Python program:
- `import math`

## 3. Different types act differently



## Operations behave differently on different types

- Moral: Python *sometimes* tells you when you do something that does not make sense.

- 3.0 + 4.0
- 3 + 4
- 3 + 4.0
- "3" + "4"
- 3 + "4" # Error
- 3 + True # What will this do?

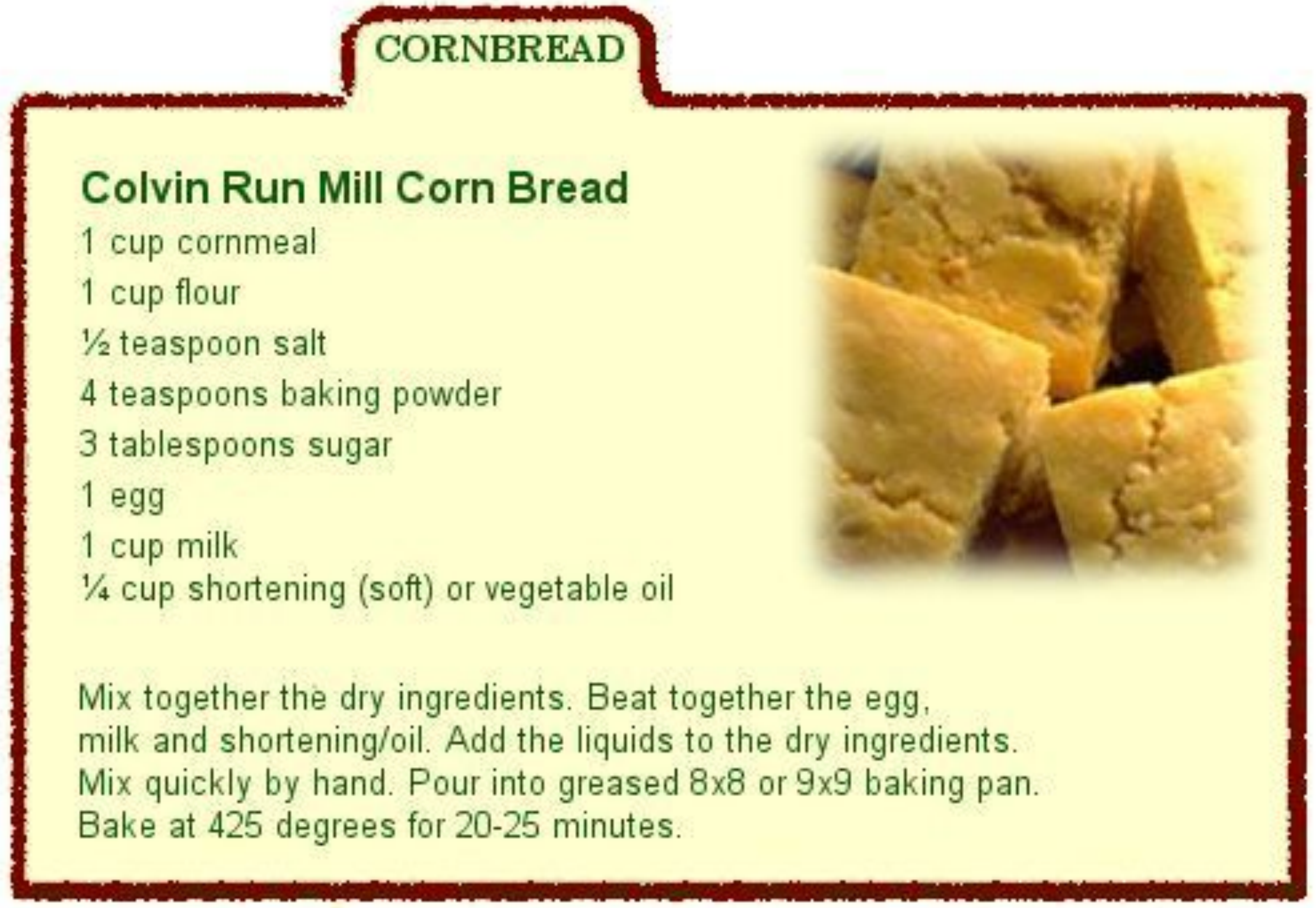
## Operations behave differently on different types

```
15.0 / 4.0
15 / 4
15.0 / 4
15 / 4.0
```

Type conversion:

```
float(15)
int(15.0)
int(15.5)
int(“15”)
str(15.5)
float(15) / 4
int(x)
```

## 4. A program is a recipe



CORNBREAD

**Colvin Run Mill Corn Bread**

1 cup cornmeal
1 cup flour
½ teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
¼ cup shortening (soft) or vegetable oil

Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.

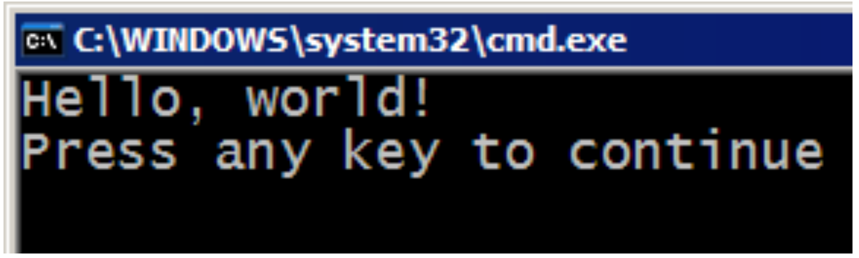
## What is a program?

- A program is a sequence of instructions
- The computer executes one after the other, as if they had been typed to the interpreter
- Saving as a program is better than re-typing from scratch

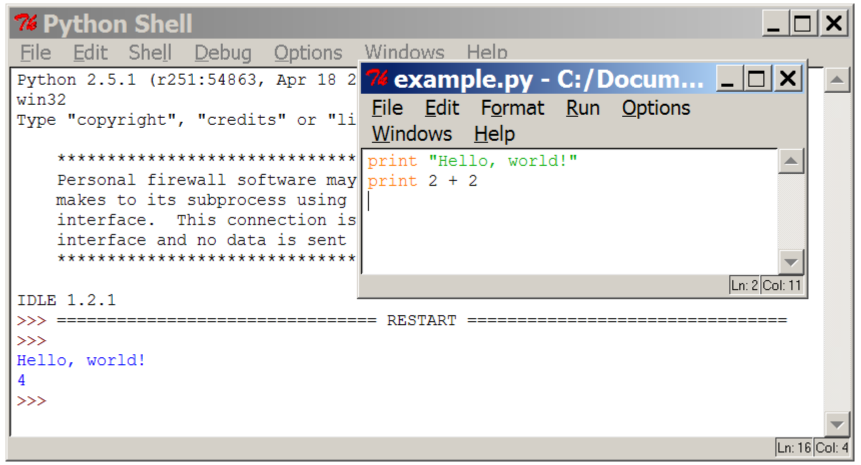
```
x = (enter some value here)
y = (enter some value here)
z = x + y
print “x=“, x
print “y=“, y
print "The sum of", x, “and", y, “is", z
```

# Programming Basics

- **code** or **source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **output**: The messages printed to the user by a program.
- **console**: The place where the user interacts with the program
    - Some source code editors pop up the console as an external window, and others contain their own console window.



```
Hello, world!
Press any key to continue
```

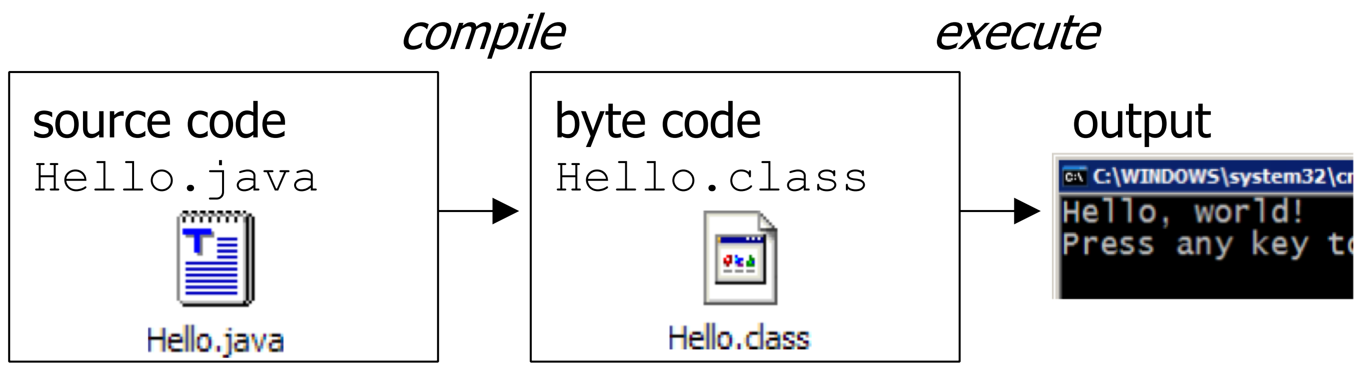

Python Shell
File Edit Shell Debug Options Windows Help
example.py - C:/Docum...
File Edit Format Run Options Windows Help
Ln: 2 Col: 11
Ln: 16 Col: 4

```
print "Hello, world!"
print 2 + 2
```

```
Python 2.5.1 (r251:54863, Apr 18 2
win32
Type "copyright", "credits" or "li

    ****************************
    Personal firewall software may
    makes to its subprocess using
    interface.  This connection is
    interface and no data is sent
    ****************************

IDLE 1.2.1
>>> ================================ RESTART ================================
>>>
Hello, world!
4
>>>
```

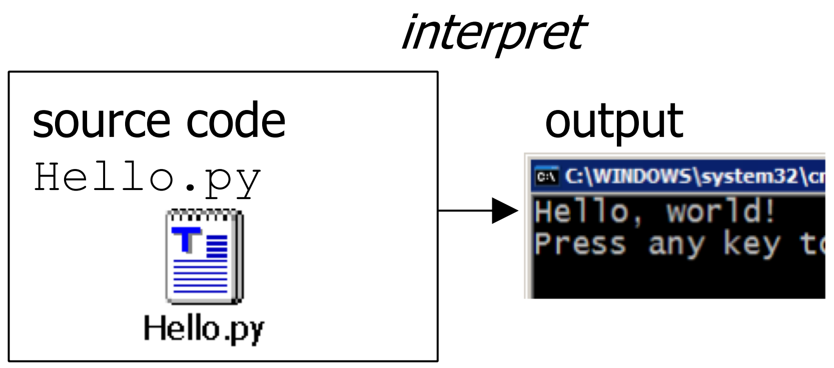
# Compiling and interpreting

Many languages require you to compile (translate) your program into a form that the machine understands.



```
Hello, world!
Press any key t
```

Python is instead directly interpreted into machine instructions.

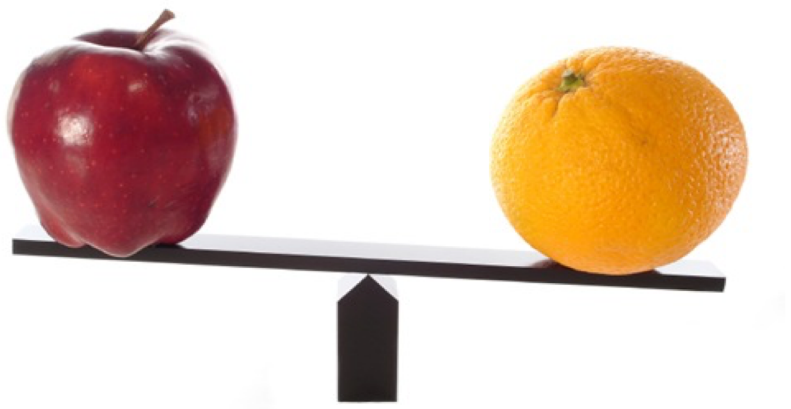


```
Hello, world!
Press any key t
```

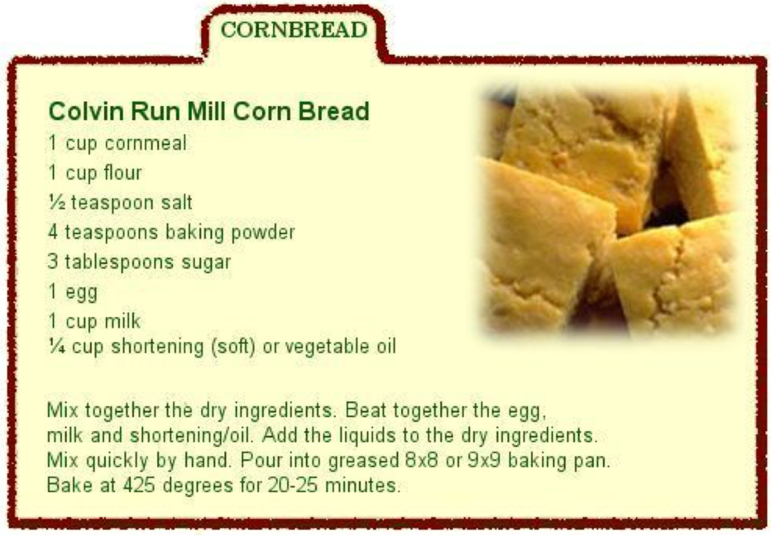
1. A variable contains a value

2. Python performs operations

3. Different types act differently



4. A program is a recipe



CORNBREAD

**Colvin Run Mill Corn Bread**

1 cup cornmeal
1 cup flour
½ teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
¼ cup shortening (soft) or vegetable oil

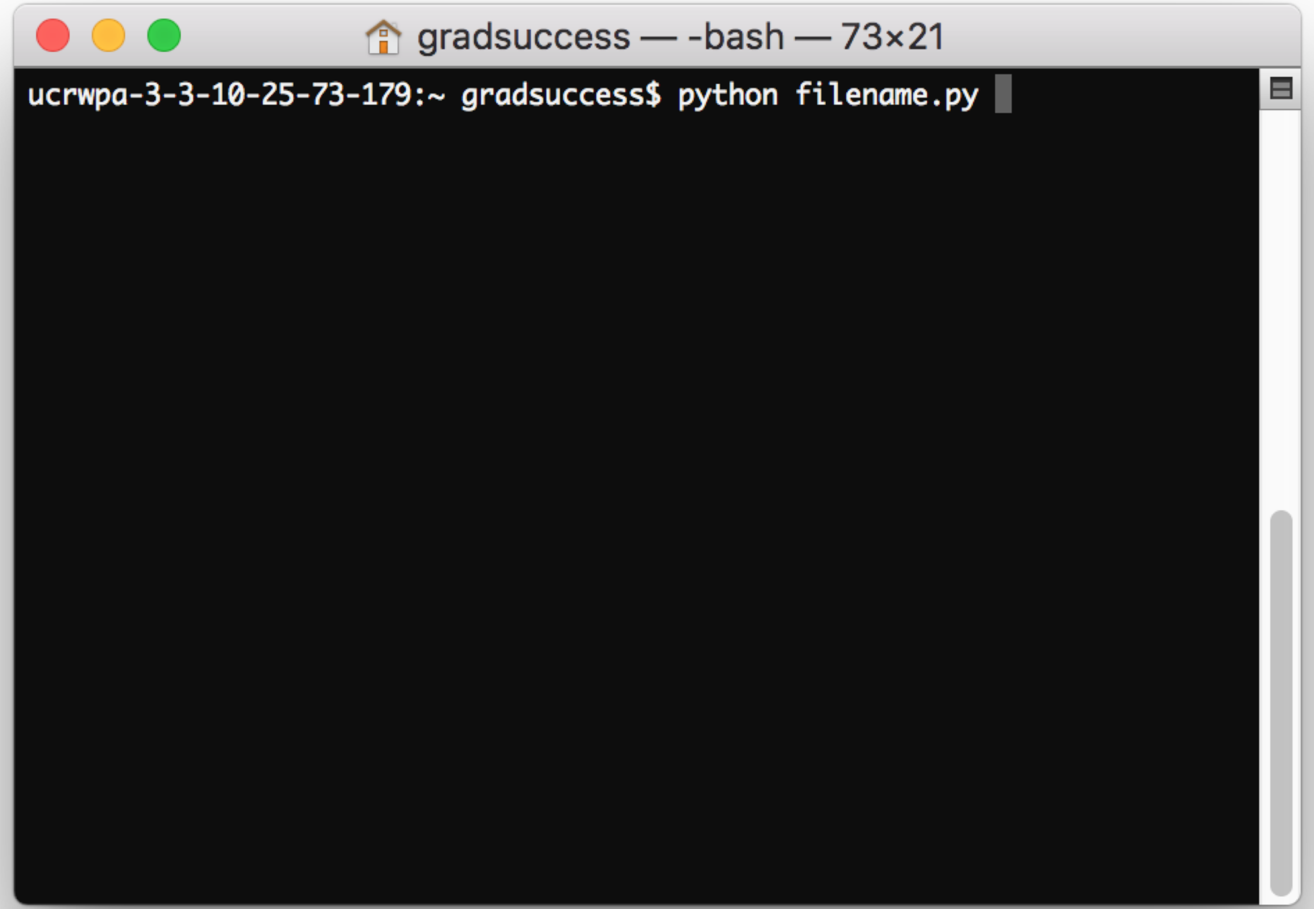
Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.

## Exercise 1:

```
x = (enter some value here)
y = (enter some value here)
z = x + y
print “x=“, x
print “y=“, y
print "The sum of", x, “and", y, “is", z
```

## Running programs on UNIX

% python filename.py



```
ucrwpa-3-3-10-25-73-179:~ gradsuccess$ python filename.py
```

## Comments

- Start comments with # – the rest of line is ignored.
- Can include a “documentation string” as the first line of any new function or class that you define.

```
# this is a comment
```

## `import` statements

- Import allows a Python script to import additional modules

```
import numpy
import os

or

import numpy as np
import os

or

import numpy as np, os
```

## Exercise 1:

```
#get inputs from the user
x = input(‘Provide a value for x:’)
y = input (‘Provide a value for y:’)

#calculate output
z = x + y

#print results to the user
print “x = “, x
print “y = “, y
print "The sum of", x, "and", y, "is", z
```

## Exercise 2: Fahrenheit to Celsius:

```
How could we take as input from the user
a Fahrenheit temperature, and then
convert it to Celsius?

Mathematical Equation for Celsius:
(F - 32) × 5/9

Think about:
Input and output
Integers vs Floats
```

## Exercise 2: Fahrenheit to Celsius:

```
#get inputs from the user
F = input(‘Provide the temperature in Fahrenheit:’)

#calculate output
#make sure you maintain floats!
#try C = (F-32) * 5 / 9
C = (F – 32) * 5.0 / 9.0

#print results to the user
print "The temperature in Celsius is", C
```

## Exercise 3: `if` statements

```
"if" provides a means of checking whether some condition is
met.
Tabs are used to show what should run if the condition is
met

if (5 < 6):
      print "five is less than six"

if (x == "banana"):
      print "x is banana"

if (y <= z):
      print "y is less than or equal to z"
      print "therefore I cannot choose the wine in front of
me"
```

## Exercise 3: `if` statements

```
Have the user input a number. If this number is greater than
1000, output a message “Wow that is a big number!”
```

## Exercise 3: `if` statements

```
#get inputs from the user
x = input('Provide a value:')

#print results to the user
if (x > 1000):
      print "Wow that is a big number!"


*ALTERNATIVELY:
if (1000 < x):
      print "Wow that is a big number!"
```

## Exercise 4: `else if`

```
else if provides a means to check alternate conditions:

Consider this code:

if (x < 5):
      print “x is pretty small”
if (x < 10):
      print “x is average”
if (x < 15):
      print “x is large”
if (x >= 15):
      print “x is huge”
```

## Exercise 4: `else if`

```
else if provides a means to check alternate conditions:

Consider this code:

if (x < 5):
      print “x is pretty small”
elif (x < 10):
      print “x is average”
elif (x < 15):
      print “x is large”
else:
      print “x is huge”
```

## Exercise 4: `else if`

**Let’s make a text-based adventure!**

**First line should be this:**

```
x = raw_input(‘You are trapped with five dragons.
                  (A)run (B)fight (C)make friends:’)
```

You should output a unique message based on whether the user types A, B, or C

**How do you handle when a user types something else?**

## Exercise 4: `else if`

```
#get inputs from the user
x = raw_input(‘You are trapped with five dragons.
                  (A)run (B)fight (C)make friends:’)

#print results to the user
if (x == “A”):
      print “You cannot escape. You die!”
elif (x == “B”):
      print “You cannot win. You die!”
elif (x == “C”):
      print “They do not want to be friends. You die!”
elif (x == “cheat”):
      print “You found the way to cheat. You win!”
else:
      print “Invalid choice. You die”
```

## Moving forward…

- There are many more tools available in Python that we can't cover here.
- If you want to move forward, the next things to look at would be:
  - Lists
  - For loops/while loops
  - Reading/Writing files

## Lists/arrays/DataFrames

- Save many values into a giant “list” (similar to a 1D array)
- Most likely needed for data analysis
- Can store any type into lists

```
myList = [1, 2.0, 3, ‘hello’, ‘bye’, 3.1415]
```

# Lists/arrays/DataFrames

- Save many values into a giant “list” (similar to a 1D array)
- Most likely needed for data analysis
- Can store any type into lists

```
myList = [1, 2.0, 3, ‘hello’, ‘bye’, 3.1415]
print myList[3]
```

## Lists/arrays/DataFrame

- Save many values into a giant “list” (similar to a 1D array)
- Most likely needed for data analysis
- Can store any type into lists

```
myList = [1, 2.0, 3, ‘hello’, ‘bye’, 3.1415]
print myList[3]
```

**Numpy**

```
numpy.array([1, 2, 3, 4])
```

## Lists/arrays/DataFrames

- Save many values into a giant “list” (similar to a 1D array)
- Most likely needed for data analysis
- Can store any type into lists

```
myList = [1, 2.0, 3, ‘hello’, ‘bye’, 3.1415]
print myList[3]
```

**Numpy**

```
numpy.array([1, 2, 3, 4])
numpy.array([1,2], [3,4])
```

## Lists/arrays/DataFrames

- Save many values into a giant “list” (similar to a 1D array)
- Most likely needed for data analysis
- Can store any type into lists

```
myList = [1, 2.0, 3, ‘hello’, ‘bye’, 3.1415]
print myList[3]
```

**Numpy**

```
numpy.array([1, 2, 3, 4])
numpy.array([1,2], [3,4])
```

## Lists/arrays/DataFrames

- Save many values into a giant “list” (similar to a 1D array)
- Most likely needed for data analysis
- Can store any type into lists

```
myList = [1, 2.0, 3, ‘hello’, ‘bye’, 3.1415]
print myList[3]
```

**Numpy**

```
numpy.array([1, 2, 3, 4])
numpy.array([1,2], [3,4])
```

**Pandas**

```
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
```

```
     col1  col2
0     1     3
1     2     4
```

## Python editors

- Eclipse with PyDev http://pydev.org/
- Sublime Text http://www.sublimetext.com/
- PyCharm http://www.jetbrains.com/pycharm/

- Why use a python editor
  - Syntax Highlighting
  - Error Detection
  - Auto-completion

- Last commonly used release before version 3
- Implements some of the new features in version 3, but fully backwards compatible

- Released a few years ago
- Many changes (including incompatible changes)
- More existing third party software is compatible with Python 2 than Python 3 right now

## Resources

Python's website
python.org/

Codeacademy
codecademy.com/tracks/python

GradQuant Resources
gradquant.ucr.edu/workshop-resources/

Stack Overflow website
stackoverflow.com/

# GradQuant

| | |
|---|---|
| Make | One-on-one Consultations: Make appointment at gradquant.ucr.edu |
| Keep | Keep an eye out for emails regarding more seminars gradquant.ucr.edu/workshop-resources/ |
| Remember | Remember to fill out the seminar survey. Thank you! |

## Other libraries/python-related tools

- PyCharm
- Sublime Text
- Numpy
- SciPy
- Seaborn
- Bokeh
- Plotly
- Pandas
- Scikit-learn
- Django
- Tensorflow (python interface)
- Anaconda
- Other libraries specific to your field (e.g. Biopython)