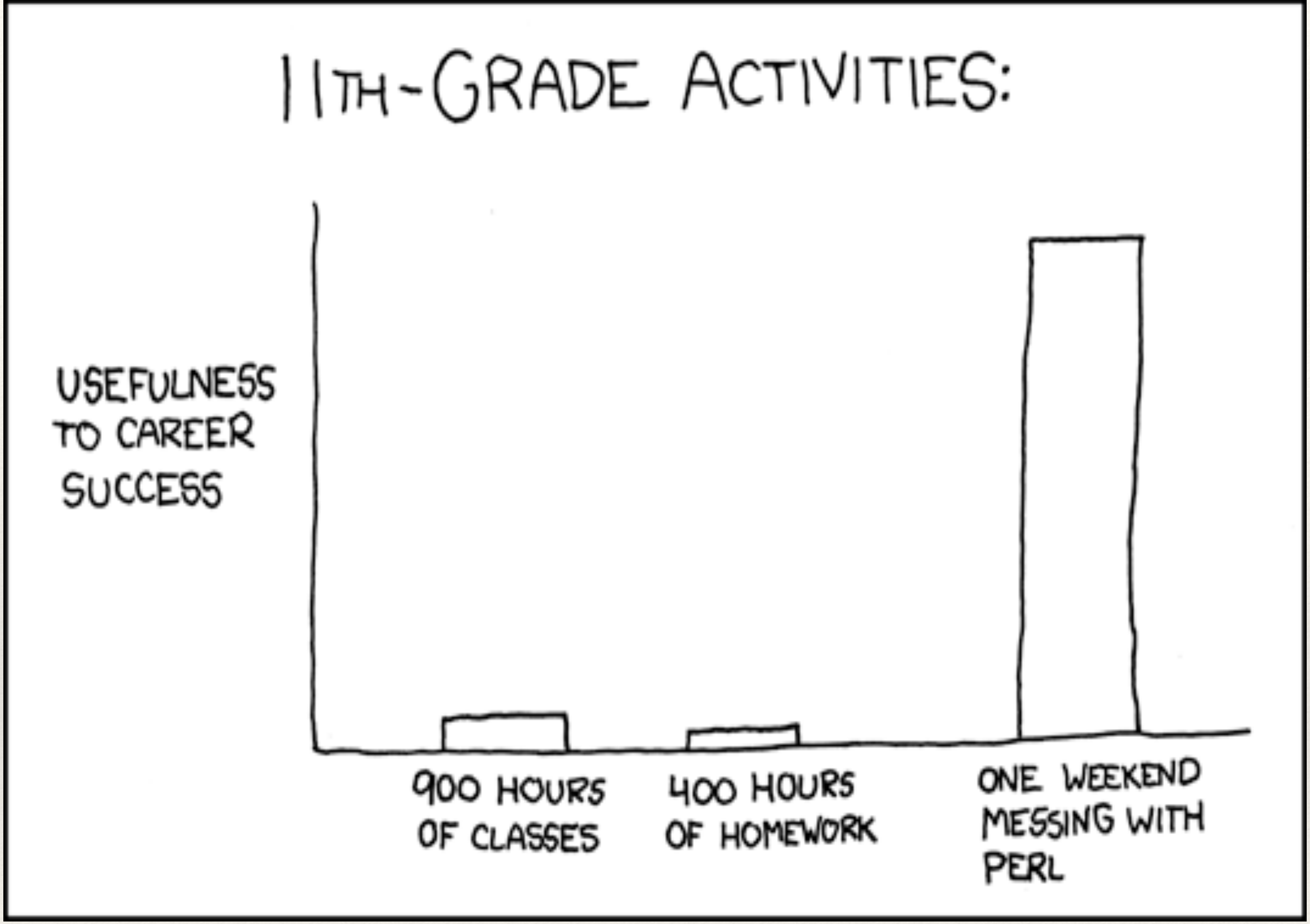
Advanced Programming Techniques

Python

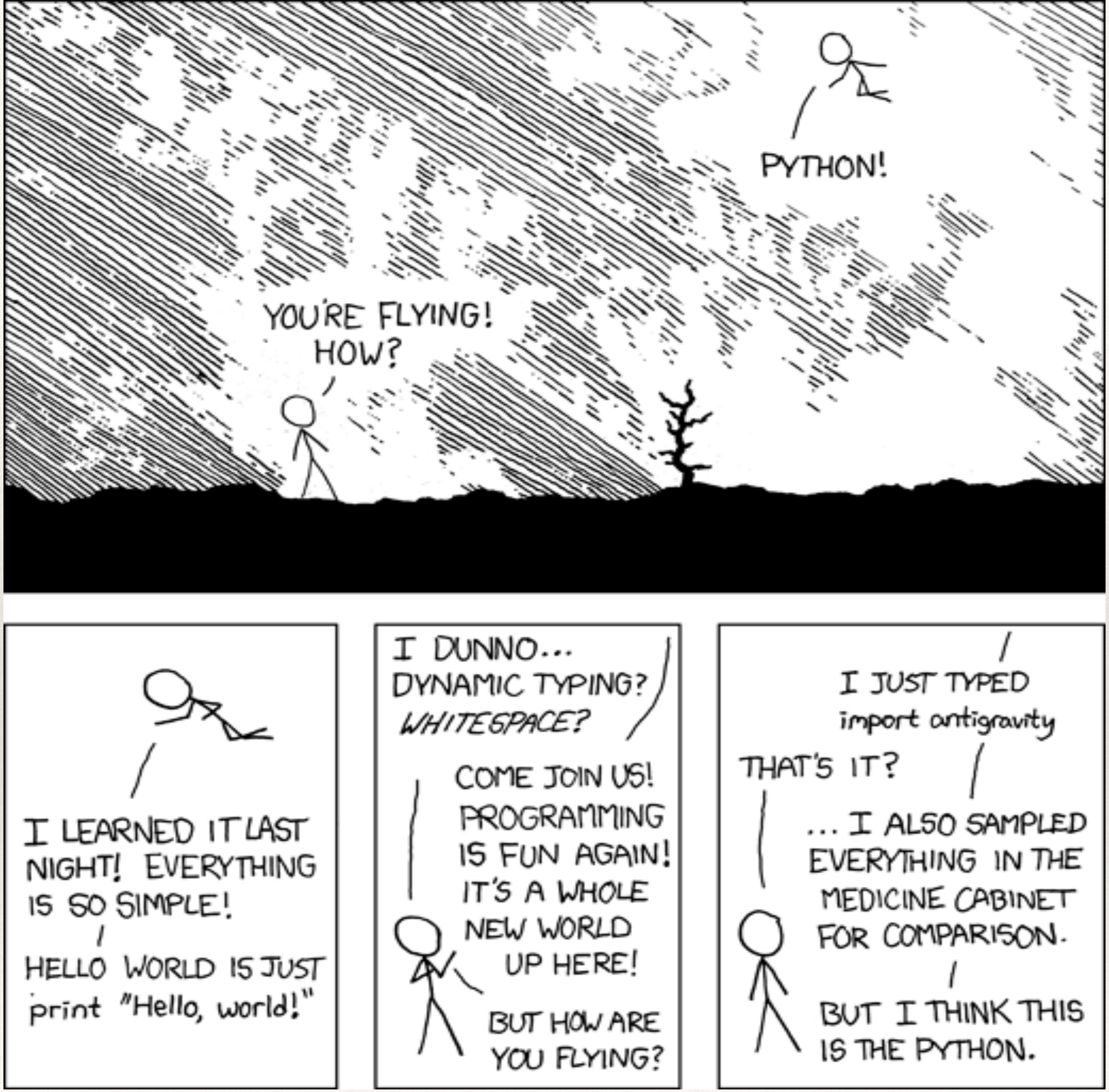
Christopher Moretti

DAVID L. WOLPER Presents
IF IT'S TUESDAY,
THIS MUST BE PYTHON
PYTHON
G
General
DVD



And the ten minutes striking up a conversation with that strange kid in homeroom sometimes matters more than every other part of high school combined.

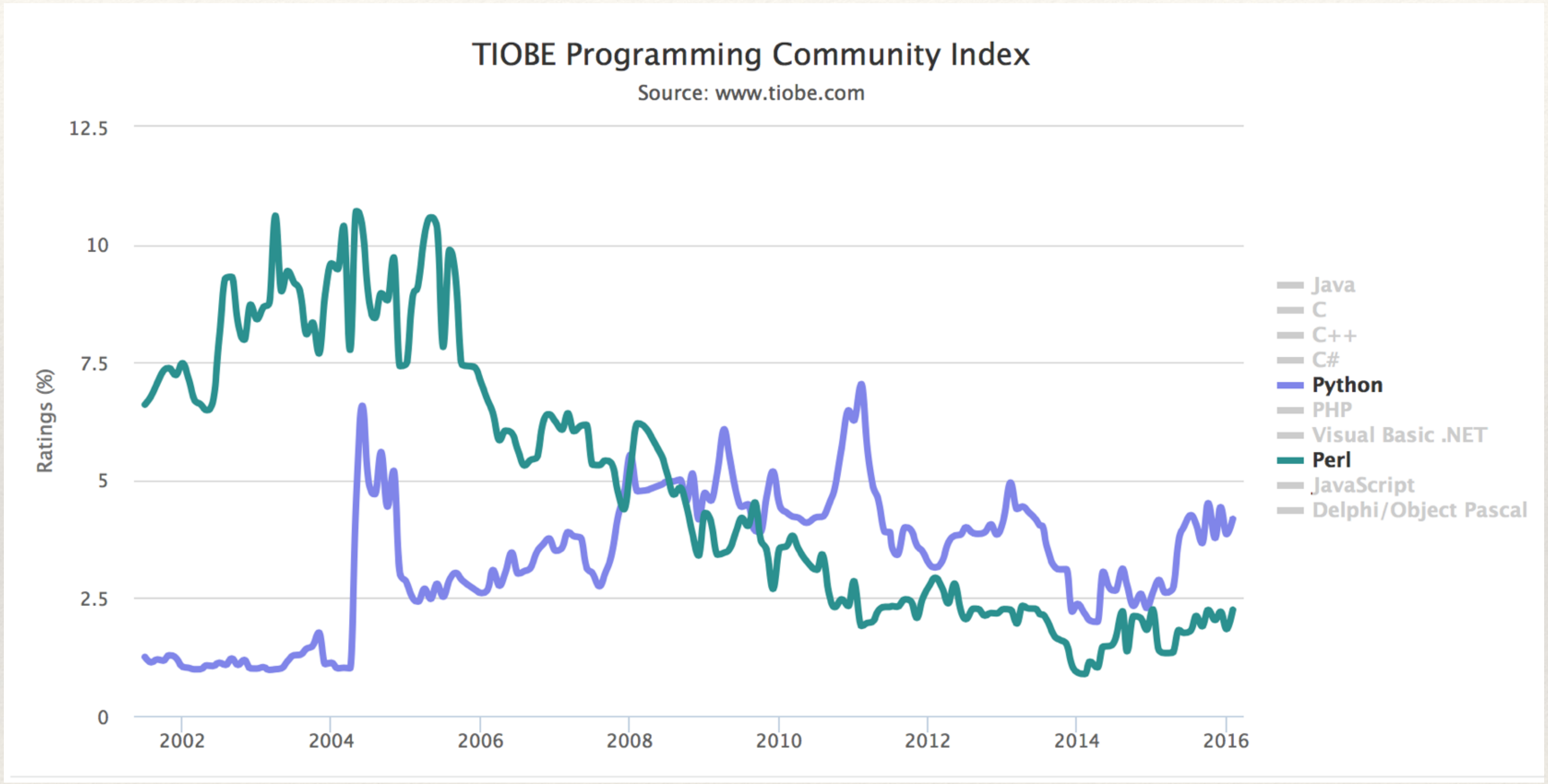
xkcd.com/519


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.

I wrote 20 short programs in Python yesterday. It was wonderful. Perl, I'm leaving you.

xkcd.com/353

Randall is not Alone



Conor Myhrvold '11 (A.B. GEO -> MIT -> Harvard -> Program Manager at Uber) wrote an interesting article on The Fall of Perl

(Article: http://bit.ly/1PBvhmq) (Image: http://www.tiobe.com)

“I was looking for a 'hobby' programming project that would keep me occupied during the week around Christmas. […] I decided to write an interpreter for the new scripting language I had been thinking about lately […] that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).”

– Guido van Rossum
Developer/BDFL of Python

Why Learn/Use Python?

- High-level, expressive, readable
- Typical of scripts: weakly-typed, interpreted, quick-start
- Atypical of scripts: reasonable class and OOP structures
- Rich libraries, good documentation, active development

- Perhaps the best option to close the gap between AWK/shells and C/Java/etc.?

Simple Python Constructs

```
print ‘Hello, world!’                      print “Hello, world!”

print ‘Two newlines\n’  print ‘One newline\n’,  print r“No newline\n”,
```

```
>>> i='4.'      >>> i=4         >>> i=4.
>>> i           >>> i           >>> i
'4.'            4               4.0
>>> float(i)    >>> float(i)
4.0             4.0
                                >>> int(i)
                                4
                >>> str(i)      >>> str(i)
                ‘4'             ‘4.0'

>>> bool(i)     >>> bool(i)     >>>bool(i)
True            True            True
                >>> i=81
                >>> chr(i)
                'Q'
```

Operators

- Generally like C:
  - `+ - * / % & | ^ ~ << >> < > <= >= == !=`
- Some differences:
  - `and or not`
  - comparison chaining is legal: `3 < x < 5`
  - no `++ -- ?:`
- Some additional operators:
  - `** // + in is`

Control Flow

```
if bool:
    statement
elif bool’:
    statement’
else:
    statement’’
```

```
while bool:
    statement1
    statement2
```

```
for i in range(m,n):
    if bool:
        continue
    else
        statements
```

```
try:
    statement
except (ThisExn,ThatExn):
    handle_this_or_that
except AnotherExn:
    handle_another
else:
    run_if_no_exception
finally:
    unconditionally_after
```

```
while True:
    if bool:
        break
    else
        stmts
```

```
for v in c:
    statement1
    statement2
```

Regular Expressions

`re.search(re, str)`	find first match of re in str
`re.match(re, str)`	test for anchored match
`re.split(re, str)`	split str into a list of matches around re
`re.findall(re, str)`	list of all matches of re in str
`re.sub(re, rpl, str)`	replace all re in str with rpl
`\d \D \w \W \s \S`	digit non-digit word non-word space non-space

Warning!

Patterns are not necessarily matched leftmost-longest
Replacements are global by default

```
>>> re.sub('in|inch', "X", s)
Xches and Xches X Xdia and Xdonesia
>>> re.sub('inch|in', "X", s)
Xes and Xes X Xdia and Xdonesia
```

Tuples

```
>>> ()
()
>>> (1,)
(1,)
>>> (1,2)
(1, 2)
>>> x = () + (1,) + ((True, "false"),2.7)
>>> x
(1, (True, 'false'), 2.7)
>>> len(x)
3
>>> len(x[1])
2
>>> y = (21, 63, 46, -2, float('nan'))
>>> y
(21, 63, 46, -2, nan)
>>> max(y)
63
>>> min(y)
-2
>>> z = (3,)*3
>>> z
(3, 3, 3)
```

```
>>> x=(True,False,(2.0,2))
>>> t,f,ttp = x
>>> t
True
>>> f
False
>>> ttp
(2.0, 2)
>>> t,f,(tf,ti) = x
>>> t
True
>>> f
False
>>> tf
2.0
>>> ti
2
>>> b=2
>>> c=4
>>> b,c=c,b
>>> b
4
>>> c
2
```

Lists

```
>>> l = []
>>> l.append("food")
>>> l
['food']
>>> l.append(["water","booze"])
>>> l
['food', ['water', ‘booze']]
>>> l+=["candy"]
>>> l
['food', ['water', 'booze'], ‘candy’]
>>> len(l)
3
>>> len(l[1])
2
>>> l[1]=['drinks']
>>> l
['food', ['drinks'], 'candy']
>>> l[1]='drinks'
>>> l
['food', 'drinks', 'candy']
>>> del l[1]
>>> l
['food', ‘candy']
```

```
>>> for i in l:
...     print i
...
food
candy
>>> max(l)
‘food'
>>> 'beer' in l
False
>>> x=[]
>>> for i in range(0,10): x.append(i)
...
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x[1:]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x[1:3]
[1, 2]
>>> x[-2]
8
>>> x[-2:]
[8, 9]
>>> x[3:-2]
[3, 4, 5, 6, 7]
```

List Comprehensions

```
>>> x=[]
>>> for i in range(0,10): x.append(i)
...
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> x= [i for i in range(10)]
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> pow2 = [2**i for i in range(10)]
>>> pow2
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

Dictionaries

```
>>> words = {'call': 1, 'me': 1, 'ishmael': 1}
>>> words['whale'] = 1
>>> words['i']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'i'
>>> words.get('i',0)
0
>>> words.setdefault('i',0)
0
>>> words.setdefault('whale',0)
2
>>> words
{'me': 1, 'i': 0, 'whale': 2, 'call': 1, 'ishmael': 1}
>>> words['whale'] = words['whale'] + 1
>>> words
{'me': 1, 'whale': 2, 'call': 1, 'ishmael': 1}
>>> len(words)
5
>>> max(words)
'whale'
>>> words.keys()
['me', 'i', 'whale', 'call', 'ishmael']
>>> words.items()
[('me', 1), ('i', 0), ('whale', 2), ('call', 1), ('ishmael', 1)]
>>> dict([('whale',2),('ishmael',1)])
{'whale': 2, 'ishmael': 1}
```

Functions

```
def name(arg, arg, arg):
    statements of function


def div(num, denom):
    ''' computes quotient & remainder. denom should be > 0'''
    q = num / denom
    r = num % denom
    return (q, r)  # returns a list
```

Functions are first-class. You can assign them, pass them to functions, return them from functions

Parameters are passed call by value. You can can have named arguments, default values, and arrays of name-value argument pairs. This is a bit tricky — use caution, use your references!

Variables in functions are local unless declared global, but globals are visible for reading.

Classes and objects

```
class Stack:
   def __init__(self):  # constructor
      self.stack = []   # instance variable
   def push(self, obj):
      self.stack.append(obj)
   def pop(self):
      return self.stack.pop()    # list.pop
   def len(self):
      return len(self.stack)

stk = Stack()
stk.push("foo")
if stk.len() != 1: print "error"
if stk.pop() != "foo": print "error"
del stk
```

Python Easter Eggs

```
>>> import __hello__
Hello world ...
```

Python Easter Eggs

```
>>> from __future__ import braces
  File "<stdin>", line 1
SyntaxError: not a chance
```

Python Easter Eggs

```
>>> 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!
```

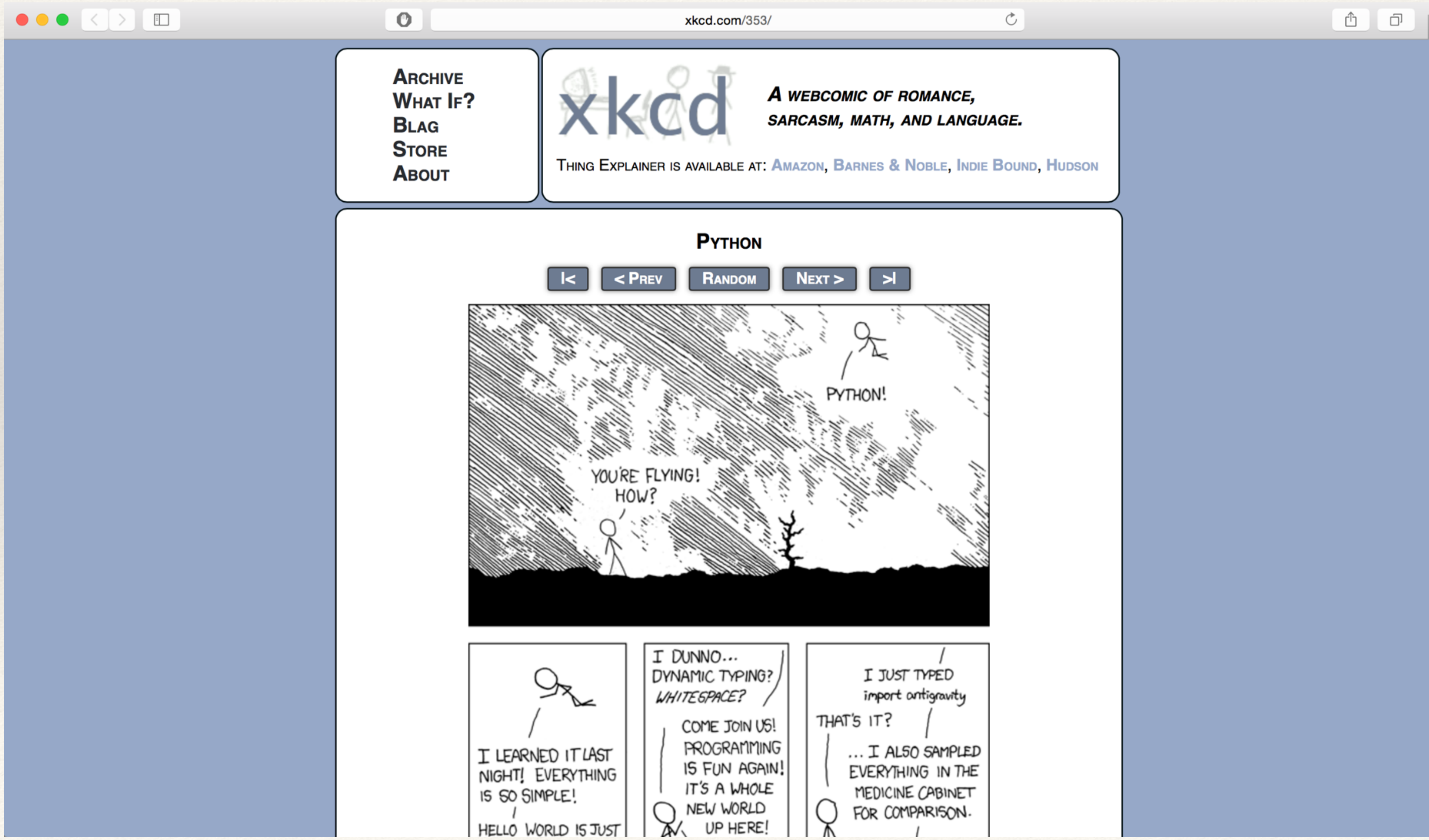
Python Easter Eggs

```
>>> print this.s
Gur Mra bs Clguba, ol Gvz Crgref

Ornhgvshy vf orggre guna htyl.
Rkcyvpvg vf orggre guna vzcyvpvg.
Fvzcyr vf orggre guna pbzcyrk.
Pbzcyrk vf orggre guna pbzcyvpngrq.
Syng vf orggre guna arfgrq.
Fcnefr vf orggre guna qrafr.
Ernqnovyvgl pbhagf.
Fcrpvny pnfrf nera'g fcrpvny rabhtu gb oernx gur ehyrf.
Nygubhtu cenpgvpnyvgl orngf chevgl.
Reebef fubhyq arire cnff fvyragyl.
Hayrff rkcyvpvgyl fvyraprq.
Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff.
Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg.
Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu.
Abj vf orggre guna arire.
Nygubhtu arire vf bsgra orggre guna *evtug* abj.
Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn.
Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn.
Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!
```

Python Easter Eggs

```
>>> import antigravity
```


xkcd.com/353/

A WEBCOMIC OF ROMANCE, SARCASM, MATH, AND LANGUAGE.

THING EXPLAINER IS AVAILABLE AT: AMAZON, BARNES & NOBLE, INDIE BOUND, HUDSON

ARCHIVE
WHAT IF?
BLAG
STORE
ABOUT
xkcd
PYTHON
|<
< PREV
RANDOM
NEXT >
>|
YOU'RE FLYING! HOW?
PYTHON!
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.

Python practice, problem solving with code, etc.

www.pythonchallenge.com

NB: don't confuse with www.pythonchallenge.org