GENERAL ASSEMBLY

INTRODUCTION TO PYTHON

Jonathan Blum
jon@jonblum.net

JONATHAN BLUM
MATCH EDUCATION

- Web and Mobile tools for exporting curriculum and best practices
- Previously, Lead at Galatea Associates, creating real-time inventory management platforms for banks

YOUR TURN

- Name
- Brief Background
- Programming skill (1-10)
- What you want to get out of this evening’s class

DECISION TIME

Introduction to Python

vs.

Introduction to Programming...
Featuring Python

AGENDA

- History of Python
- Why Python? Why not?
- Installing and Running Python
- Exercise: Set Up Your Environment
- Programming in Python: The Basics
- Exercise: Weather Tracker
- Questions & Next Steps

INTRODUCTION TO PYTHON

HISTORY OF PYTHON

HISTORY

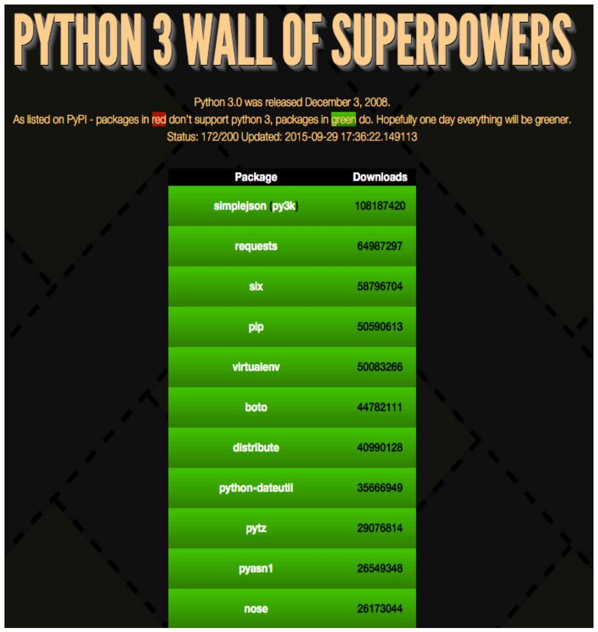
- First released in 1991 by Dutch programmer Guido van Rossum
- CWI -> Google -> Dropbox
- Derived from teaching language ABC
- Named after Monty Python
- 2.0 in 2000
- 3.0 in 2008

(Want more? https://www.youtube.com/watch?v=ugqu10JV7dk)

PYTHON 3: THE HEADACHE

- Released in 2008
- Breaking changes: printing, objects
- Slow library support (but it's finally there)
- Python 2 still supported
- Many new features backported
- Adoption rate < 20% (but it depends!)

- <u>Use Python 3. Unless you can't.</u>



PYTHON 3 WALL OF SUPERPOWERS

Python 3.0 was released December 3, 2008.
As listed on PyPI - packages in red don't support python 3, packages in green do. Hopefully one day everything will be greener.
Status: 172/200 Updated: 2015-09-29 17:36:22.149113

Package	Downloads
simplejson (py3k)	108187420
requests	64987297
six	58796704
pip	50590613
virtualenv	50083266
boto	44782111
distribute	40990128
python-dateutil	35666949
pytz	29076814
pyasn1	26549348
nose	26173044

https://python3wos.appspot.com/

WHY PYTHON?

Python is **many things**.

PYTHON IS GENERAL-PURPOSE

- Data Science (vs. R, SAS…)
- Scripting (vs. Bash, Perl, Ruby…)
- Web Development (vs. Ruby, JavaScript…)
- Hard Sciences (vs. Fortran et al…)
- Learning (vs. Logo, Scratch…)

- “Python is everyone’s second-favorite language”

PYTHON IS MULTI-PARADIGM

- Want to write strictly **object-oriented** code (like Java/C#?)

```
class Dog(object):
   def bark(self):
       …
```

- Want to write **functional** code (like Javascript/Lisp/Haskell?)

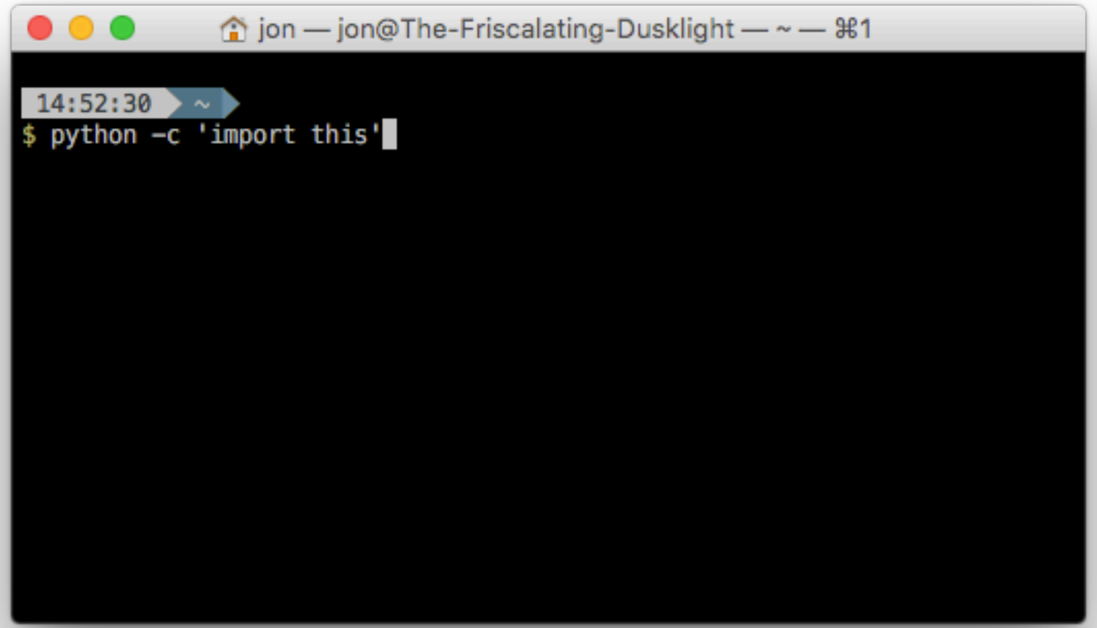
```
reduce(lambda x,y: x+y, map(lambda x: x*x, [1, 2, 3]))
```

- Want to write **imperative** code (like BASIC/C/Fortran?)

```
name = input(`Enter your name: `)
print(`Hello`, name)
```

PYTHON IS OPINIONATED

- Guido is “Benevolent Dictator For Life”
- Clear style guide: PEP-8
- The Zen of Python – baked right into the language…



```
14:52:30 ~
$ python -c 'import this'
```

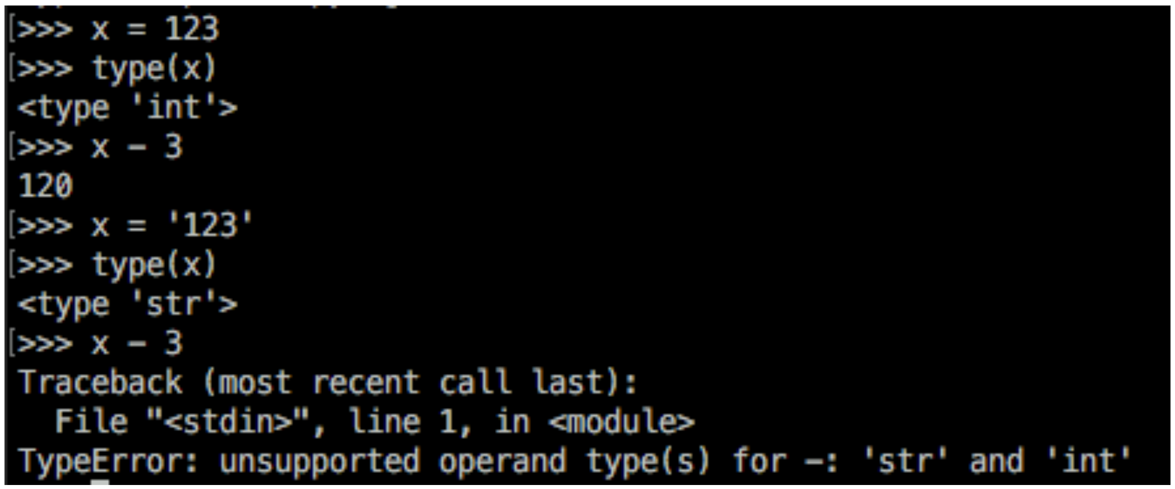
```
>>> import this

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one -- and preferably only one -- obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

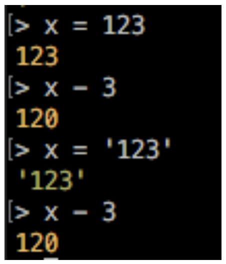
PYTHON IS DYNAMICALLY TYPED…

- Variables checked at run-time, not compilation-type
- In fact, no compilation at all!
- Interpreted language
- “Duck-typing”
- Much more flexible
- …But much easier to shoot yourself in the foot

BUT IT STILL IS TYPED (STRONGLY!)



```
>>> x = 123
>>> type(x)
<type 'int'>
>>> x - 3
120
>>> x = '123'
>>> type(x)
<type 'str'>
>>> x - 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```



```
> x = 123
123
> x - 3
120
> x = '123'
'123'
> x - 3
120
```

Contrast with weakly-typed JavaScript!

PYTHON IS CLEAN

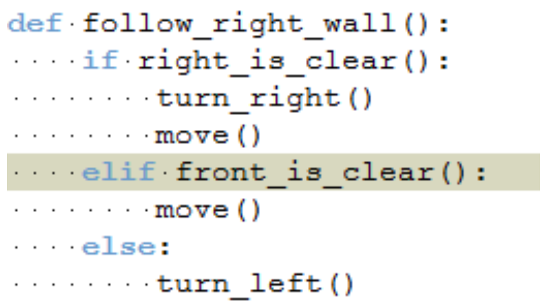
HELLO.PY

```
print(‘Hello World!’)
```

HELLO.JAVA

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println(“Hello World!”);
    }
}
```

BUT WHITESPACE MATTERS



```
def·follow_right_wall():
····if·right_is_clear():
········turn_right()
········move()
····elif·front_is_clear():
········move()
····else:
········turn_left()
```

- No line-ending semicolons;
- … But one line at a time!

- No mess of nested braces { { { } } }
- … But pay attention to your spaces!

-
  ```
  if user_input == password:
     print('Authorized.')
     log_in()
  ```

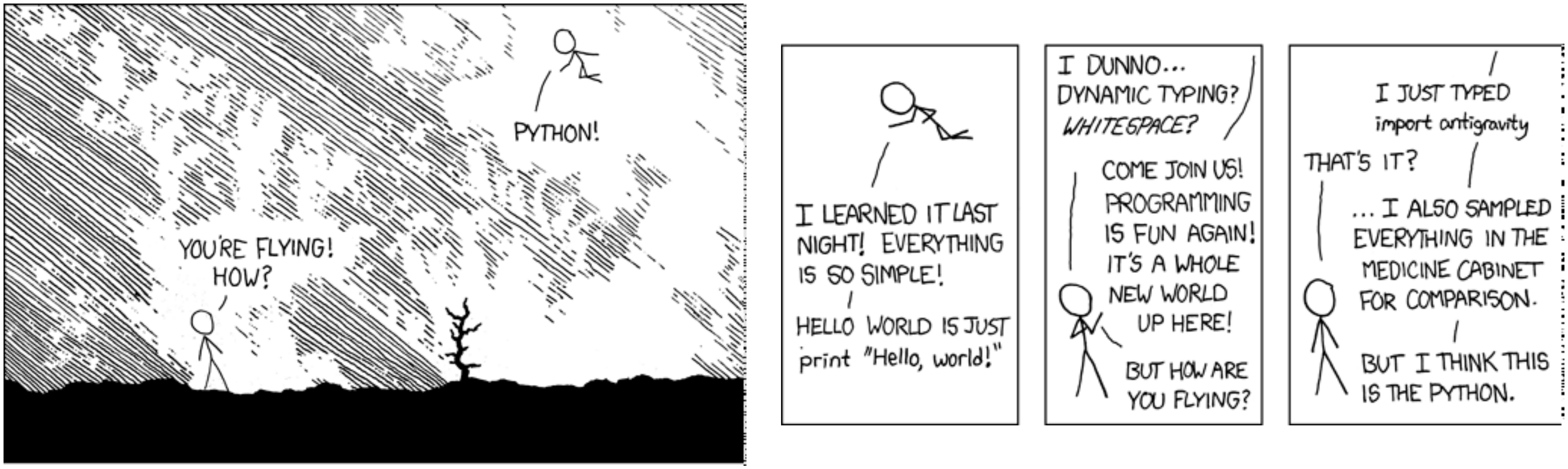
-
  ```
  if user_input == password:
     print('Authorized.')
  log_in()
  ```

PYTHON IS FULLY-LOADED

- Thorough standard library
  json, csv, re, math, datetime, logging, random…
- Large, active 3rd-party community
- Functionality organized into *modules*
- In Python: `import X` or `import Y from X`
- Outside (if not in the stdlib): `pip install X`

WHY PYTHON

FINALLY… PYTHON IS FUN!


YOU'RE FLYING! HOW?
PYTHON!
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

https://xkcd.com/353/

WHY NOT PYTHON?

- Speed is the most important factor
- Memory utilization is the most important factor
- …**Low-level systems programming**
- …**Realtime/safety-critical applications**

- *You're not allowed to ("Real enterprise developers use X")*
- *You need to look trendy ("Today it's all about X.js")*

INTRODUCTION TO PYTHON

INSTALLING AND RUNNING PYTHON

OUT OF THE BOX

- OS X: Python 2.7
- Windows: Nothing!
- Linux: Depends… likely 2.7 (but let's see!)

AVAILABLE DISTRIBUTIONS

- Homebrew (OS X only):
  http://brew.sh
  ```
  brew install python
  brew install python3
  ```

- Official installer:
  https://www.python.org/downloads/

- Anaconda:
  https://www.continuum.io/downloads

ANACONDA

- Separate from any other Python installation
- Includes iPython shell, iPython notebook server, and Spyder IDE
- Includes many popular math, science, and data science libraries
- Separate package system from “normal” Python

```
conda install
```

vs.

```
pip install
```

TEXT EDITOR

- Some people prefer to use a dedicated Python IDE
    - IDLE
    - iPython Notebook
    - JetBrains PyCharm
    - Spyder
- Most just use a favorite text editor
    - SublimeText
    - Atom
    - Notebook++ (Windows only)
- Matter of personal preference

INTRODUCTION TO PYTHON

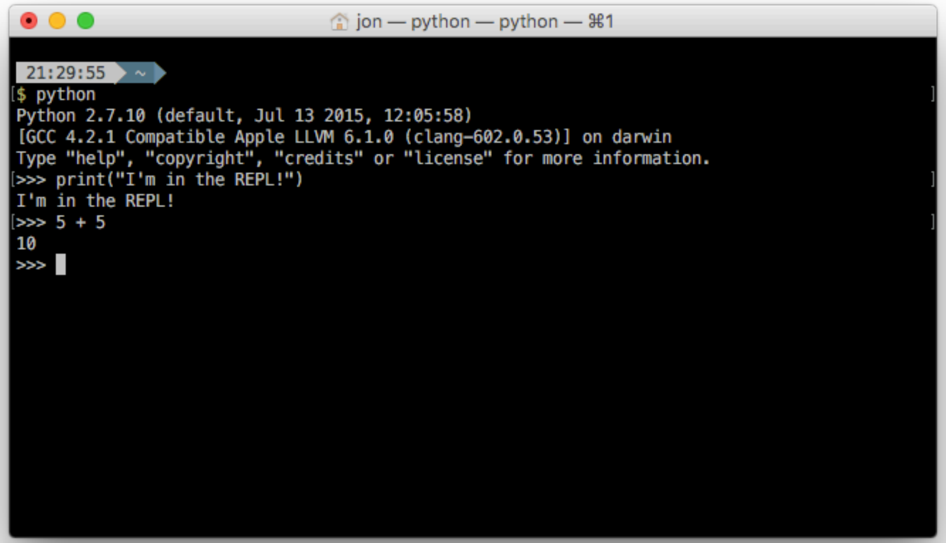
EXERCISE: SET UP YOUR ENVIRONMENT

INTRODUCTION TO PYTHON

PROGRAMMING IN PYTHON

FOLLOW ALONG!

- For most real-world purposes, the Python interpreter is run against Python (.py) files
- Python also features a REPL
- Read-Eval-Print Loop
- Great for learning!



```
 21:29:55  ~
$ python
Python 2.7.10 (default, Jul 13 2015, 12:05:58)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("I'm in the REPL!")
I'm in the REPL!
>>> 5 + 5
10
>>> 
```

BASIC MATH

STRINGS

PROGRAMMING IN PYTHON

VARIABLES

LISTS

DICTIONARIES

DICTIONARIES

PROGRAMMING IN PYTHON

CONDITIONALS

PROGRAMMING IN PYTHON

FOR LOOPS

PROGRAMMING IN PYTHON

IMPORTS

INTRODUCTION TO PYTHON

EXERCISE: WEATHER TRACKER

INTRODUCTION TO PYTHON

NEXT STEPS

NEXT STEPS

LEARNING

- Official Python Tutorial: https://docs.python.org/3/tutorial/
- Learn Python the Hard Way: http://learnpythonthehardway.org
- Codecademy: https://www.codecademy.com/tracks/python
- LOTS more!

- General Assembly!

NEXT STEPS

DATA SCIENCE

- NumPy
- Pandas
- Scikit-learn
- Statsmodels
- NLTK
- Seaborn
- Matplotlib

WEB DEVELOPMENT

- Flask
- Django
- Pyramid
- Requests
- SQLAlchemy
- Jinja2

WRAP-UP

- History of Python
- Why Python? Why not?
- Installing and Running Python
- Exercise: Set Up Your Environment
- Programming in Python: The Basics
- Exercise: Weather Tracker
- Questions & Next Steps

INTRODUCTION TO PYTHON

Q&A