Python: The magic, the mystery

Angela Ambroz

Global Staff Training
August 24-28, 2015
Camp Yavneh, New Hampshire

ipa
INNOVATIONS FOR POVERTY ACTION

ABDUL LATIF JAMEEL
Poverty Action Lab
TRANSLATING RESEARCH INTO ACTION

Getting off the ground with Python

Angela Ambroz

Global Staff Training
August 24-28, 2015
Camp Yavneh, New Hampshire

Learning Objectives

At the end of this session, you will be able to:

1. Write a basic Python program.
2. Run it on your computer.
3. Be excited about integrating more Python stuff into your work and personal and romantic life.
4. Be empowered to learn more.

Session Outline

1. Disclaimers
2. Pep talk
3. History of Python
4. Context: how to run some Python
5. Getting set up
6. Example programs
7. `print “hello, syntax.”`
8. Automation
9. Libraries, libraries and more libraries
10. Resources for the future

Disclaimers

- This lecture assumes zero previous programming experience, but some Stata experience.
- I am merely an enthusiastic beginner.
- This .pptx is a little Mac-centric (sorry).
- Questions? Interrupt me.

How familiar are you with Python?

A. I've never heard of it.
B. I've heard of it, but never used it.
C. I've dabbled in it (e.g. wrote 1-2 programs, took an online course).
D. I code in it regularly.

Pep talk

Python is... *a popular programming language.*

Pep talk

Python is… *basically digital duct tape.*

Pep talk

Python is… *a way to put the tubes together.*

Pep talk

Python is… *a smarter way to talk to your computer.*

Pep talk

Python is… *a way to draw, calculate, analyze, scrape, spy, download, upload, loop…*

Pep talk

Python is... *very simple.*

Pep talk

Python is... *very sexy.*

History of Python

- Yes, it is named after Monty Python.

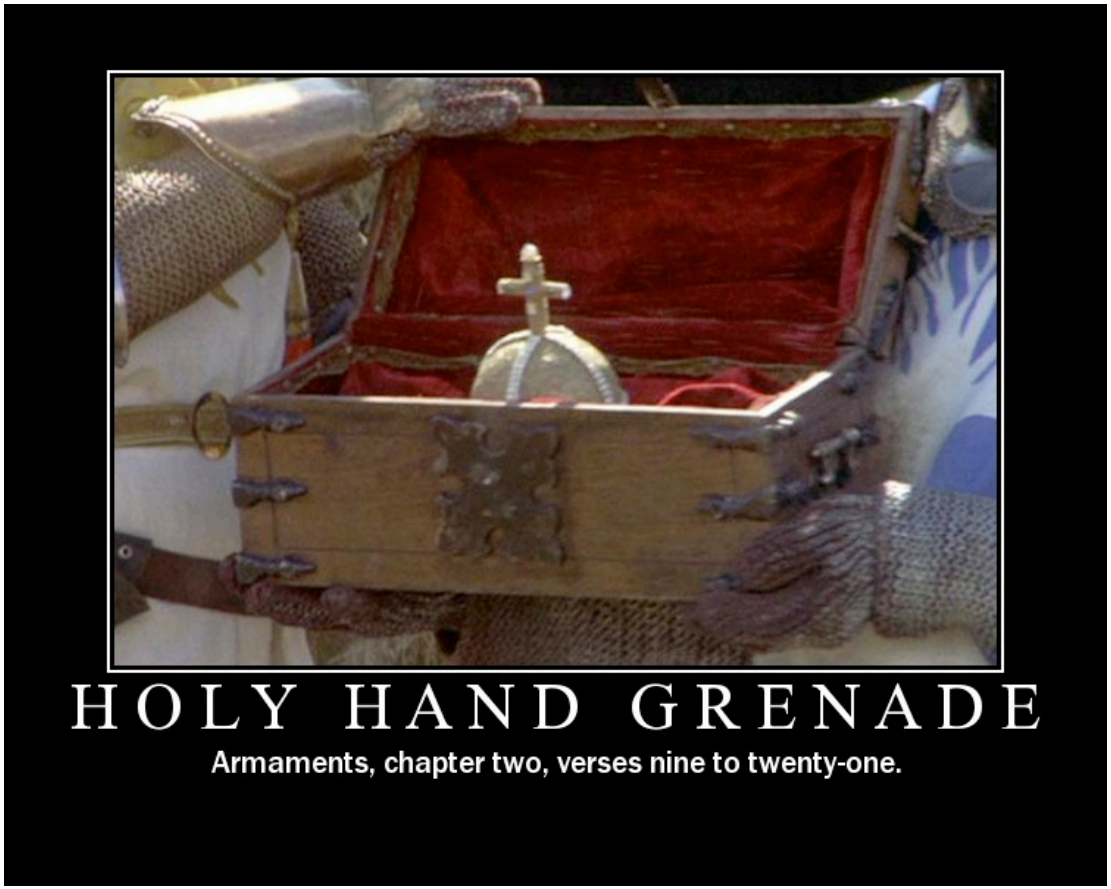
History of Python

- Yes, it is named after Monty Python.
- Officially invented in 1991, by this guy →
- Open source, free to learn.
- Companies using Python: Google, Disney, Yahoo!, NASA, IBM, **J-PAL/ IPA/EPoD…**

Guido van Rossum

History of Python

- Python 2.7 vs. Python 3 = a thing.



History of Python

PROs

1. Huge ecosystem.
2. It can do basically anything.
3. Huge community (way bigger than Stata!).
4. Flexible, powerful, **fun**.

CONs

1. Initial cost of getting set up and learning some syntax.
2. Compsci people say it's "slow"/inefficient.
3. Brogrammers.

Context: How to run some Python

1. Write your program in Sublime Text editor.
2. Save it as a `.py` file (e.g. `my-cool-program.py`).
3. Maybe test in a Python interpreter, if you wish
4. Open Terminal (OSX) or PowerShell (Windows)
5. Run your program by typing `python my-cool-program.py`.

A .py file is basically a .do file.

A. True
B. False

A Stata-Python phrasebook

A Stata-Python phrasebook

Stata	Python
1. Write your program in a plain text editor (like your .do file editor).	1. Write your program in a plain text editor (like Sublime Text).

A Stata-Python phrasebook

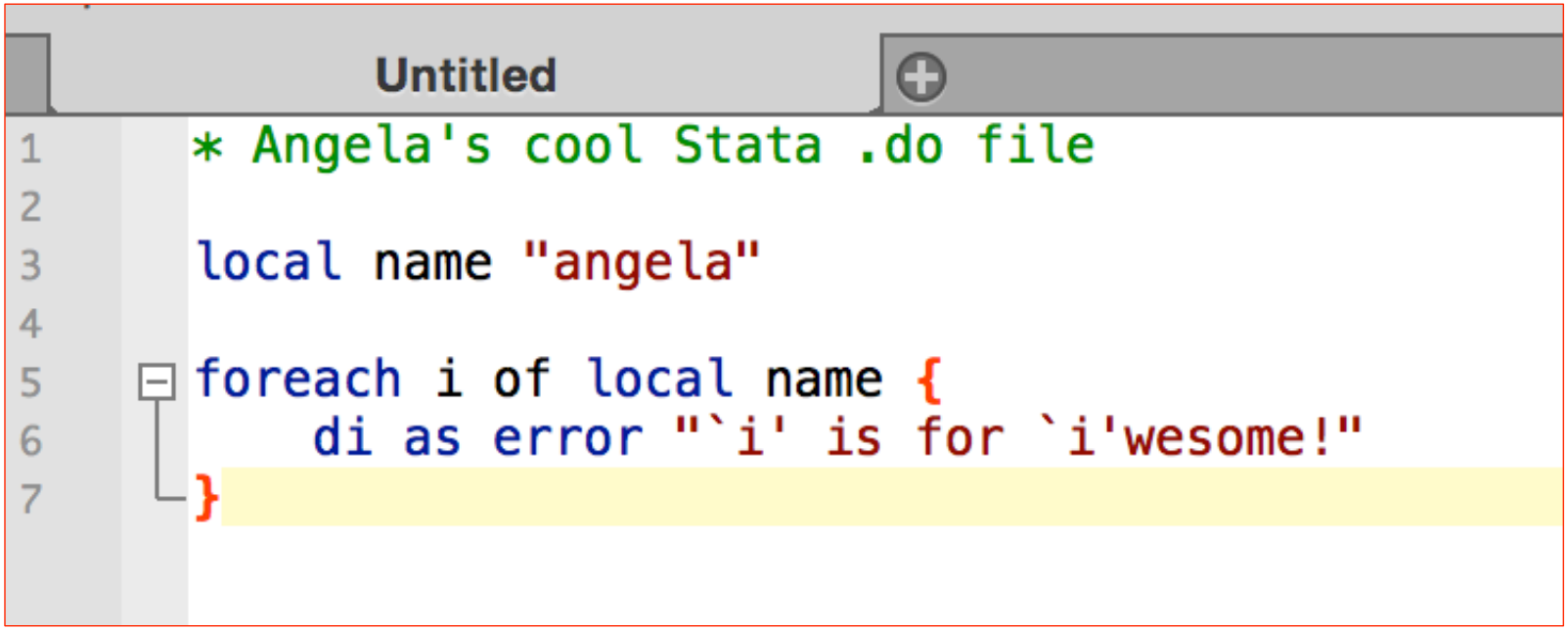
ABDUL LATIF JAMEEL Poverty Action Lab
TRANSLATING RESEARCH INTO ACTION

Stata

1. Write your program in a plain text editor (like your .do file editor).

Python

1. Write your program in a plain text editor (like Sublime Text).



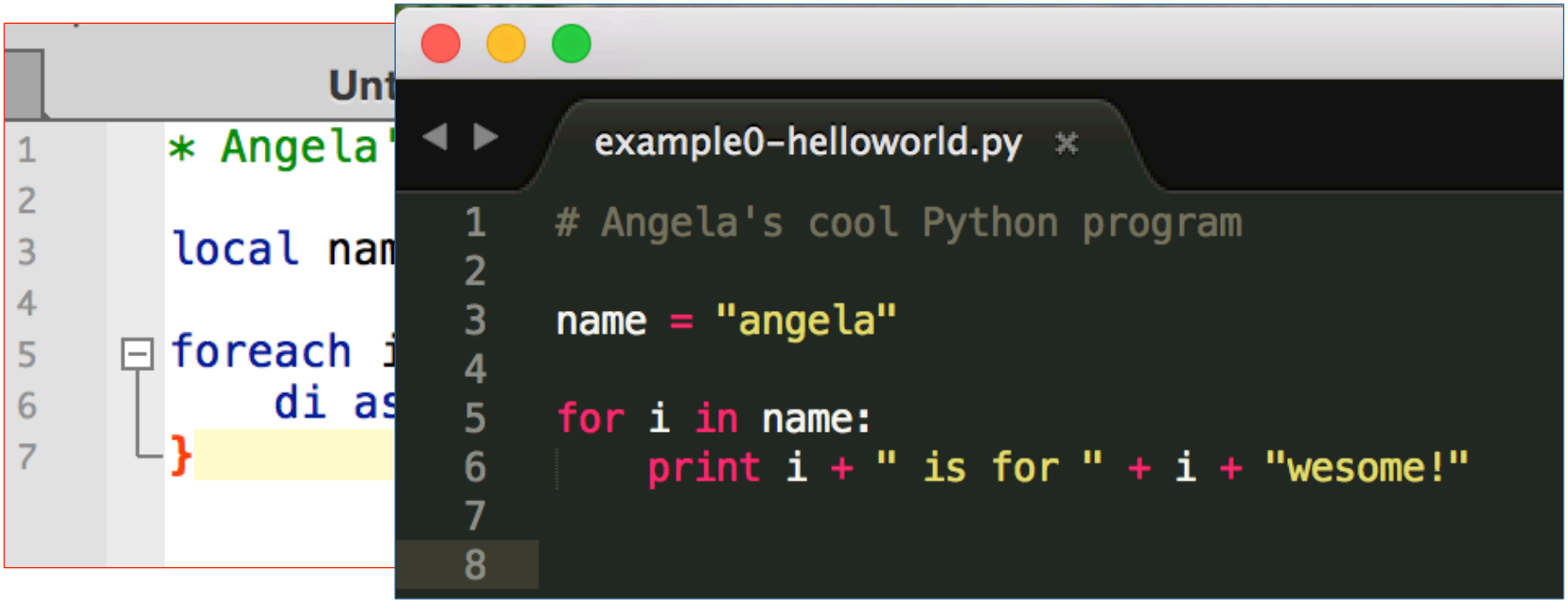
A Stata-Python phrasebook

Stata

1. Write your program in a plain text editor (like your .do file editor Text).

Python

1. Write your program in a plain text editor (like Sublime Text).



A Stata-Python phrasebook

ABDUL LATIF JAMEEL Poverty Action Lab
TRANSLATING RESEARCH INTO ACTION

Stata

1. Write your program in a plain text editor (like your .do file editor).
2. **Save it as a .do file.**

Python

1. Write your program in a plain text editor (like Sublime Text).
2. **Save it as a .py file.**

A Stata-Python phrasebook

Stata

1. Write your program in a plain text editor (like your .do file editor).
2. Save it as a .do file.
3. **If need be, test your code in the Command Editor.**

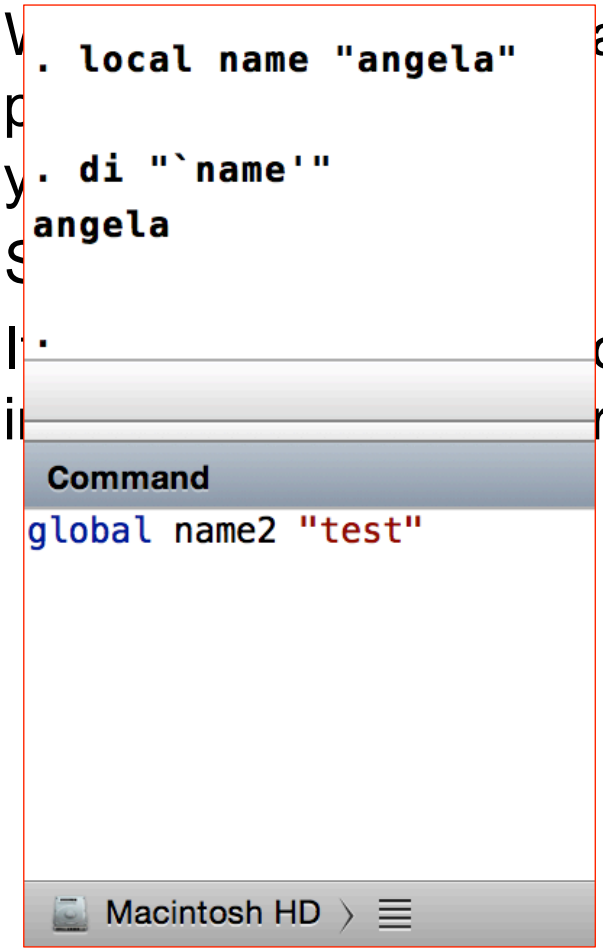
Python

1. Write your program in a plain text editor (like Sublime Text).
2. Save it as a .py file.
3. **If need be, test your code in a Python interpreter.**

A Stata-Python phrasebook

Stata

1. W… a
   p…
   y…
2. S…
3. I… ode
   i… r.



Python

1. Write your program in a plain text editor (like Sublime Text).
2. Save it as a .py file.
3. If need be, test your code in a Python interpreter.

A Stata-Python phrasebook

Stata

1. W… a p… S…
2. S…
3. I… ode i… r.

```
. local name "angela"

. di "`name'"
angela
```

Command

```
global name2 "test"
```

Macintosh HD

Python

```
Last login: Tue Aug 11 15:26:44 on ttys000
USSPhobos:~ angelaambroz$ python
Python 2.7.6 (default, Dec 19 2013, 19:43:31)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500
Type "help", "copyright", "credits" or "license"
>>> name = "angela"
>>> print name
angela
>>> for i in name:
...     print i
...
a
n
g
e
l
a
>>>
```

A Stata-Python phrasebook

ABDUL LATIF JAMEEL Poverty Action Lab
TRANSLATING RESEARCH INTO ACTION

Stata

1. Write your program in a plain text editor (like Sublime Text).
2. Save it as a .do file.
3. If need be, test your code in the Command Editor.
4. **In Stata's Command Editor (or in the .do file editor), run your .do file.**

Python

1. Write your program in a plain text editor (like your .do file editor).
2. Save it as a .py file.
3. If need be, test your code in a Python interpreter.
4. **Open Terminal/ Powershell, and run your .py file.**

A Stata-Python phrasebook

Stata

```
. do example0-helloworld.do

. local name "angela"


.
. foreach i of local name {
  2.          di as error "`i' is `i'wesome"
  3. }
angela is angelawesome


.
end of do-file
```

Python

am in a
like
or).
file.
our code in
eter.
owershell,
file.

A Stata-Python phrasebook

Stata **Python**

```
. do example0-helloworld.do
```

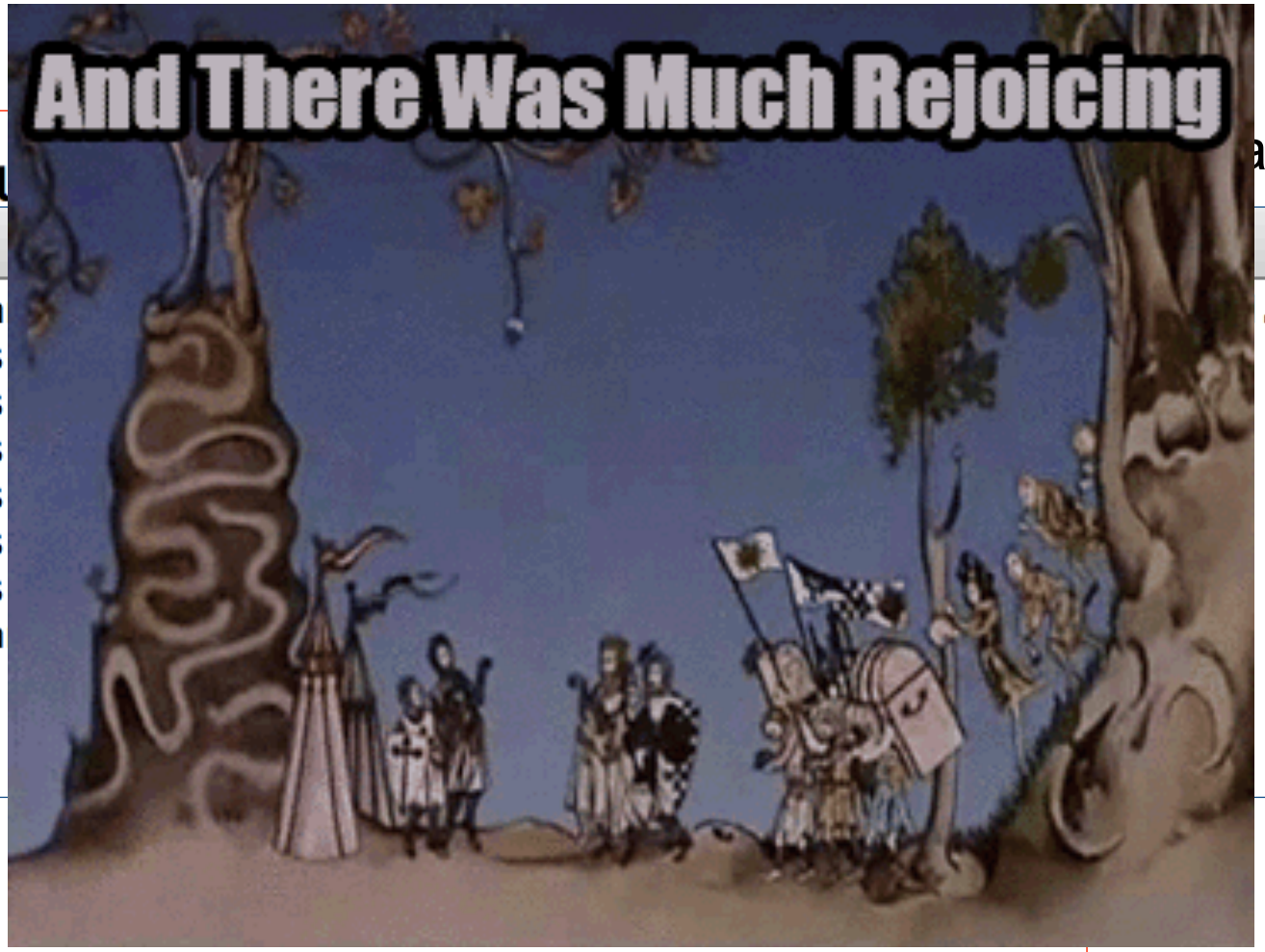
example programs — bash — 80×24

```
USSPhobos:example programs angelaambroz$ python example0-helloworld.py
a is for awesome!
n is for nwesome!
g is for gwesome!
e is for ewesome!
l is for lwesome!
a is for awesome!
USSPhobos:example programs angelaambroz$
```

```
.
end of do-file
```

A Stata-Python phrasebook

ABDUL LATIF JAMEEL Poverty Action Lab
TRANSLATING RESEARCH INTO ACTION




```
. do exampl
```

```
USSPhobos:exa
a is for awes
n is for nwes
g is for gwes
e is for ewes
l is for lwes
a is for awes
USSPhobos:exa
```

```
.
end of do-file
```

Let's get set up!

Getting set up

ABDUL LATIF JAMEEL Poverty Action Lab
TRANSLATING RESEARCH INTO ACTION

Mac OSX

1. Open Terminal (it's in the Applications→Utilities folder)
2. Type `which python --a` to confirm you have Python installed
3. Type `python` to open the Python interpreter in your Terminal
4. **You should have Python 2.7 installed!**
5. Type `quit()` (or `CTRL+D`) to exit the Python interpreter and get back to regular Terminal

→ reason to switch to Apple?

Windows

1. Open PowerShell.
2. Type `python` to see if you have Python installed already.
3. If not, download Python **2.7** from here: https://www.python.org/downloads/
4. Run the installer .exe.
5. Close and re-open PowerShell.
6. Type `python`. If you get an error message, you probably need to fix you PATH environment variable.
7. Type this into PowerShell and press enter:

```
[Environment]::SetEnvironmentVaria
ble("Path", "$env:Path;C:
\Python27", "User")
```

8. Close, re-open PowerShell and try `python` again.

After all this, reward yourself by downloading Sublime Text.

Do you have Python set up on your computer?

A. Yes
B. No
C. Who?

Example programs: random art

example0-helloworld.py × example1-randomwalk.py ×

```
#!/usr/bin/env python

import turtle       # import the "Turtle" library, for dr
import random       # import the "random" library, to ger

# Change this filepath variable to the filepath where you
DIR = "/Users/angelaambroz/Dropbox (Personal)/EPoD (Perso

# Define a function
def draw_stuff():

    # Create a "Turtle" canvas, with a white background
    window = turtle.Screen()
    window.bgcolor("white")

    # Add the JPAL logo to your image
    window.register_shape(DIR + "jpal.gif")

    # Create an instance of a "turtle", i.e. an imaginary
    # Call that instance "Brad", after Brad Pitt
    brad = turtle.Turtle()
    brad.shape(DIR + "jpal.gif") #Brad's head will be the
    brad.color("black") #Brad's line will be black
    brad.speed(20) #Brad will move at a speed of 20

    # Random art
    # Create a loop that runs 100 times
    for i in range(0,100):

        # Each time the loop runs, reset our two random r
```

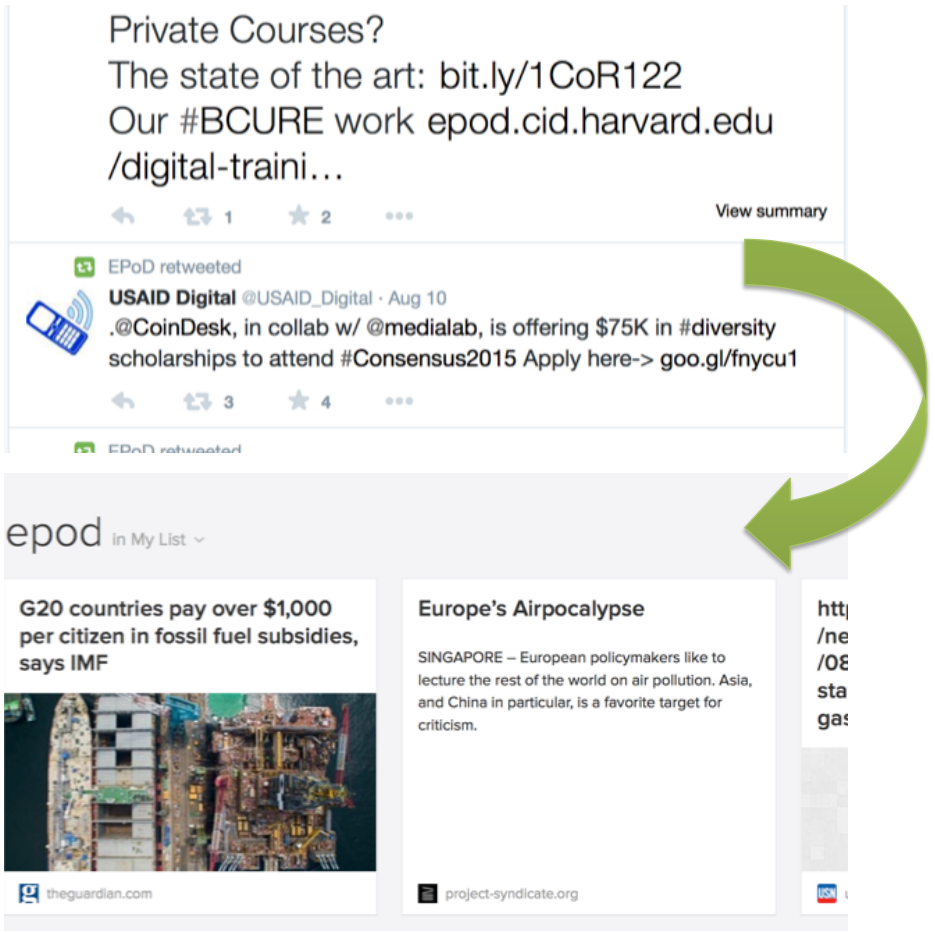
the python script

the output

Example programs: custom reading list

```
# Pulling the latest tweets and adding to Po

statuses = api.GetUserTimeline(epod)

pocket_instance = pocket.Pocket(pckey, acces

for s in statuses:
    for u in s.urls:
        print u.expanded_url
        if "paper.li" or "ln.is" in u.expand
            print "Skipping #smartpolicydesi
            pass
        else:
            print "Now adding something."
            pocket_instance.add(url=u.expand


print "Done! Check yo' Pocket."
```

the python script

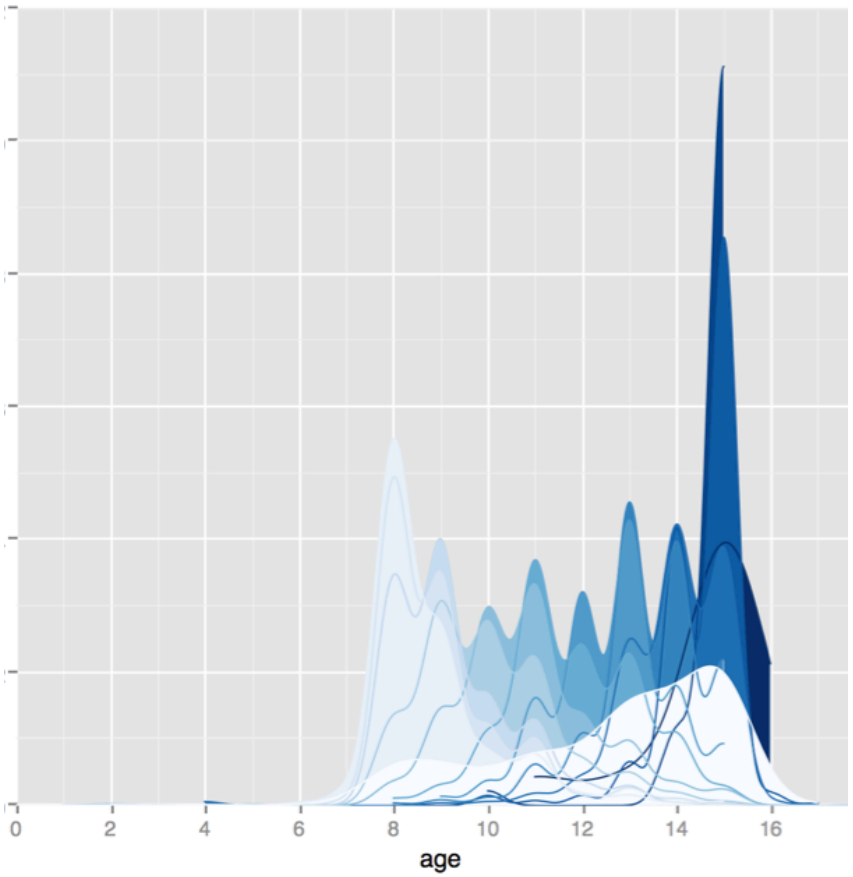


the output

Example programs: data visualizations

```
#!/usr/bin/env python

from ggplot import *
import pandas

# Hi, how are you?

MAHABS = "/Users/angelaambroz/Document


# Fine, fine. Thanks. You?

df = pandas.read_stata(MAHABS + "readi

print df.describe()

gg = ggplot(df, aes('age', color='stan

print gg
```

the python script



the output

Example programs: logistic regressions


```python
#!/usr/bin/env python

import numpy as np
import pandas as pd
import statsmodels.api as sm

DIR = "/Users/angelaambroz/python/_resource

df = pd.read_csv(DIR + "/titanic_data.csv")

df['intercept'] = 1.0

dummy_sex = pd.get_dummies(df['Sex'], prefi

controls = ['Survived', 'Pclass', 'SibSp',

data = df[controls].join(dummy_sex.ix[:,'se

print "The mean fare was ", data['Fare'].me

logit_controls = data.columns[1:]
logit = sm.Logit(data['Survived'], data[log
result = logit.fit()
```

the python script

```
The mean fare was  32.2042079686
Optimization terminated successfully.
         Current function value: 0.458333
         Iterations 6
                           Logit Regressi
==========================================
Dep. Variable:                  Survived   N
Model:                             Logit   D
Method:                              MLE   D
Date:                  Wed, 12 Aug 2015   P
Time:                          09:42:50   L
converged:                         True   L
                                          L
==========================================
                 coef    std err
------------------------------------------
Pclass        -0.8360      0.127     -6.5
SibSp         -0.2564      0.101     -2.5
Parch         -0.0888      0.113     -0.7
Fare           0.0034      0.002      1.4
intercept      3.1473      0.375      8.3
sex_male      -2.7594      0.196    -14.0
==========================================
```

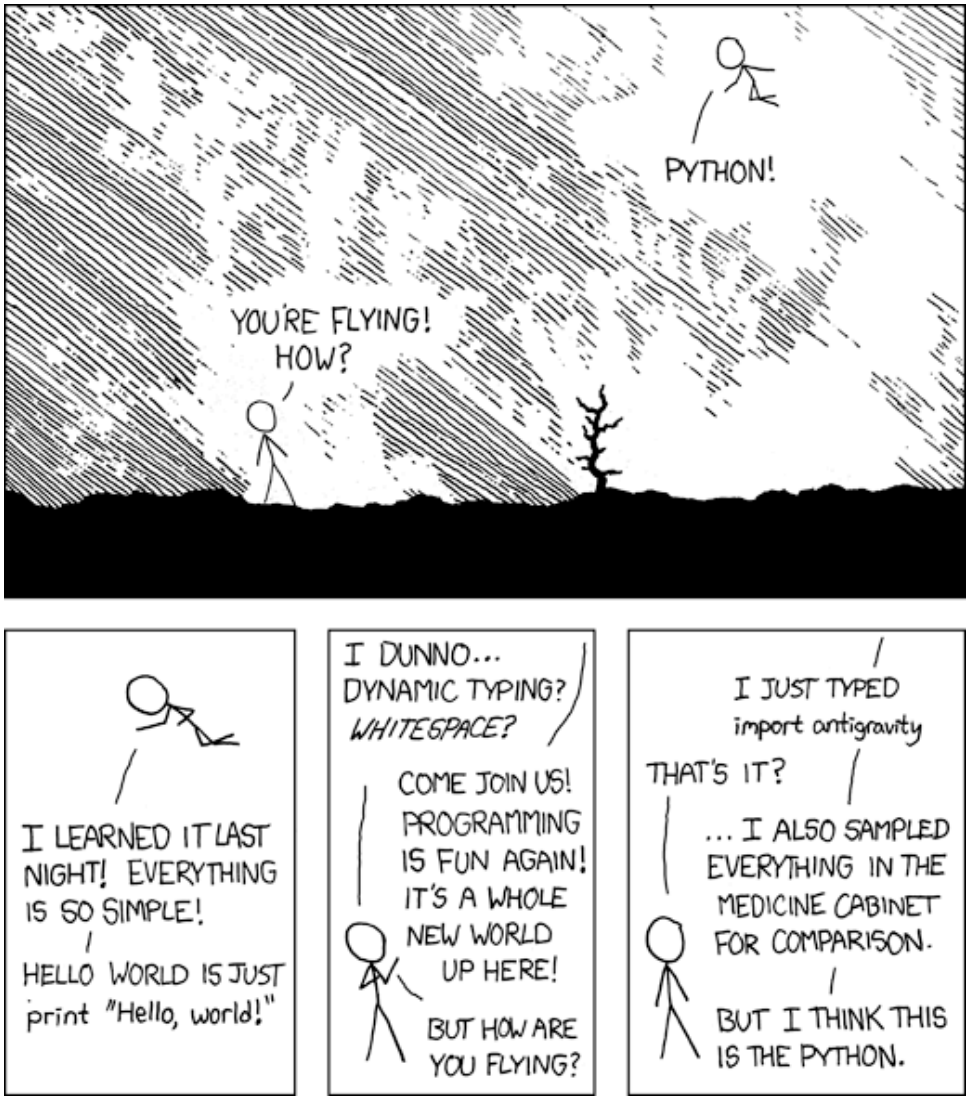
the output

`print "hello, syntax."`

- Python syntax is famously easy
- You'll recognize a lot from Stata
- Case matters
- Indents matters
- Colons, semi-colons, commas matter
- To paraphrase Allen B. Downey from *Think Python: How to Think Like a Computer Scientist*:

```
All computer
programming is just
loops and if/else
statements.
```

Obligatory xkcd cartoon



print “hello, syntax.”

example6-dataviz.py | example7-logreg.py | example2-backgrounds.py

```
1  #!/usr/bin/python
2
3  import praw              # Python Reddit API wrapper
4  import urllib            # for saving the top /r/Museum image URL
5  import random            # to randomly select subreddit
6  import datetime          # to name the file
7  import os                # to delete old files
8
9  from AppKit import NSWorkspace, NSScreen    # to change the desktop in 10.
10 from Foundation import NSURL                # same as above?
11
12
```

```
#!/usr/bin/python
```

The “**shebang**” - a flag to tell your computer to run this as a Python script (needed for Unix, maybe not for Windows)

print “hello, syntax.”

example6-dataviz.py | example7-logreg.py | example2-backgrounds.py

```
1   #!/usr/bin/python
2
3   import praw          # Python Reddit API wrapper
4   import urllib        # for saving the top /r/Museum image URL
5   import random        # to randomly select subreddit
6   import datetime      # to name the file
7   import os            # to delete old files
8
9   from AppKit import NSWorkspace, NSScreen    # to change the desktop in 10.
10  from Foundation import NSURL                # same as above?
11
12
```

```
import [some library]
```

Libraries(/modules) in Python are similar to `.ado` files in Stata; they are user-written Python programs that make your life much easier.

print “hello, syntax.”

example6-dataviz.py | example7-logreg.py | example2-backgrounds.py

```
1   #!/usr/bin/python
2
3   import praw          # Python Reddit API wrapper
4   import urllib        # for saving the top /r/Museum image URL
5   import random        # to randomly select subreddit
6   import datetime      # to name the file
7   import os            # to delete old files
8
9   from AppKit import NSWorkspace, NSScreen    # to change the desktop in 10.
10  from Foundation import NSURL                # same as above?
11
12
```

For efficiency reasons, you have to tell Python which libraries you’ll be using in each script. You **import** them at the top.

print “hello, syntax.”

example6-dataviz.py | example7-logreg.py | example2-backgrounds.py

```
1   #!/usr/bin/python
2
3   import praw             # Python Reddit API wrapper
4   import urllib           # for saving the top /r/Museum image URL
5   import random           # to randomly select subreddit
6   import datetime         # to name the file
7   import os               # to delete old files
8
9   from AppKit import NSWorkspace, NSScreen    # to change the desktop in 10.
10  from Foundation import NSURL                # same as above?
11
12
```

```
from [library] import [methods]
```

You can also only import **some parts of a library**, if you want to make things speedier (some libraries are big and bulky).

print “hello, syntax.”

```
# a comment
```

As in Stata, you can comment out code by using either the `#` symbol (like `*` in Stata), or three quotation marks (“”“) for multi-line comments (`/* */` in Stata)

```
16
17  # Some globals
18
19  DIR = "/Users/angelaambroz/python/"
20  SUBREDDITS = ["museum", "ColorizedHistory", "TheWayWeWere",\
21      "HumanPorn", "MacroPorn", "FuturePorn", "spaceporn", "historyporn"]
22
```

print “hello, syntax.”

```
var = “some stuff”
var = [“some”, “long”, “list”]
```

You define **variables** (equiv. to `globals`/`locals` in Stata) very simply, with one equals sign (=).

```
16
17  # Some globals
18
19  DIR = "/Users/angelaambroz/python/"
20  SUBREDDITS = ["museum", "ColorizedHistory", "TheWayWeWere",\
21      "HumanPorn", "MacroPorn", "FuturePorn", "spaceporn", "historyporn"]
22
```

print “hello, syntax.”

```
var = “some stuff”
var = [“some”, “long”, “list”]
```

Variables can be integers, floating points, tuples, strings, lists/arrays…

```
16
17  # Some globals
18
19  DIR = "/Users/angelaambroz/python/"
20  SUBREDDITS = ["museum", "ColorizedHistory", "TheWayWeWere",\
21      "HumanPorn", "MacroPorn", "FuturePorn", "spaceporn", "historyporn"]
22
```

`print "hello, syntax."`

`library.method()`

After importing a library (such as `random`), we can access its functions (or "methods", such as `choice`).

To do this, you use **chain(-like) syntax**.

```
5   import random
```

```
22
23
24  SUB = random.choice(SUBREDDITS)
25
```

`print "hello, syntax."`

`library.method()`

After importing a library (such as `random`), we can access its functions (or "methods", such as `choice`).

To do this, you use **chain(-like) syntax**.

```
5   import random
```

```
22
23
24  SUB = random.choice(SUBREDDITS)
25
```

print “hello, syntax.”

```
# Deleting the old stuff

if os.path.isfile(DIR + YESTERDAY + ".jpg"):
    print "Deleting desktop background from " + YESTERDAY + "."
    os.remove(DIR + YESTERDAY + ".jpg")
else:
    print "Apparently, there is no desktop image from " + YESTERDAY + "."
```

```
if [something]:
    [do something else]
```

Python uses **white space** to indicate loops or if/ else statements. Just remember to **tab/indent**.

print “hello, syntax.”

```
# Deleting the old stuff

if os.path.isfile(DIR + YESTERDAY + ".jpg"):
    print "Deleting desktop background from " + YESTERDAY + "."
    os.remove(DIR + YESTERDAY + ".jpg")
else:
    print "Apparently, there is no desktop image from " + YESTERDAY + "."
```

Stata-Python phrasebook:

```
foreach i in 1 2 3 {
    di `i'
}
```

```
for i in range(1,4):
    print i
```

print “hello, syntax.”

```
def
some_function(para
meter):
     [do stuff]

some_function(spec
ific_parameter)
```

Chunks of reusable code are functions - which you can **define** and **call**. Also indented.

```
# Setting the desktop background

def change_desktop_background(file):

    print "Now changing desktop background..."

    ws = NSWorkspace.sharedWorkspace()

    print ws

    file_url = NSURL.fileURLWithPath_(file)

    print file_url

    for screen in NSScreen.screens():
        print screen
        (result, error) = ws.setDesktopImageURL_for

change_desktop_background(DIR + TODAY + ".jpg")
```

`print "hello, syntax."`

- That's basically it!
- Two important concepts I'm **not** covering here, but you'll eventually want to learn about:
  - `Classes`
  - `Object-oriented programming`

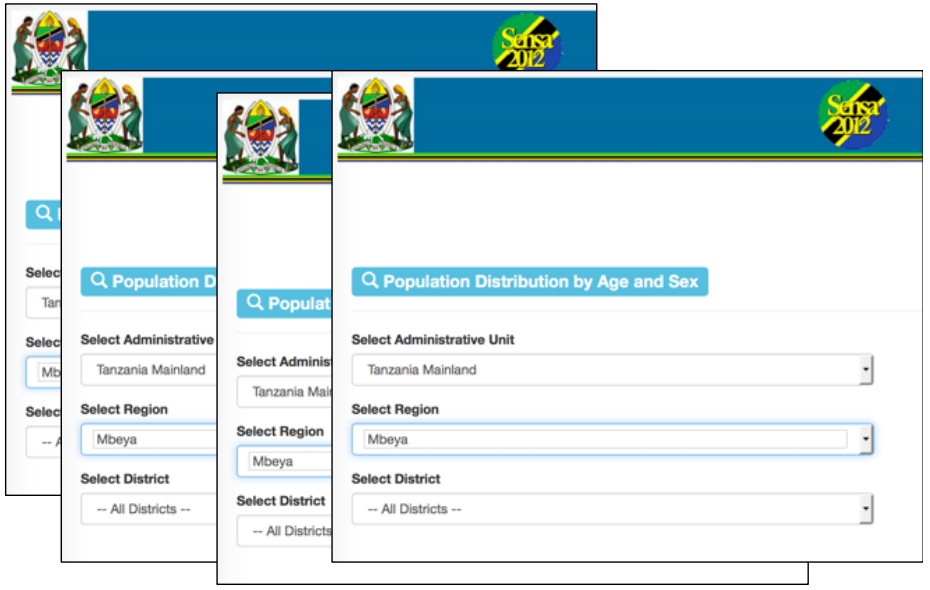
Automatons: scheduling tasks

A hypothetical situation

Things are lonely in the village of Akelapur.

Professor Knowus Everythingus, your PI, wants up-to-date administrative data from the 1995-style state government website. The only way you can get it is by copy+pasting rows and rows of awful HTML tables into an Excel every single day.

You feel like a sad minion doing admin tasks. Life is hard.



Automatons: scheduling tasks

There is a better way.

It is the ~~Python way~~.

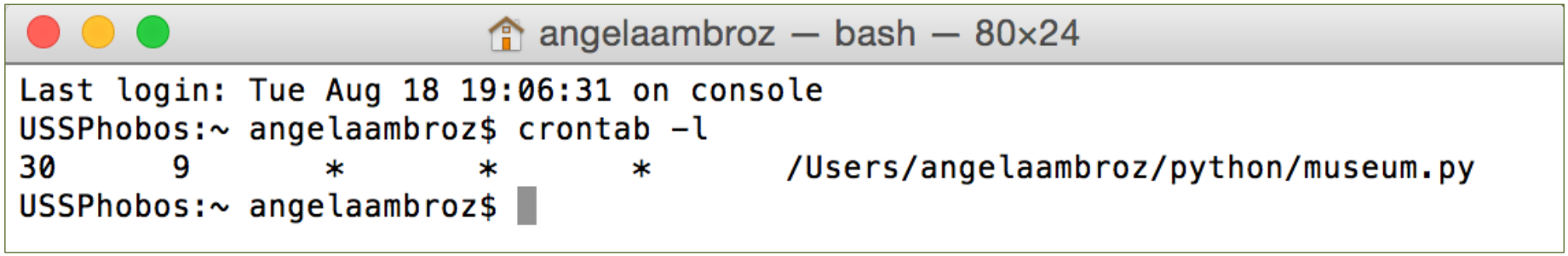
1. Write a simple web scraping script in Python (e.g. you can use a library called `BeautifulSoup`).
2. Schedule it to run every day.

```
79
80   #a similar issue on the nrega site
81   district = "19"
82   mandal = "03"
83   panch = "17"
84   exe = "muster"
85   url = ("http://nrega.ap.g
86             "PaymentsWork_eng
87             "Print_eng&Distri
88             +panch+"&exec="+e
89
90   response = grab_data_with
91   data  = response.read()
92
93   output = open('C:/Users/m
94                 mandal + panc
95   output.write(data)
96   output.close()
97   print "done"
98
99
```

Automatons: scheduling tasks

You can schedule Python scripts to run using:

- `cron` (OSX) - easy but deprecated
- `launchd` (OSX) - tricky but more powerful
- Windows Task Scheduler (Windows)



And now for something completely different

Resources: Libraries

library	what it does	how you get it
`shutil`	Copying and deleting files.	Comes with Python's standard library
`os`	More file/folder manipulation.	ibid.
`BeautifulSoup`	Web scraping (without an API).	http://www.crummy.com/software/BeautifulSoup/
`requests`	More web scraping (with an API).	http://www.python-requests.org/en/latest/user/install/#install
`ggplot`	Data visualizations.	http://ggplot.yhathq.com/install.html
`pandas (+ numpy/scipy)`	Replace Stata.	http://pandas.pydata.org/pandas-docs/stable/install.html
`Twilio`	Make phone calls, send SMS messages.	https://www.twilio.com/docs/python/install
`smptlib + email`	Read/write/send/receive email.	Comes with Python's standard library
`PyQGIS`	Maps and other geo things.	http://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/intro.html
`pip`	Installs all of the above!	Should have it - otherwise, https://pip.pypa.io/en/latest/installing.html

Resources: Learning more

- The best way to learn is by picking an **interesting project to build**.
- Basic rule of thumb: **Python can do anything**, start simple/small and then grow.
- The hardest part is thinking up a good project!
- Happy to talk about this.
- Some ideas:
  - automagically change your desktop background to the NASA picture of the day, or the Atlantic Photo picture of the day
  - scrape the JPAL website and turn it into a beautiful data visualization using `ggplot`

Resources: Learning more

- Another way to learn is by **solving puzzles**
- PROs: Lots of existing Python puzzles out there
- CONs: You might find the puzzles themselves boring
- Links:
  - http://puzzles.bostonpython.com/
  - https://projecteuler.net/ (logic puzzles)
  - https://codecombat.com/ (video game)

Resources: Learning more

- The third best way to learn is **by working through an online course**
- Find one that is **project-based** (rather than syntax-based)
- Links (in order of recommendedness):
  - Programming Foundations with Python (**Udacity**)
  - Codecademy, Code School
  - Coursera, edX

Resources: Learning more

- The fourth best way is **by reading a book about it**
- Lots of free, online books, e.g.:
  - Learn Python the Hard Way (http://learnpythonthehardway.org/book/index.html)
  - Think Python (http://www.greenteapress.com/thinkpython/)

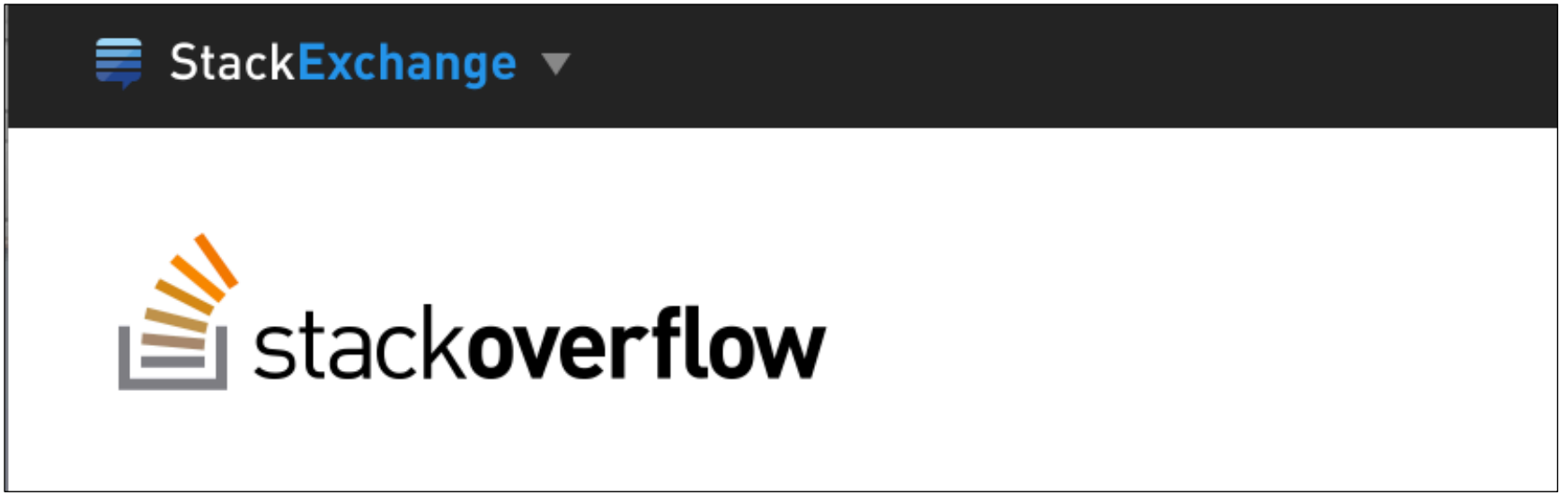
Resources: Dealing with errors

- A journey through Python is a journey through error messages.
- Do not fear the error message.
- Stuck?
  - Method 1:

  Rubber ducking

Resources: Dealing with errors

Method 2: *Stack Overflow*



www.stackoverflow.com

Resources: Understanding the tubes

- It'll be easier to learn Python if you understand some **context.**
- i.e. What is the internet? What is email? What is your computer? What is computing?
- I recommend getting at least a basic handle on (i.e. what is it? what can it do?):
  - HTML and CSS
  - JavaScript
  - APIs
  - Basic command line (shell)
  - Version control (git)

Homework (sort of)

For those who would like to begin their Python journey:

1. Complete your computer setup.
   - Do you have **Python 2.7+** installed?
   - Do you have **Sublime Text** downloaded?
   - Have you found **Terminal/PowerShell**?
2. Look through the folder **example-programs.**
3. Open each .py file in Sublime Text and read through. What do you think the output will be?
4. Run each .py file. What's the output?
5. Edit these to learn more about how they work.
6. Come talk to me about Python project ideas.

Questions?

An afternoon to learn, a lifetime to master.

Send complaints to:
Angela Ambroz
Senior Research and Training Manager, EPoD
angela_ambroz@hks.harvard.edu