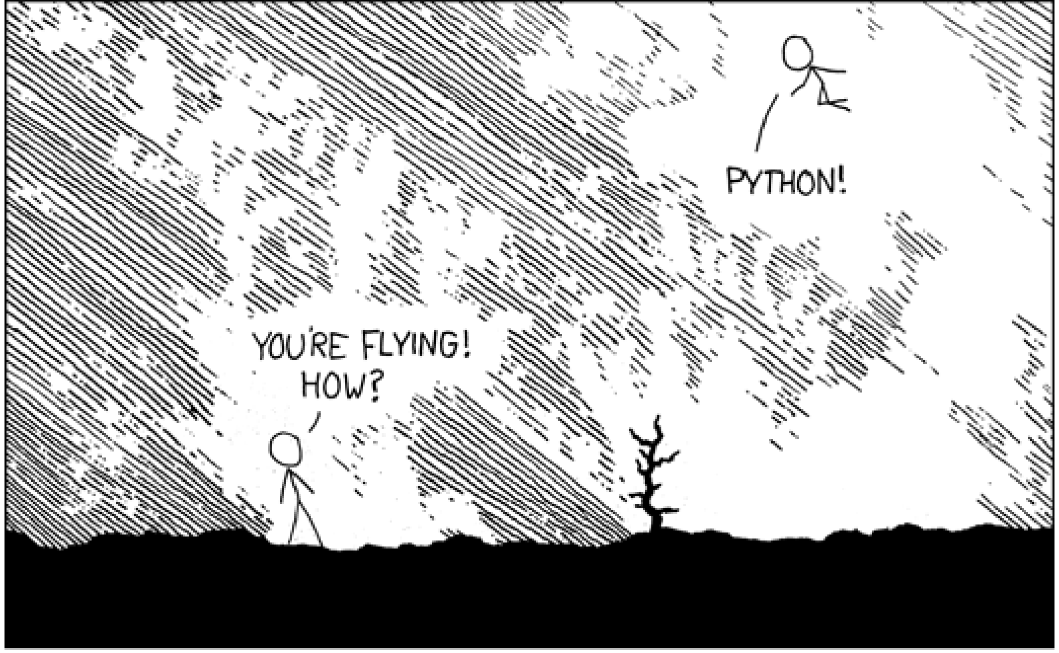
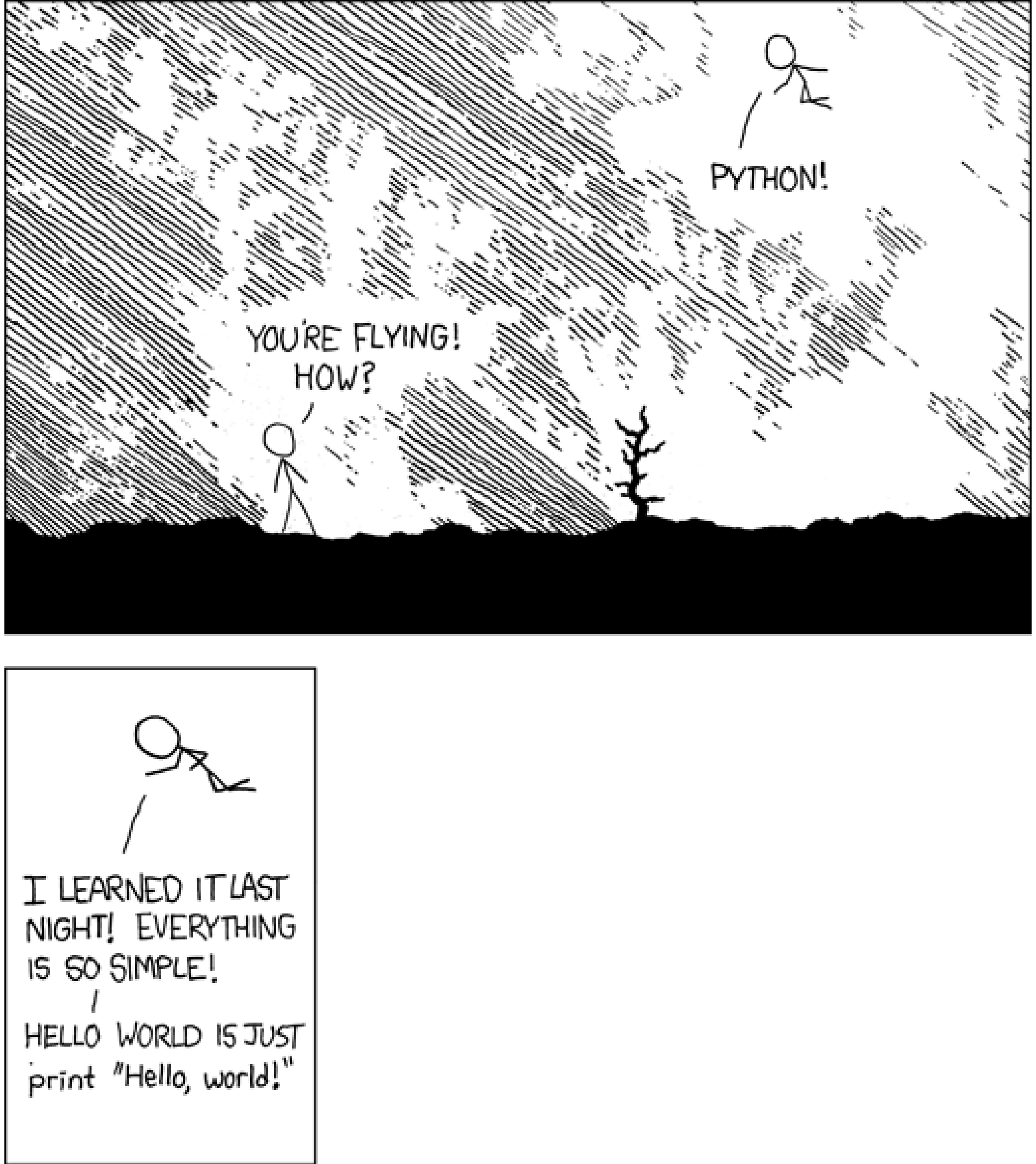
APS Workshop: **Introduction to Python**
San Francisco, CA, 24 May 2018

Python Basics

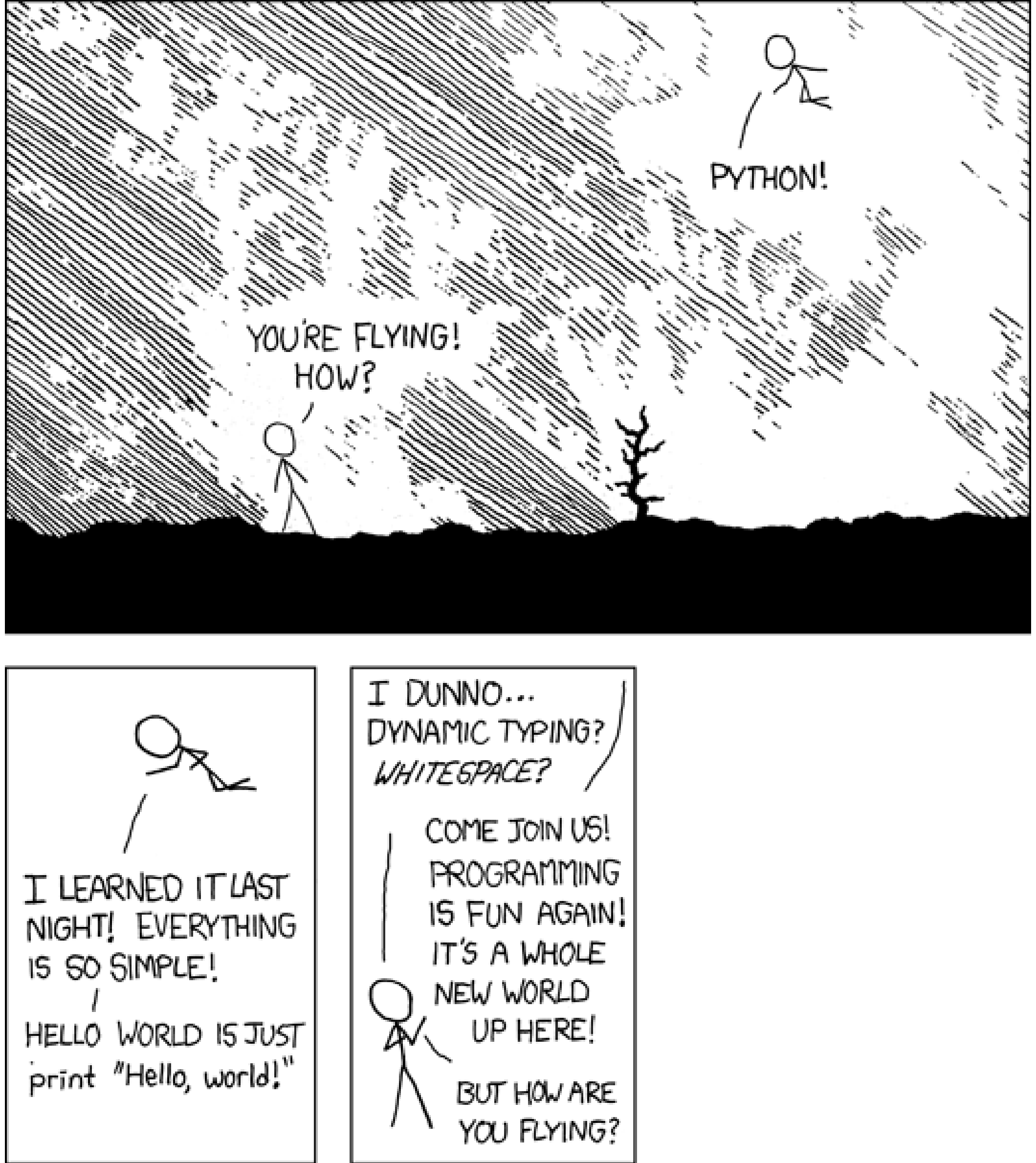
Christian C. Luhmann
Stony Brook University



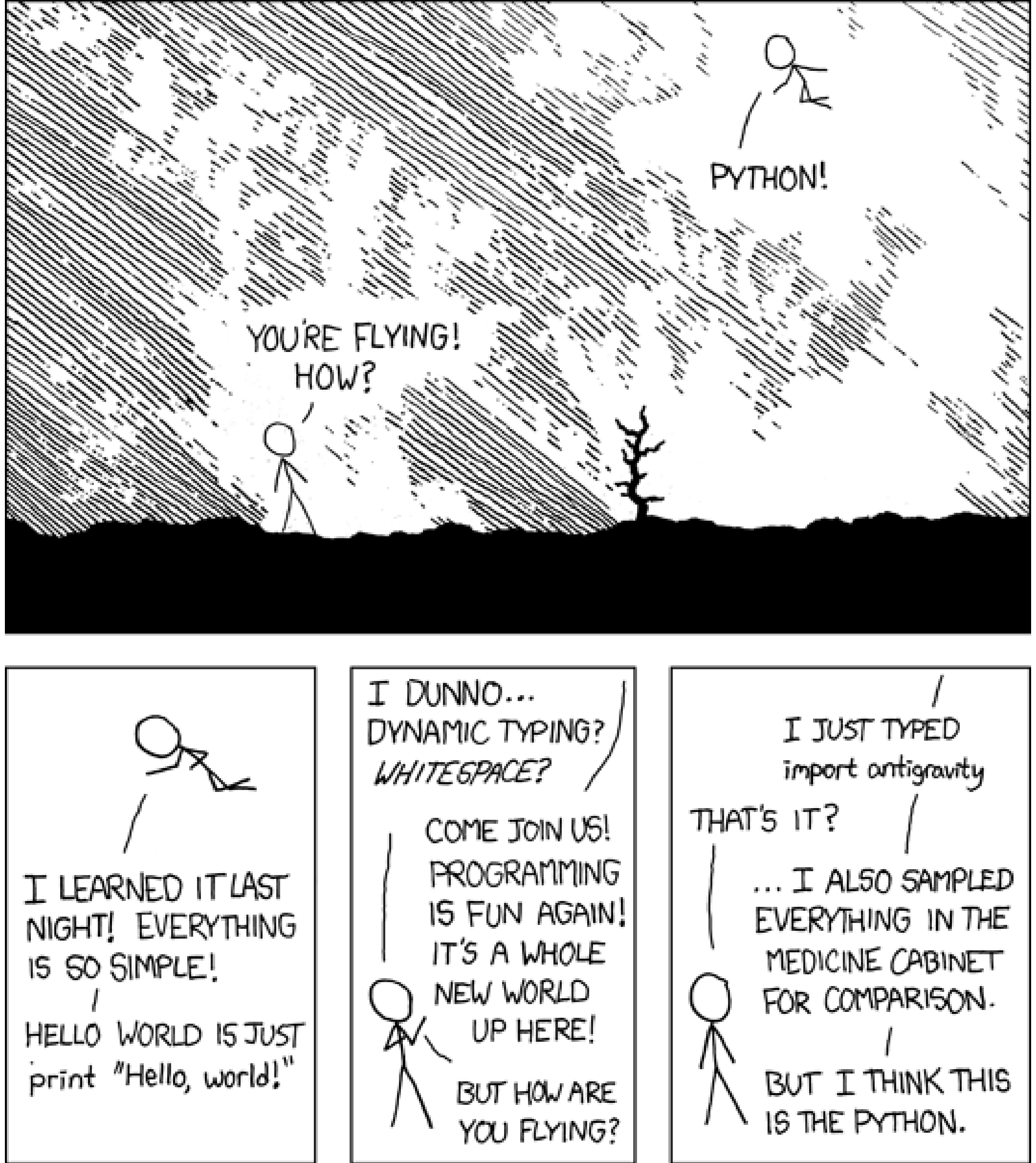
xkcd.com/353/


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"

xkcd.com/353/


PYTHON!
YOU'RE FLYING! HOW?
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?

xkcd.com/353/


PYTHON!
YOU'RE FLYING! HOW?
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.

xkcd.com/353/

```
>>> import this
The Zen of Python, by Tim Peters

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

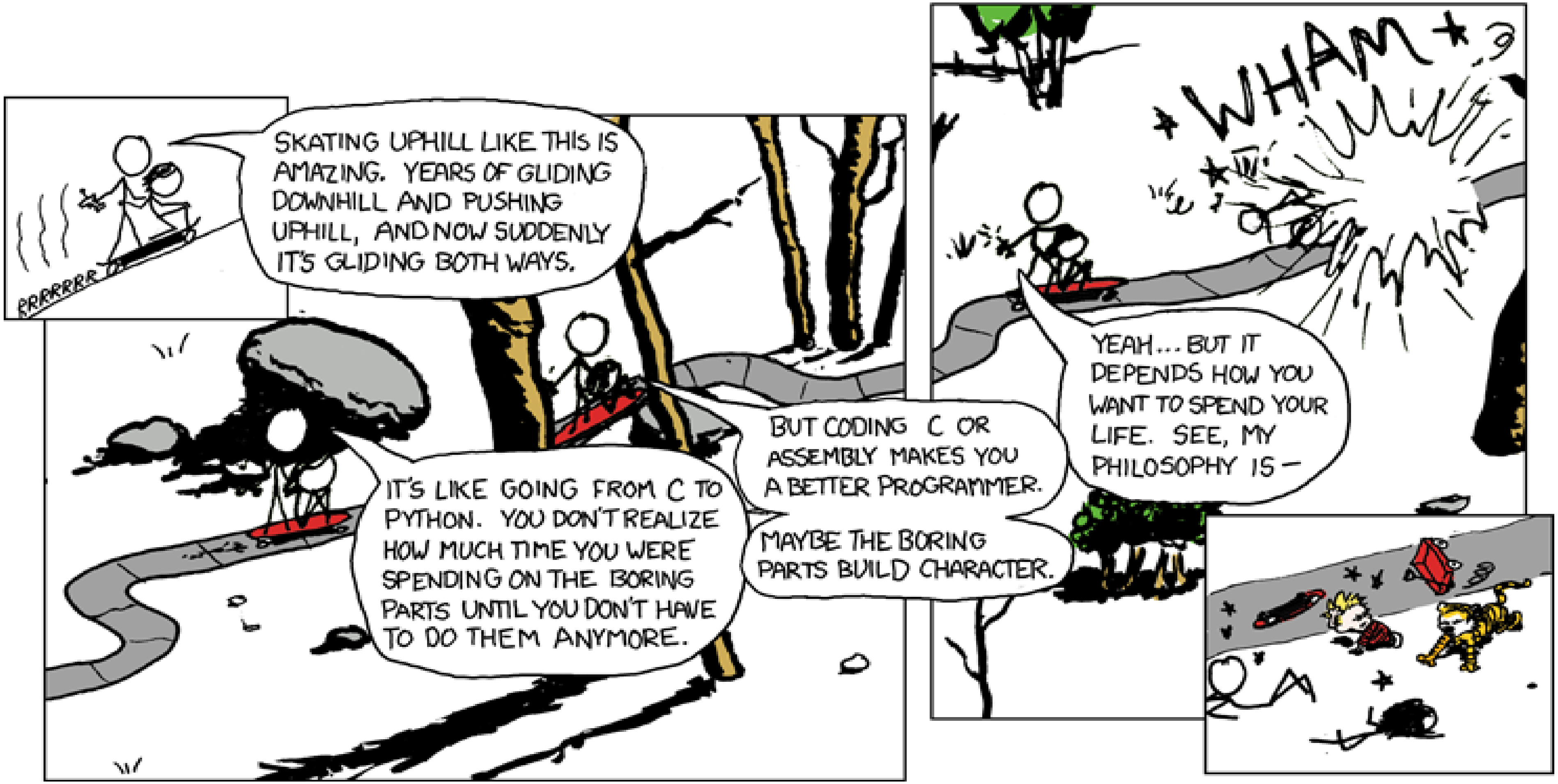
So let’s take a closer look at the language itself

Readability counts

- Every programming language represents a tradeoff between...
  - time it takes **to write** a program (your time)
  - time it takes **to run** a program (computer’s time)

- Python prioritizes **your** time

Python


PRRRRRR
SKATING UPHILL LIKE THIS IS AMAZING. YEARS OF GLIDING DOWNHILL AND PUSHING UPHILL, AND NOW SUDDENLY IT'S GLIDING BOTH WAYS.
IT'S LIKE GOING FROM C TO PYTHON. YOU DON'T REALIZE HOW MUCH TIME YOU WERE SPENDING ON THE BORING PARTS UNTIL YOU DON'T HAVE TO DO THEM ANYMORE.
BUT CODING C OR ASSEMBLY MAKES YOU A BETTER PROGRAMMER.
MAYBE THE BORING PARTS BUILD CHARACTER.
YEAH... BUT IT DEPENDS HOW YOU WANT TO SPEND YOUR LIFE. SEE, MY PHILOSOPHY IS —
WHAM

xkcd.com/409/

Python

For researchers, this means...

- students are less intimidated
- collaboration is easier
- improved transparency

Outline

1. Overview
2. Ways of using Python
3. Python basics
4. Data set overview
5. Data wrangling
6. Statistics
7. Plotting
8. Experiment creation