python™

Why You'll Love Python

Shawn Milochik
Shawn@Milochik.com
@ShawnMilo

Slides as PDF: http://bit.ly/zcDRO

Named after Monty Python

BBC records
Monty Python's
Flying Circus
With John Cleese, Michael Palin, Grahame Chapman, Eric Idle, Terry Jones, Carol Cleveland
from the BBC Television Series

I rest my case.

Argument from Authority*

"Perl still has its uses. For tiny projects (100 lines or fewer) that involve a lot of text pattern matching... "

"For anything larger or more complex, I have come to prefer the subtle virtues of Python—and I think you will, too."

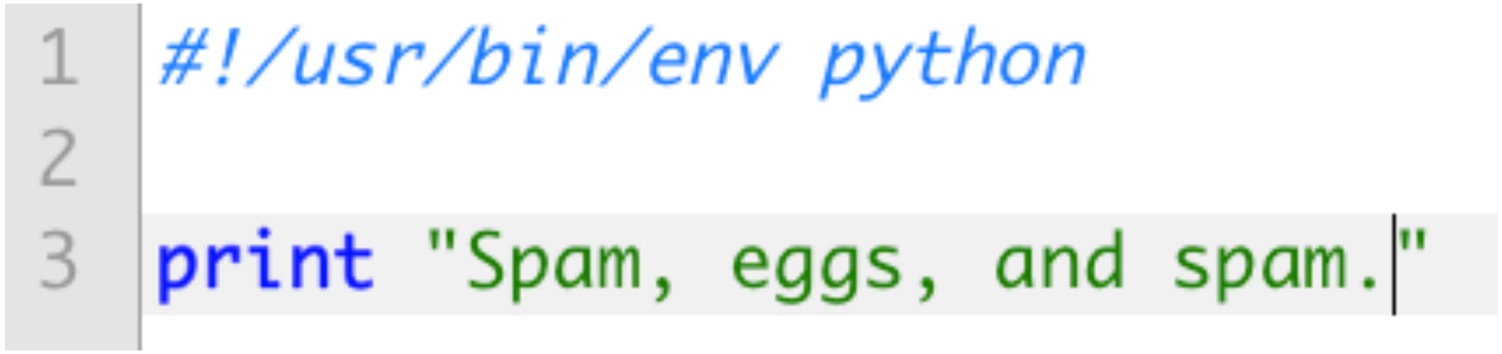
--Eric S. Raymond

http://www.linuxjournal.com/article/3882

*logical fallacy, but ESR *is* a programming expert

Why it's awesome to program in Python

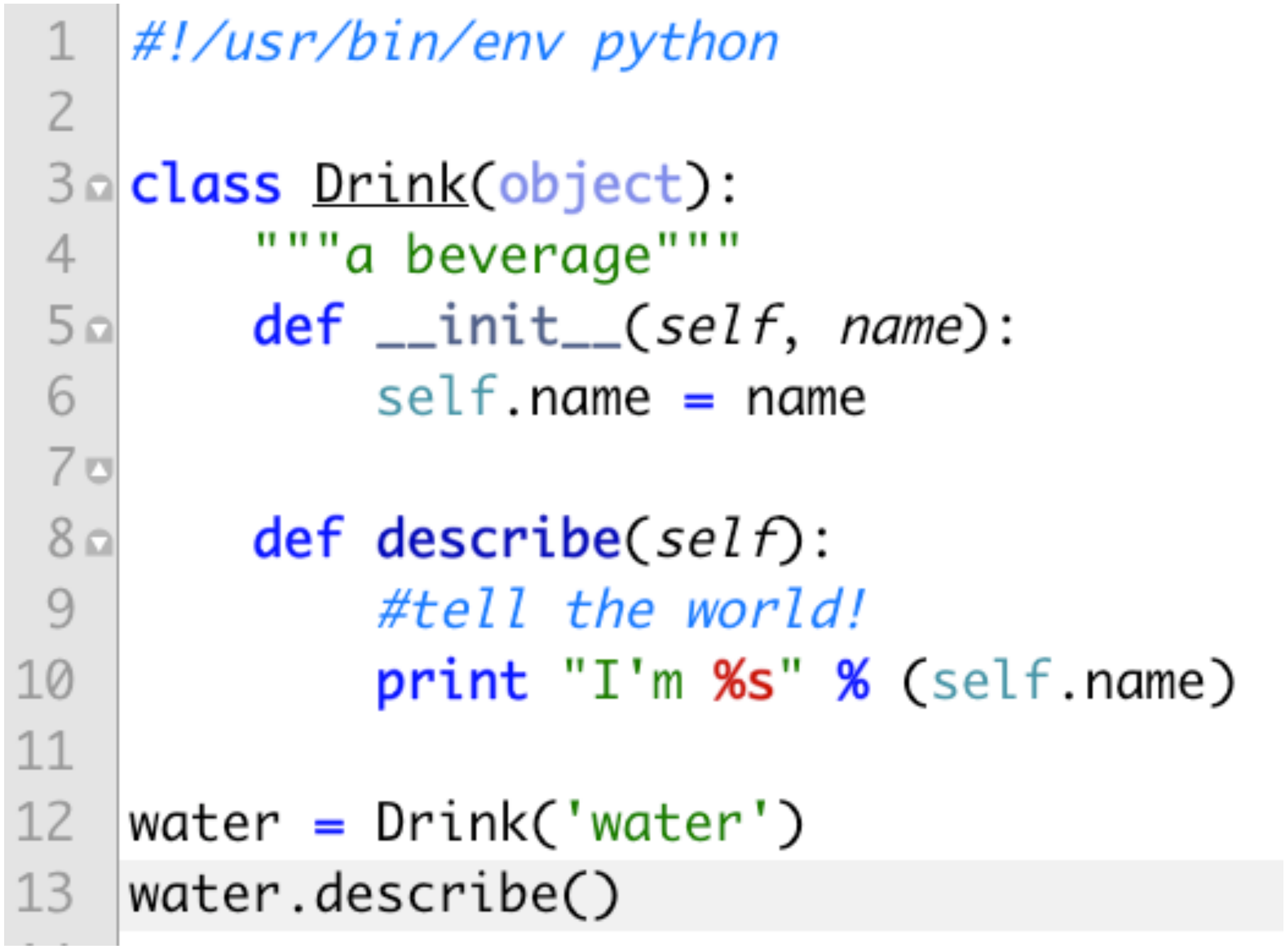
Python is non-verbose.



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

print "Spam, eggs, and spam."
```

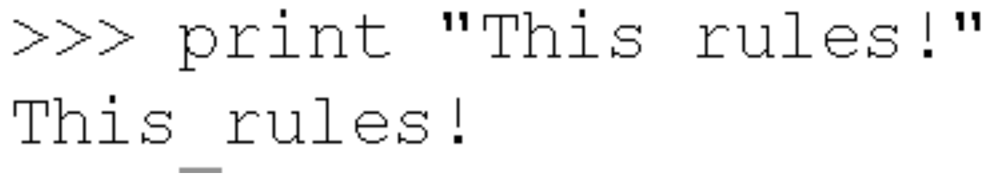
(Try that in Java.)

OOP is *easy*.



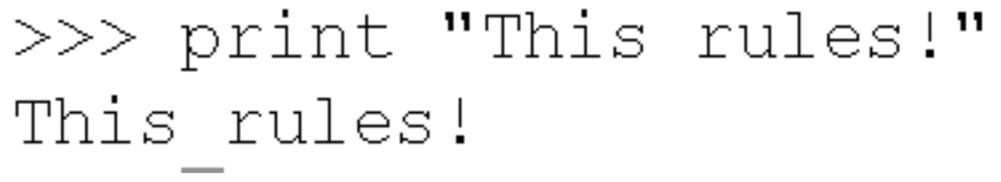
```python
#!/usr/bin/env python

class Drink(object):
    """a beverage"""
    def __init__(self, name):
        self.name = name

    def describe(self):
        #tell the world!
        print "I'm %s" % (self.name)

water = Drink('water')
water.describe()
```

Interactive Interpreter

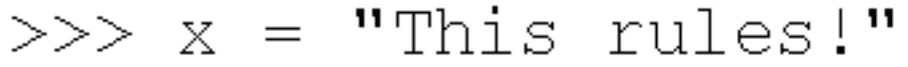


```
>>> print "This rules!"
This_rules!
```

Interactive Interpreter



```
>>> print "This rules!"
This rules!
_
```



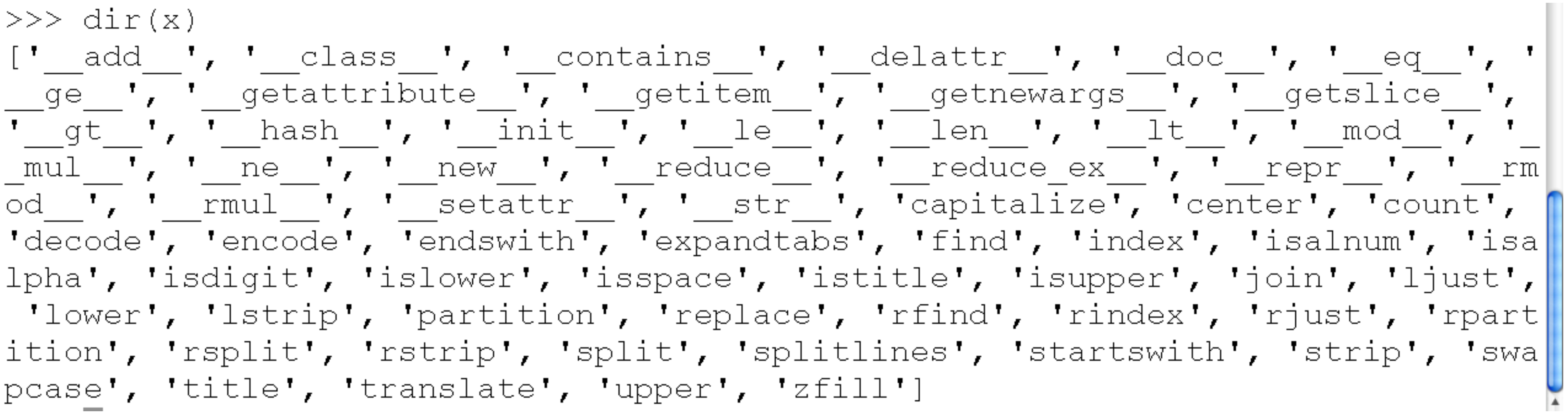
```
>>> x = "This rules!"
```

Interactive Interpreter

```
>>> print "This rules!"
This rules!
```

```
>>> x = "This rules!"
```

```
>>> dir(x)
```


```
>>> dir(x)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '
__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__',
'__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '_
_mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rm
od__', '__rmul__', '__setattr__', '__str__', 'capitalize', 'center', 'count',
'decode', 'encode', 'endswith', 'expandtabs', 'find', 'index', 'isalnum', 'isa
lpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust',
 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpart
ition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swa
pcase', 'title', 'translate', 'upper', 'zfill']
```

Fun with “x”

```
>>> x.capitalize()
'This rules!'
>>> x.lower()
'this rules!'
>>> x.isalpha()
False
>>> x.isdigit()
False
>>> x[5]
'r'
>>> x[:5]
'This '
```

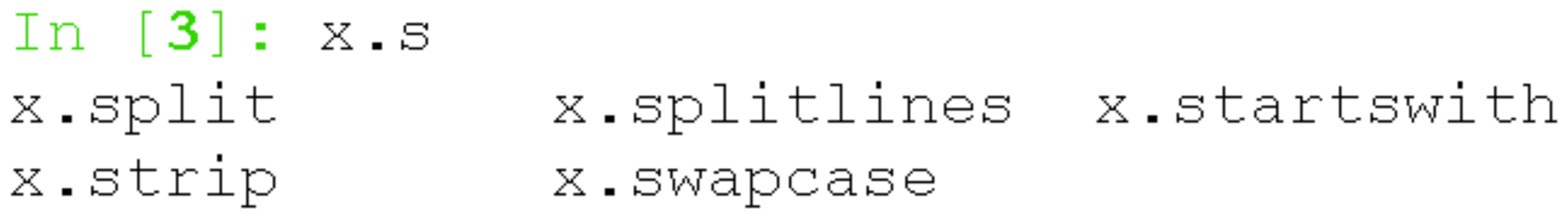
Helpful

```
>>> help(x.upper)
```

```
Help on built-in function upper:

upper(...)
    S.upper() -> string

    Return a copy of the string S converted
to uppercase.
```

```
>>> x.upper()
'THIS RULES!'
```

iPython



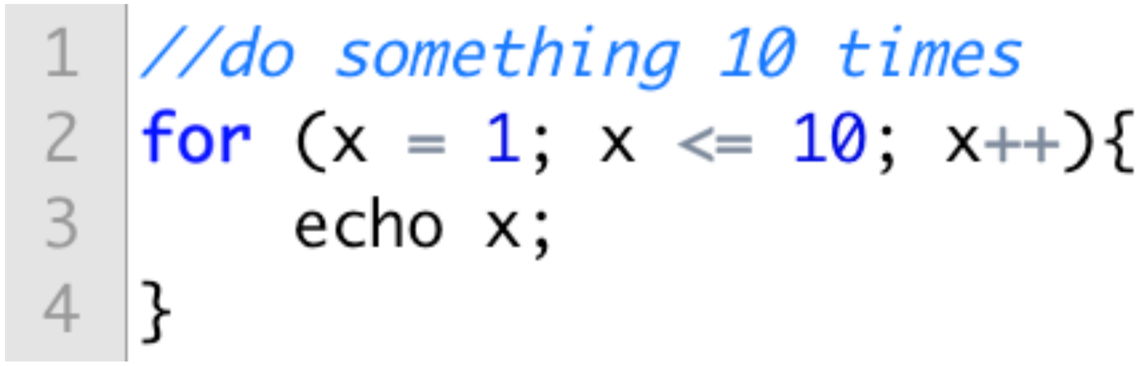
```
In [3]: x.s
x.split           x.splitlines  x.startswith
x.strip           x.swapcase
```

Great Documentation

- http://docs.python.org/

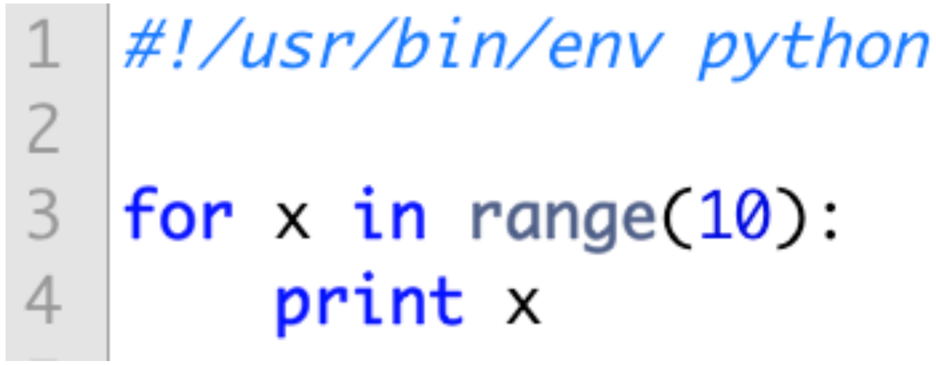
Common Tasks Made Easy

Ugly



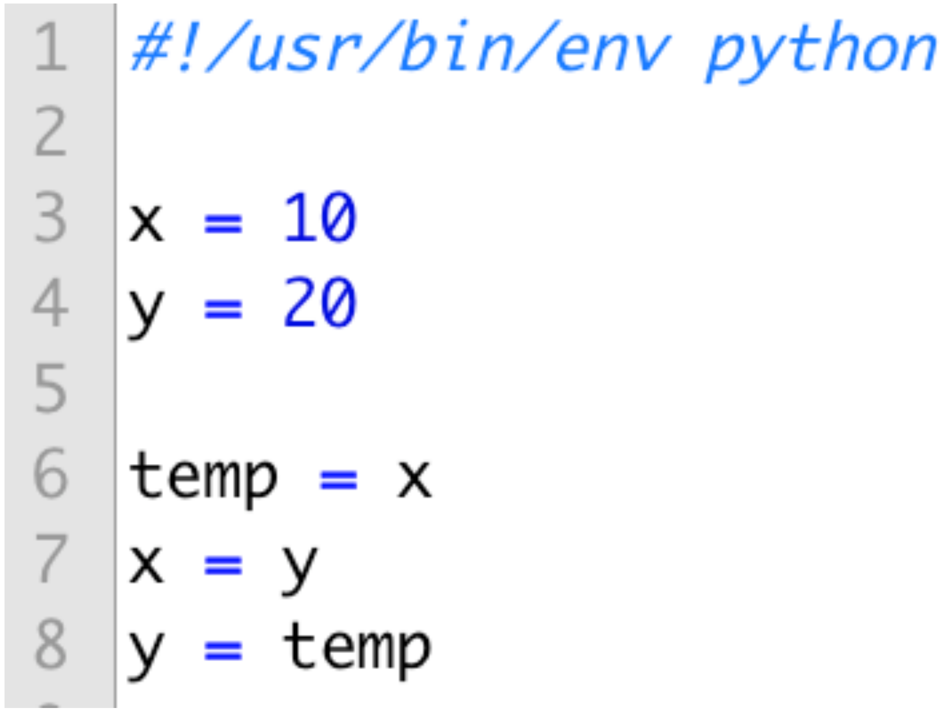
```
//do something 10 times
for (x = 1; x <= 10; x++){
    echo x;
}
```

Pretty



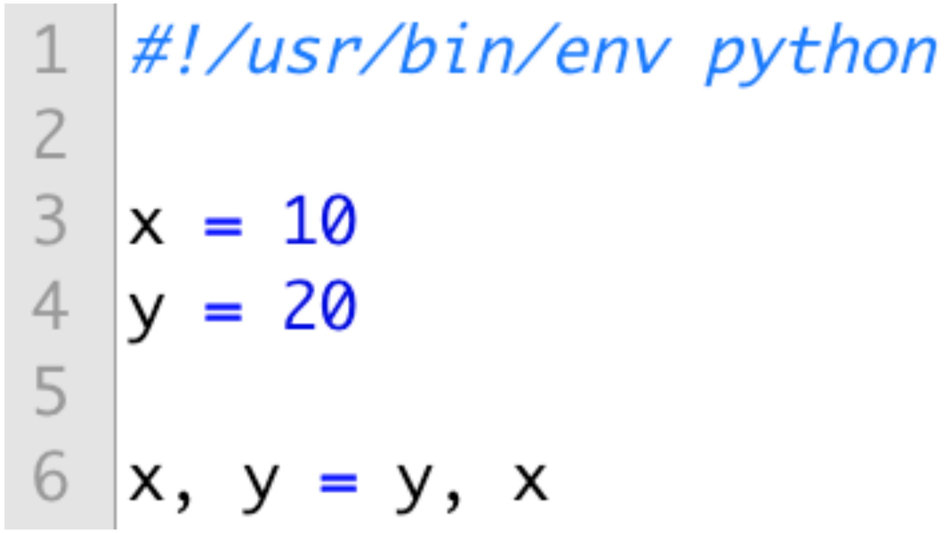
```
#!/usr/bin/env python

for x in range(10):
    print x
```

Ugly



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

x = 10
y = 20

temp = x
x = y
y = temp
```

Pretty



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

x = 10
y = 20

x, y = y, x
```

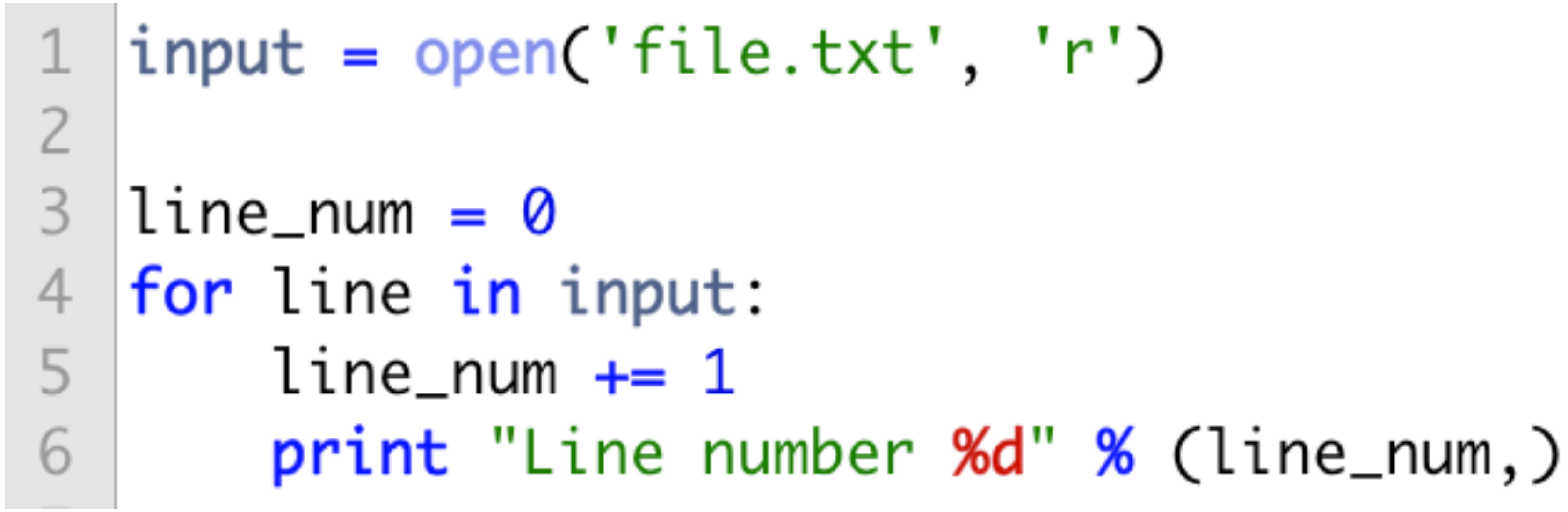
Ugly

```python
numbers = range(10)
doubles = []

for x in numbers:
    doubles.append(x * 2)
```

Pretty

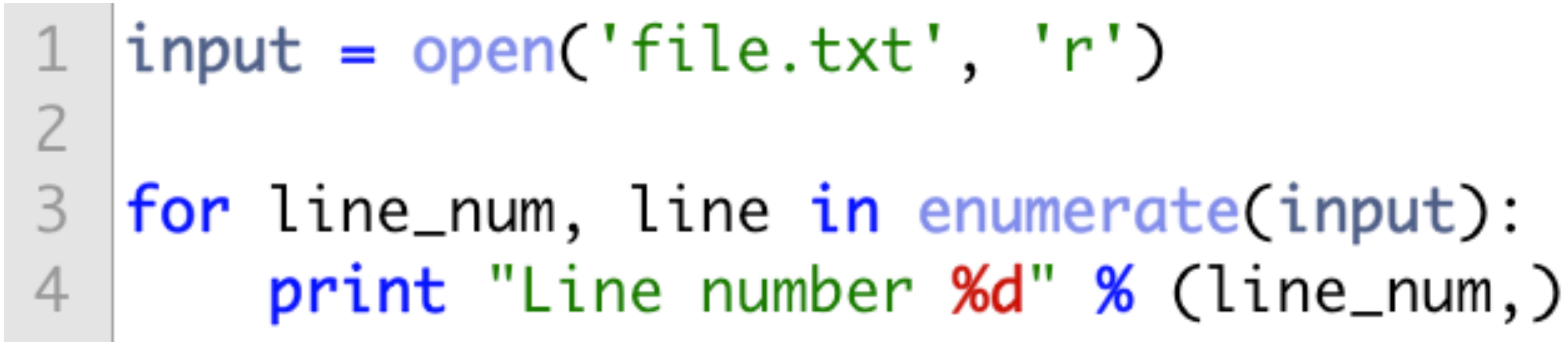
```python
numbers = range(10)

doubles = [x * 2 for x in numbers]
```

Ugly



```python
input = open('file.txt', 'r')

line_num = 0
for line in input:
    line_num += 1
    print "Line number %d" % (line_num,)
```

Pretty



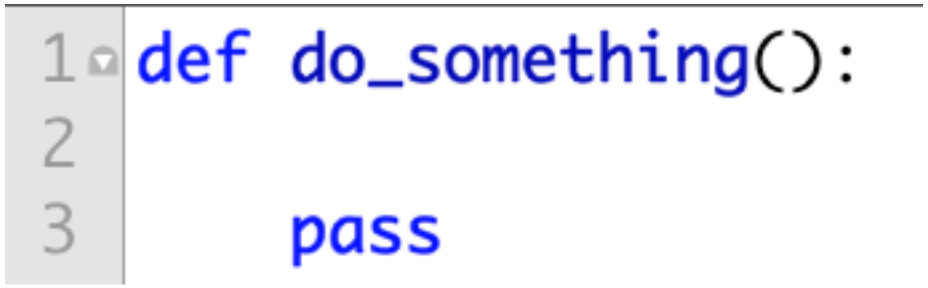
```python
input = open('file.txt', 'r')

for line_num, line in enumerate(input):
    print "Line number %d" % (line_num,)
```

```
>>> numbers = range(10)
>>> numbers
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
>>> doubles = [x * 2 for x in numbers]
>>> doubles
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>>
>>> evens = [x for x in numbers if x % 2 == 0]
>>> evens
[0, 2, 4, 6, 8]
```

Functions

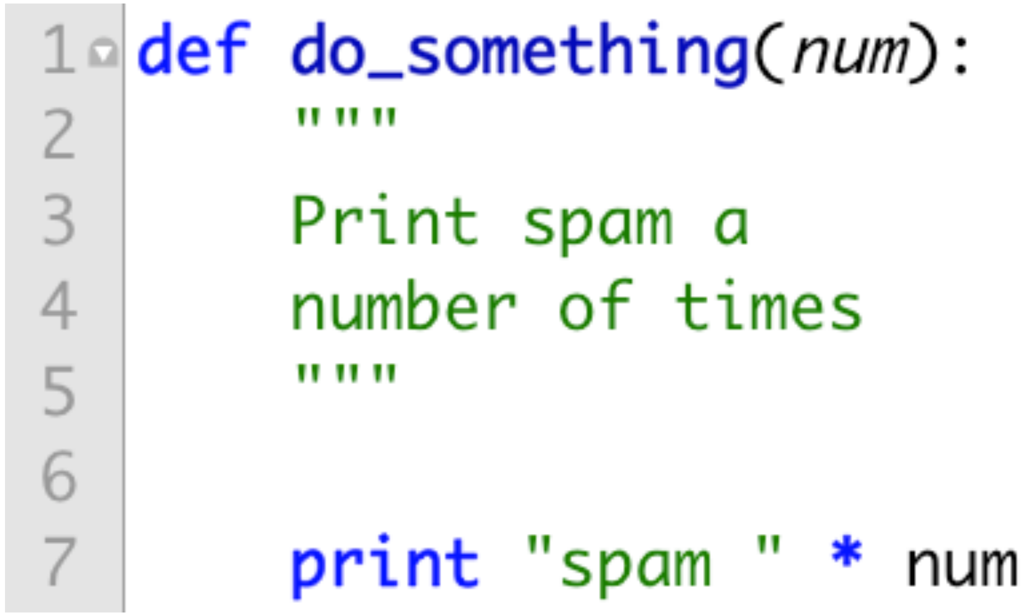
http://www.city-data.com/picfilesc/picc26126.php

Simple Syntax



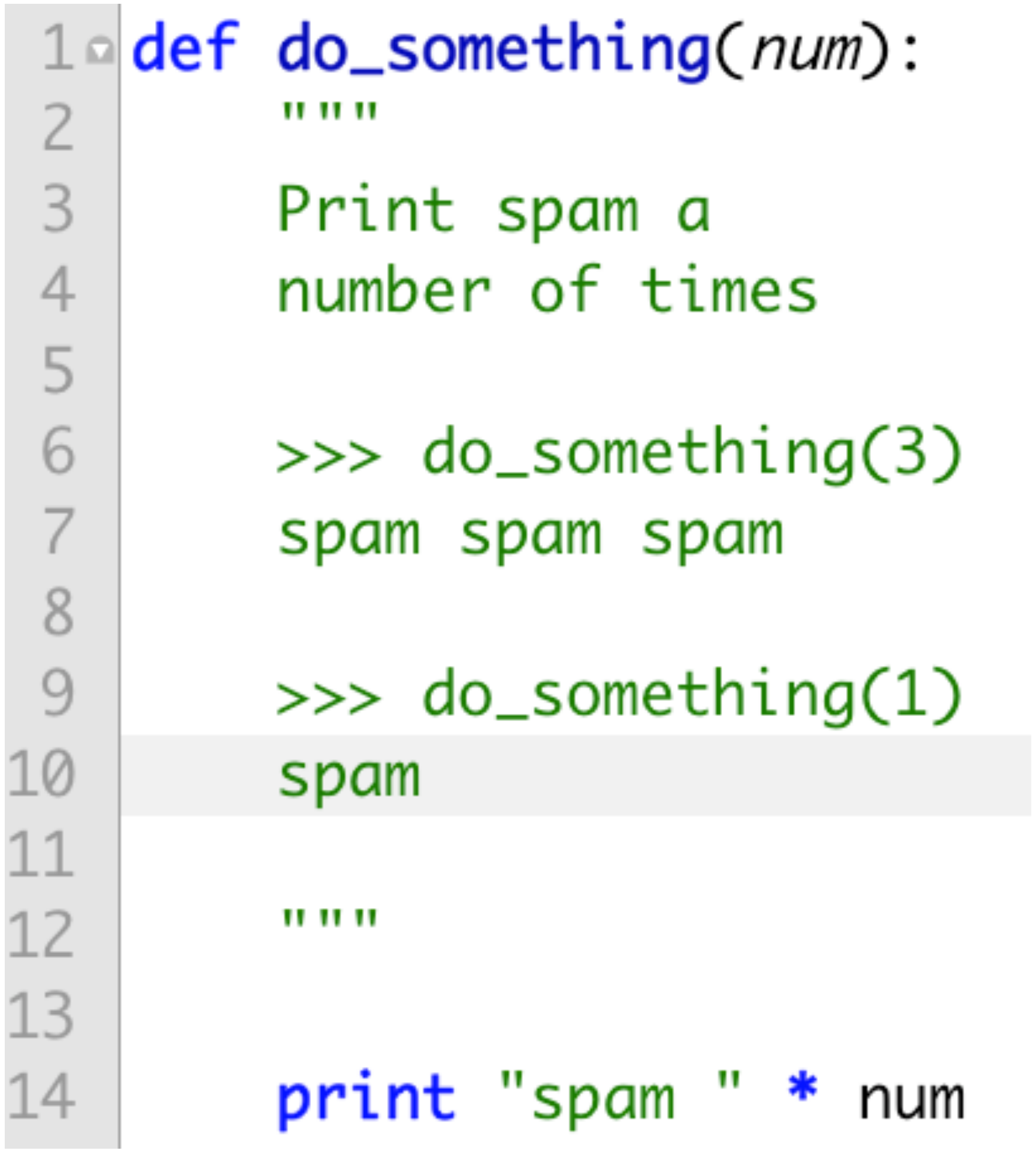
```python
def do_something():

    pass
```

Docstrings



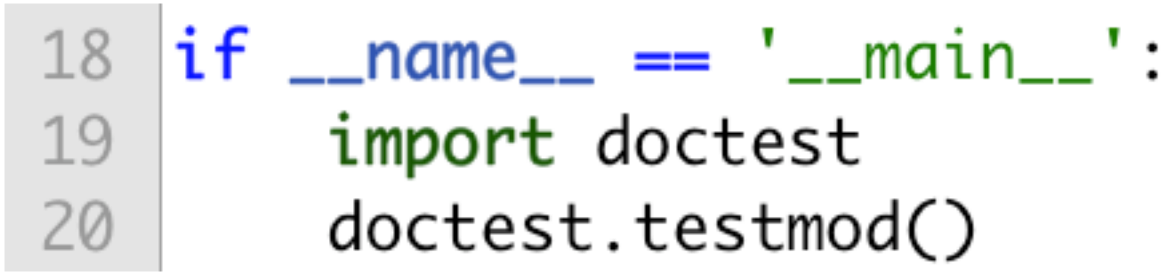
```
def do_something(num):
    """
    Print spam a
    number of times
    """

    print "spam " * num
```

doctest



```python
def do_something(num):
    """
    Print spam a
    number of times

    >>> do_something(3)
    spam spam spam

    >>> do_something(1)
    spam

    """

    print "spam " * num
```

Running a doctest



```python
if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

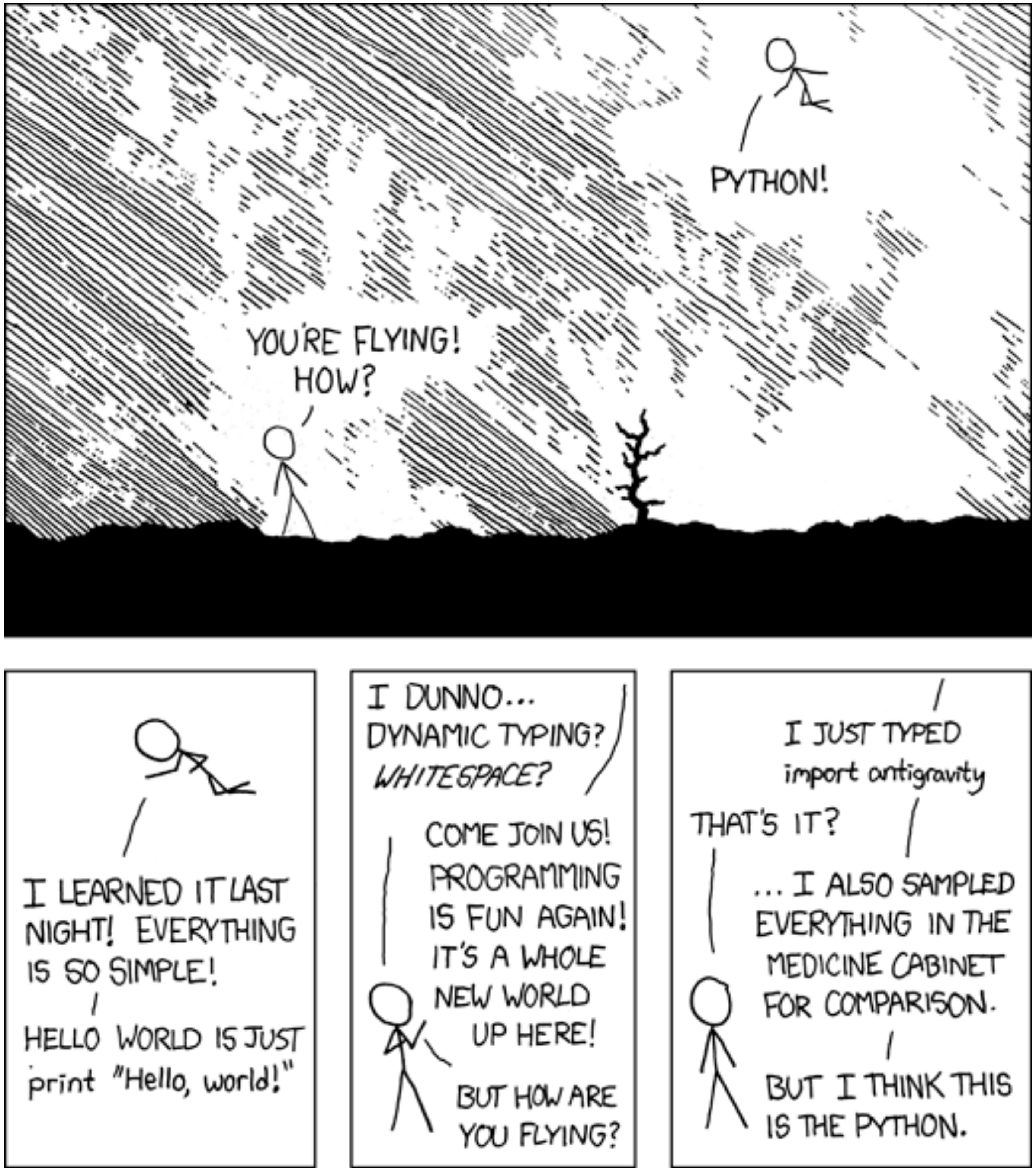
Batteries Included:

http://docs.python.org/library/

sqlite
regular expressions
date, time, calendar
os (files, directories, directory trees)
sys (stdin, stdout)
serialization
compression
threading
e-mail
JSON
XML
CGI
HTTP (urllib, urllib2)
unit testing

import

```
import sqlite3

conn = sqlite3.connect('/tmp/db.sqlite3')
cursor = conn.execute("SELECT first, last FROM people")

for first, last in cursor.fetchall():
    print "%s, %s" % (last, first)
```


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.

http://xkcd.com/353/

alt text:

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

```python
import this
```

```
In [1]: 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!
```

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Readability counts.

Special cases aren't special enough to break the rules.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Now is better than never.

Now is better than never.

Although never is often better than *right* now.

No Errors!

No Errors!

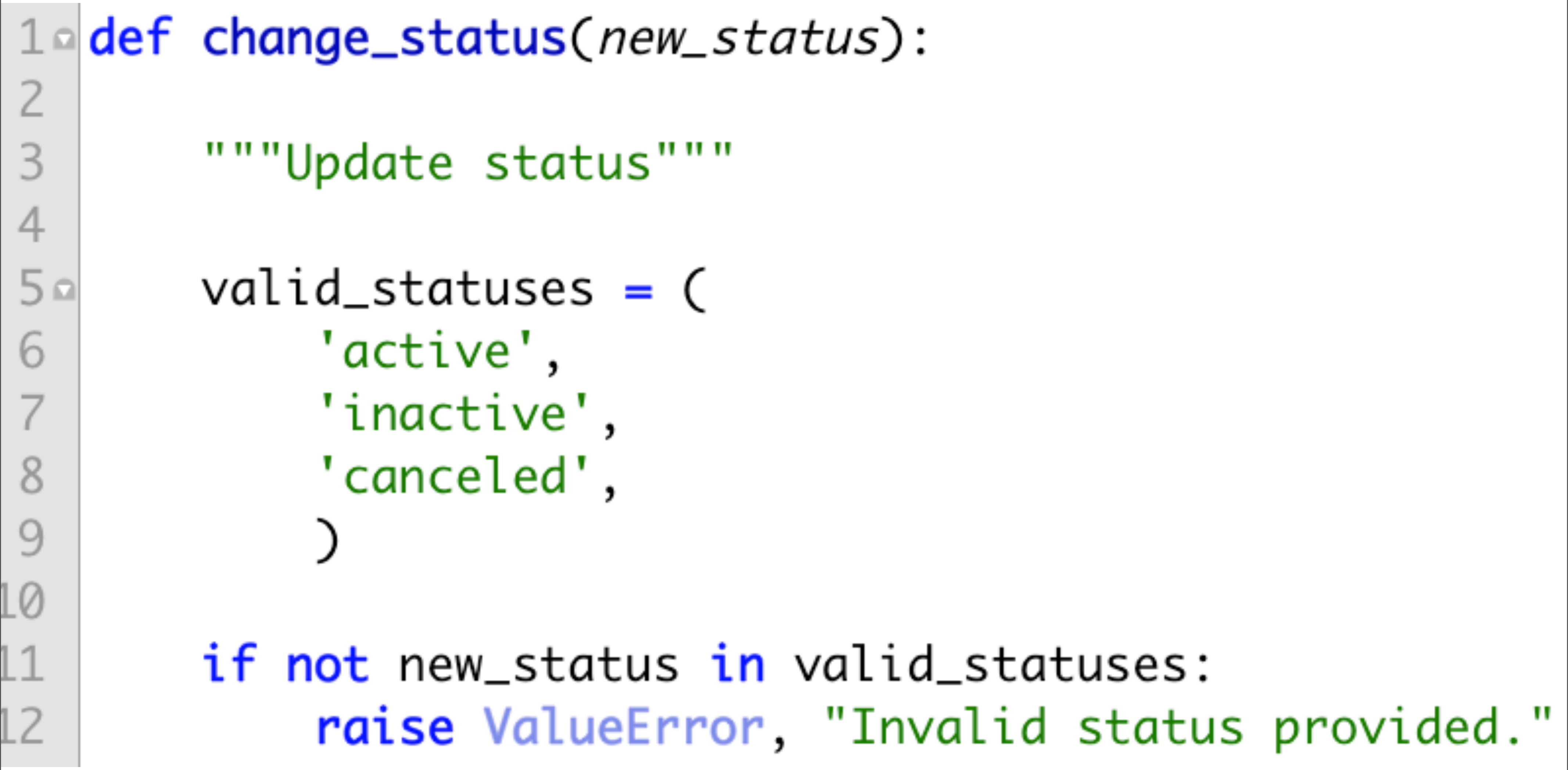
(we call them ‘exceptions’)

No Errors!

(we call them ‘exceptions’)

Errors need to be “trapped.”
Exceptions are “handled.”

```python
try:
    print x * 2

except TypeError:
    print "Non-numeric input!"
```

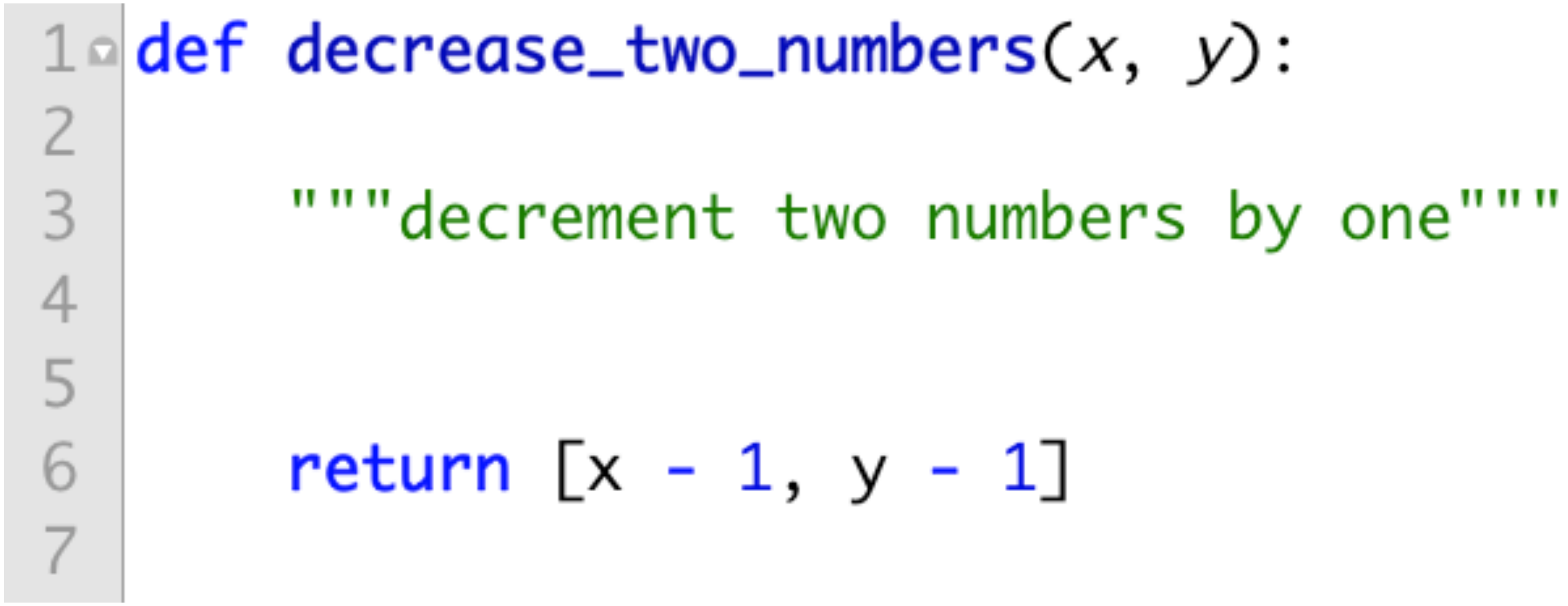


```python
def change_status(new_status):

    """Update status"""

    valid_statuses = (
        'active',
        'inactive',
        'canceled',
        )

    if not new_status in valid_statuses:
        raise ValueError, "Invalid status provided."
```

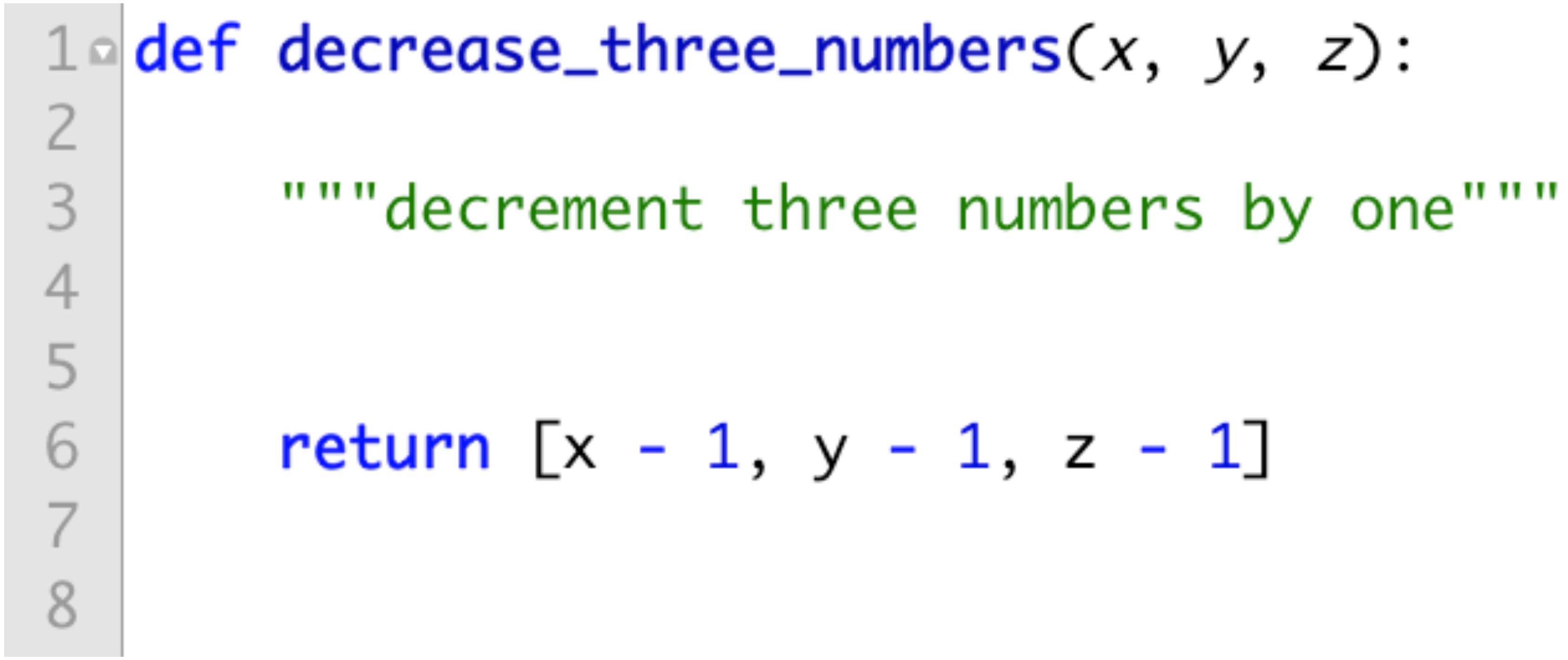
Time Travel!

```
In [17]: 9 / 5
Out[17]: 1

In [18]:

In [19]: from __future__ import division

In [20]:

In [21]: 9 / 5
Out[21]: 1.8
```

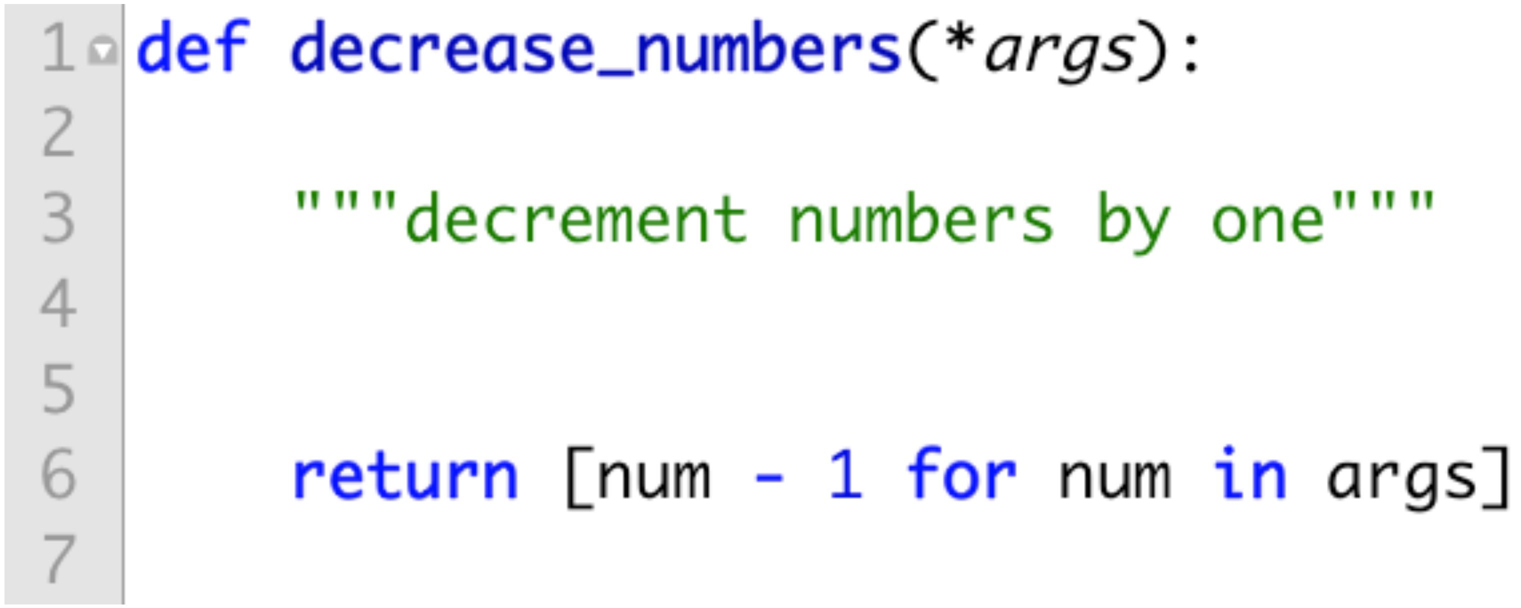
```
In [10]: from __future__ import braces
---------------------------------------------
SyntaxError: not a chance (<ipython console>,
```



```python
def decrease_two_numbers(x, y):

    """decrement two numbers by one"""


    return [x - 1, y - 1]

```



```python
def decrease_three_numbers(x, y, z):

    """decrement three numbers by one"""


    return [x - 1, y - 1, z - 1]


```

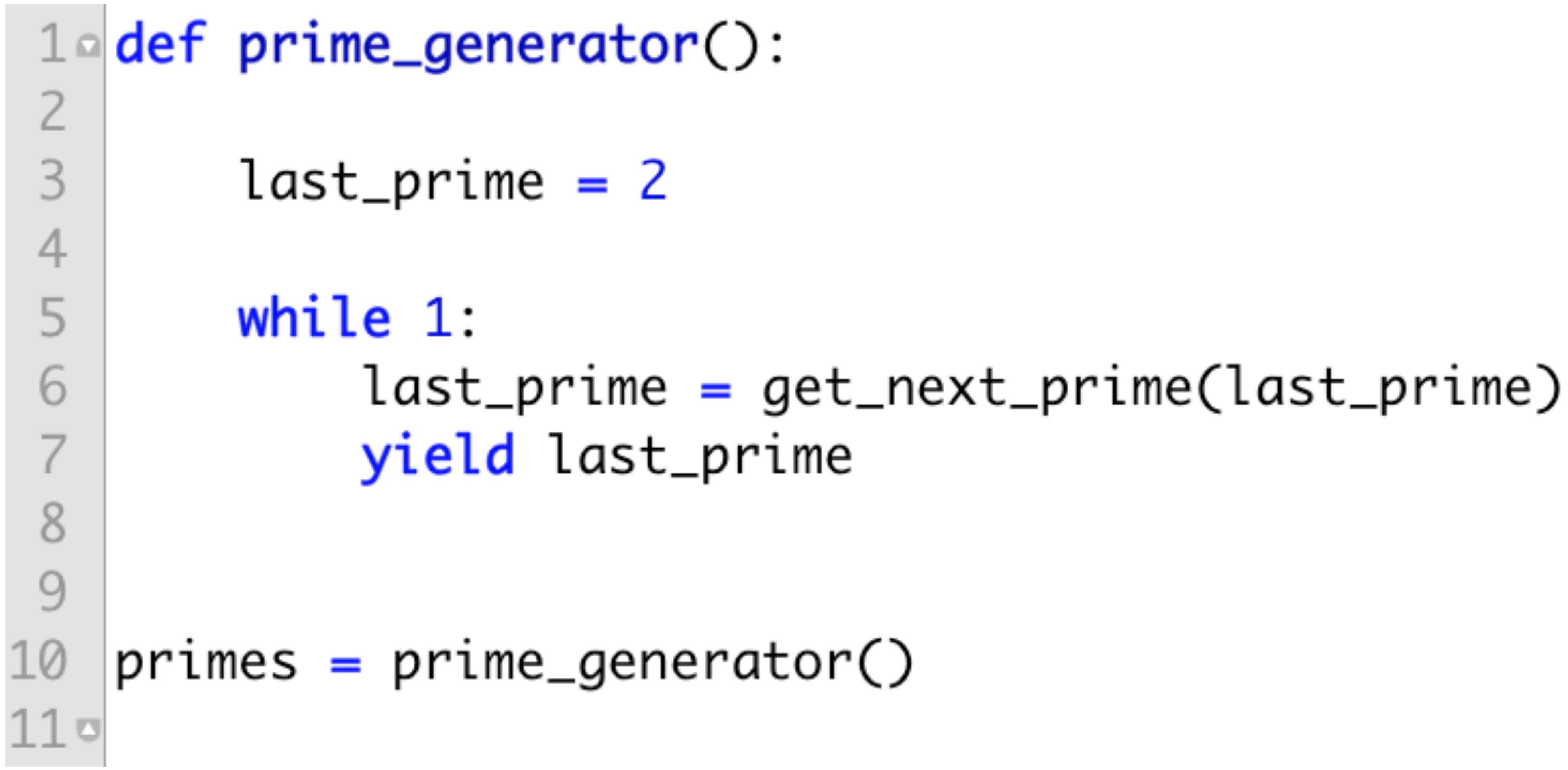


```python
def decrease_numbers(*args):

    """decrement numbers by one"""


    return [num - 1 for num in args]

```

```python
decrease_two_numbers(3, 4)
decrease_three_numbers(3, 4, 5)

some_numbers = [1, 3, 9, 4, 2, 17]
decrease_numbers(*some_numbers)
```

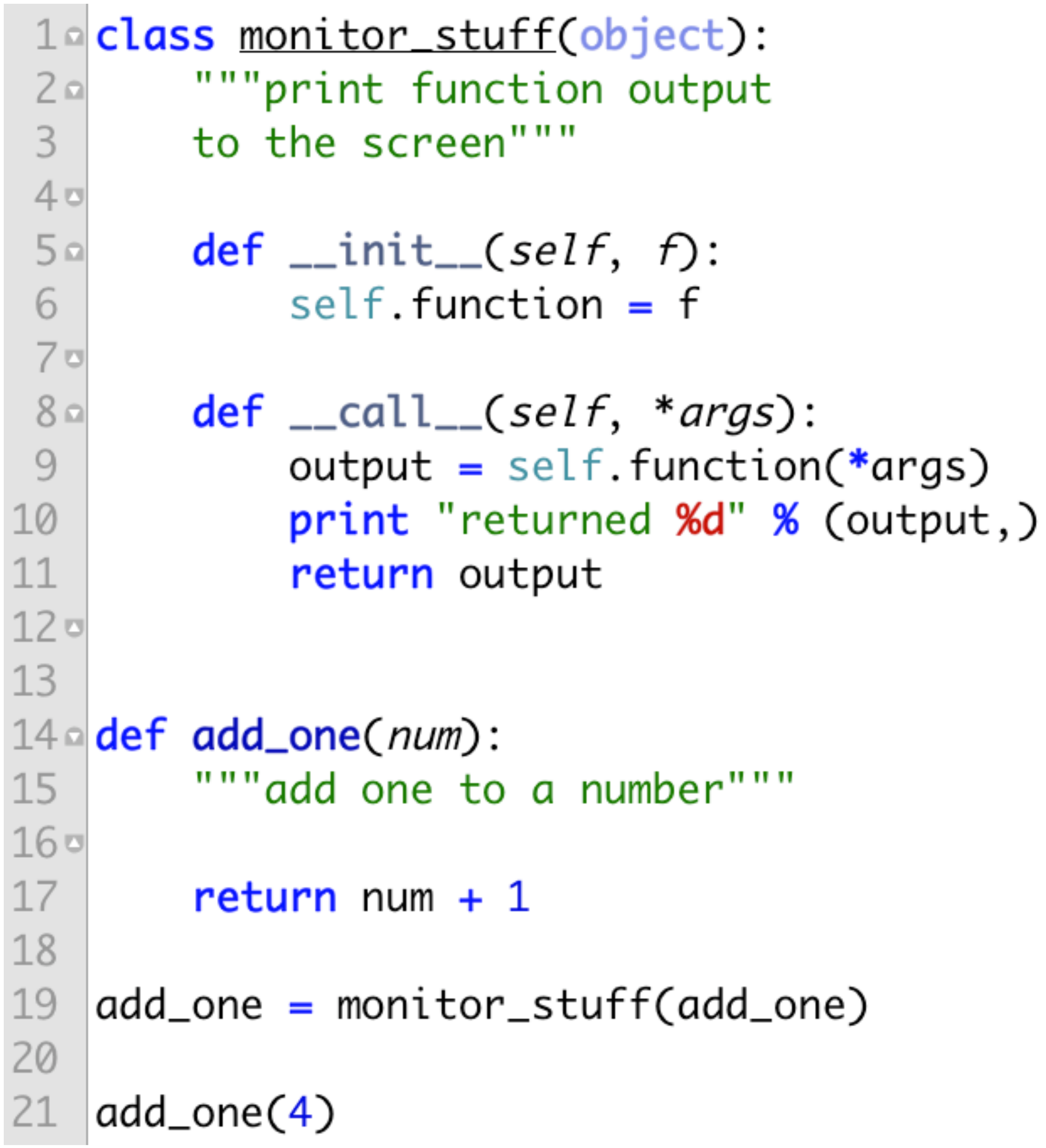
Advanced Stuff Made Easy

Generators



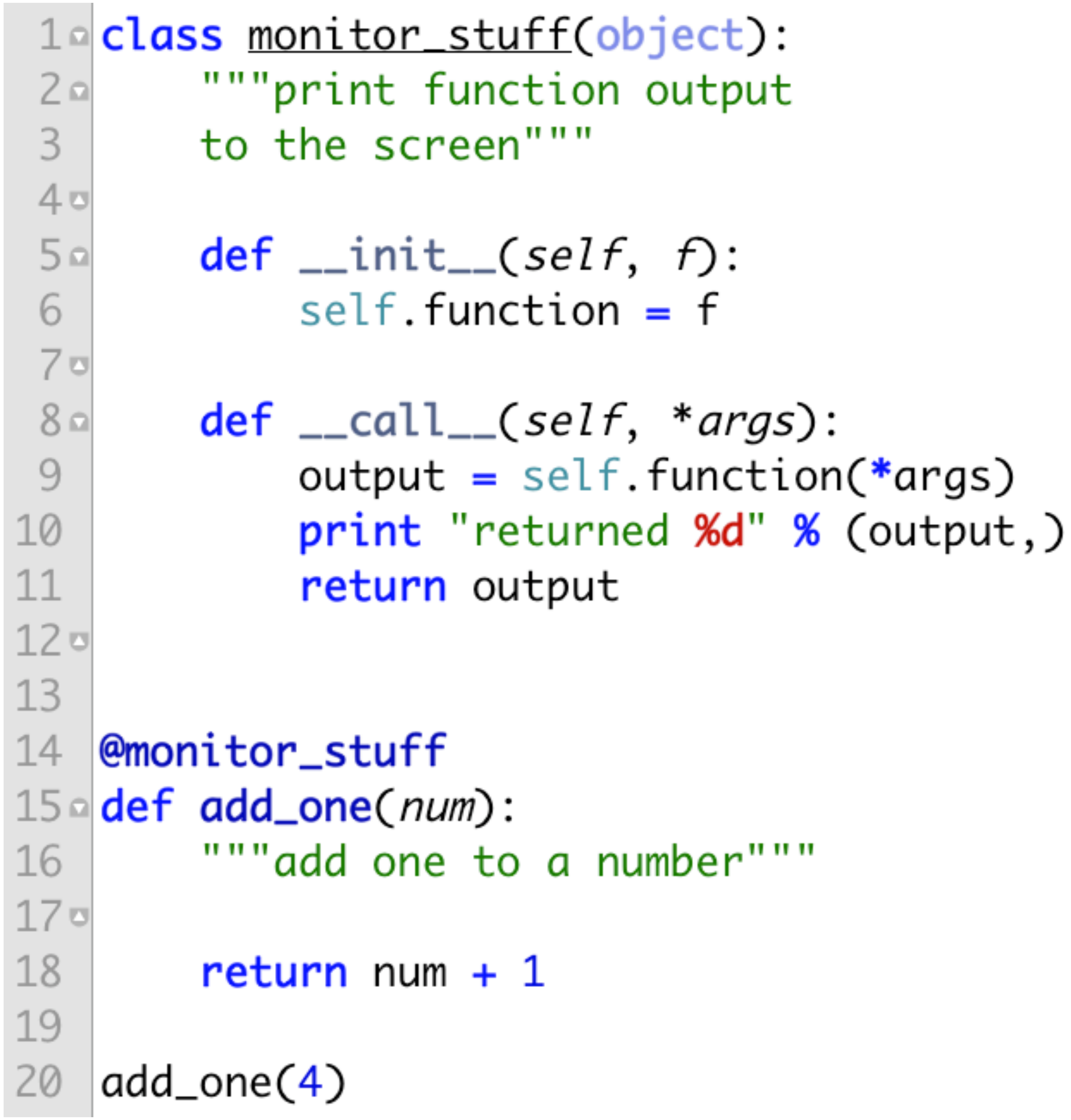
```python
def prime_generator():

    last_prime = 2

    while 1:
        last_prime = get_next_prime(last_prime)
        yield last_prime


primes = prime_generator()

```

Decorators



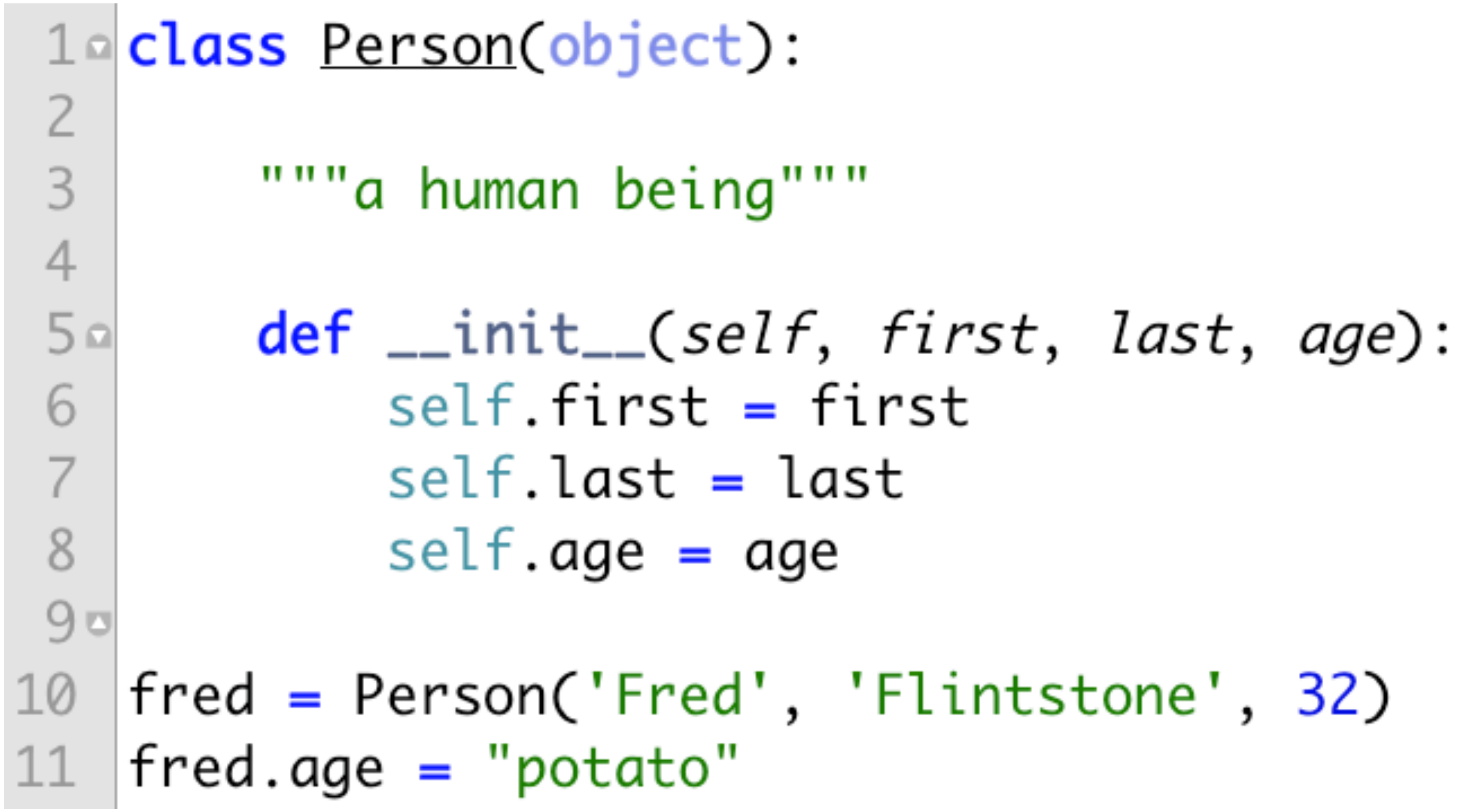
```python
class monitor_stuff(object):
    """print function output
    to the screen"""

    def __init__(self, f):
        self.function = f

    def __call__(self, *args):
        output = self.function(*args)
        print "returned %d" % (output,)
        return output


def add_one(num):
    """add one to a number"""

    return num + 1

add_one = monitor_stuff(add_one)

add_one(4)
```

Decorators



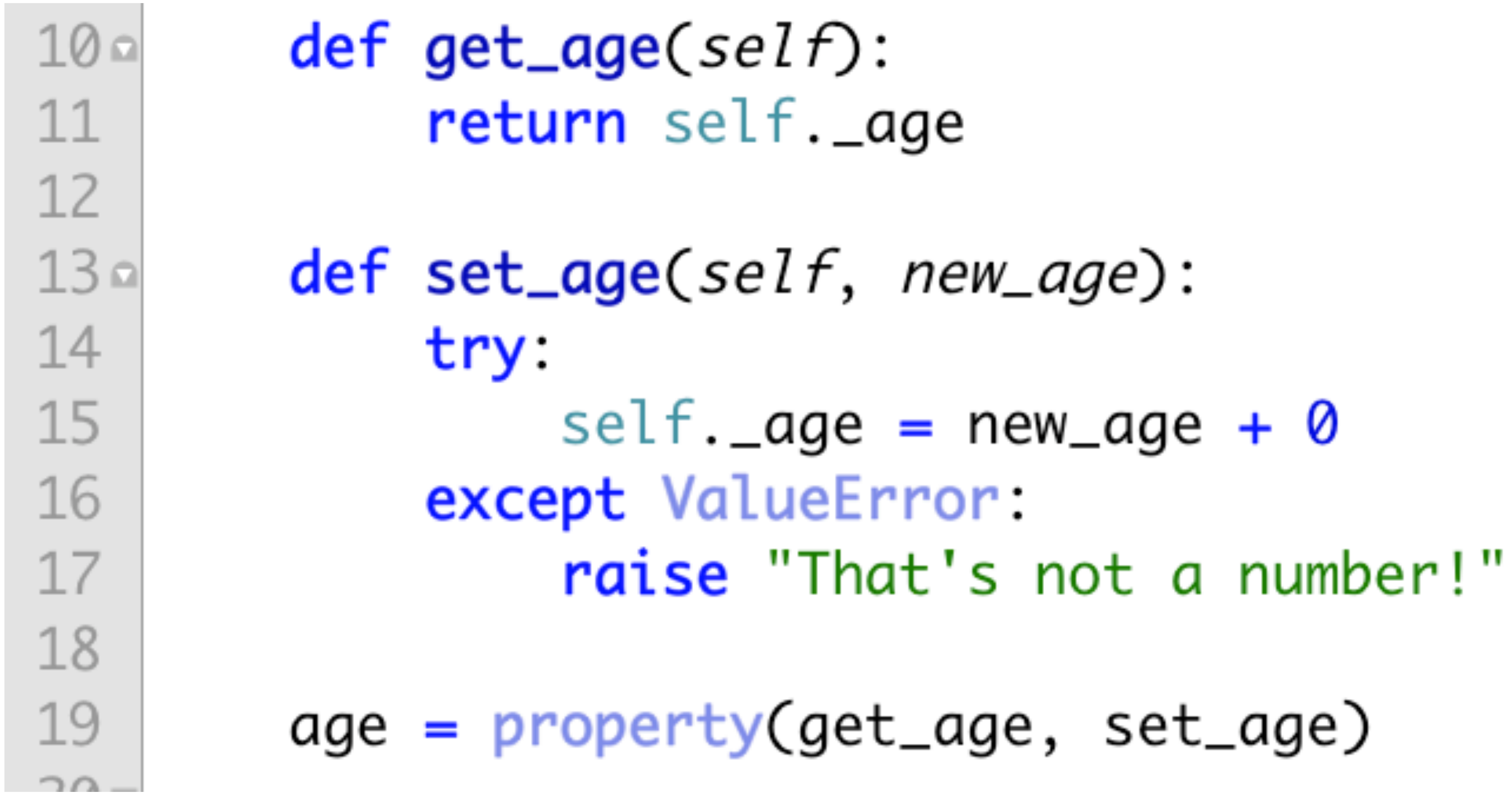
```python
class monitor_stuff(object):
        """print function output
        to the screen"""

        def __init__(self, f):
                self.function = f

        def __call__(self, *args):
                output = self.function(*args)
                print "returned %d" % (output,)
                return output


@monitor_stuff
def add_one(num):
        """add one to a number"""

        return num + 1

add_one(4)
```

Class Attributes 1



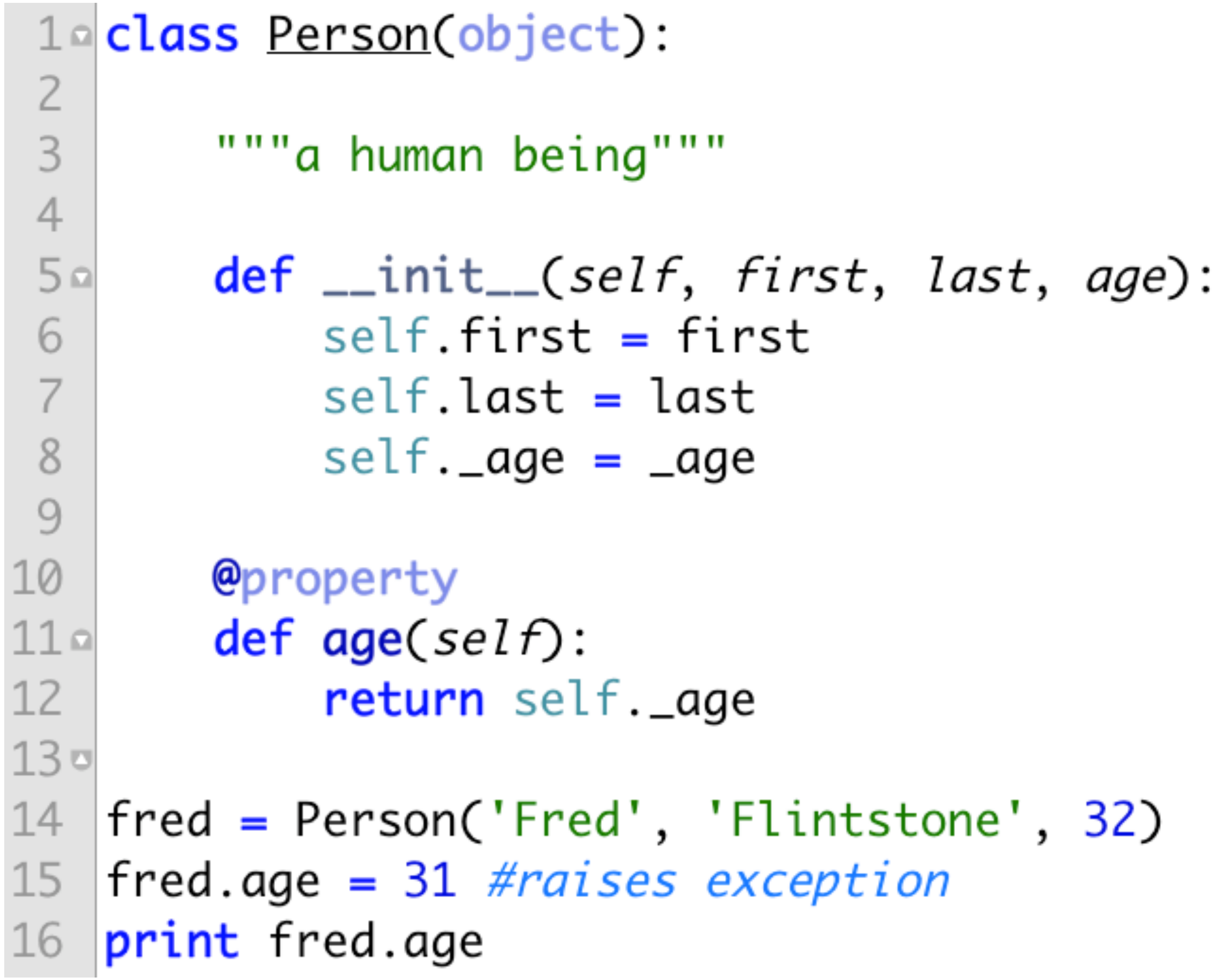
```python
class Person(object):

    """a human being"""

    def __init__(self, first, last, age):
        self.first = first
        self.last = last
        self.age = age

fred = Person('Fred', 'Flintstone', 32)
fred.age = "potato"
```

Class Attributes 2



```
    def get_age(self):
        return self._age

    def set_age(self, new_age):
        try:
            self._age = new_age + 0
        except ValueError:
            raise "That's not a number!"

    age = property(get_age, set_age)
```

Class Attributes 3 (Static)



```python
class Person(object):

    """a human being"""

    def __init__(self, first, last, age):
        self.first = first
        self.last = last
        self._age = _age

    @property
    def age(self):
        return self._age

fred = Person('Fred', 'Flintstone', 32)
fred.age = 31 #raises exception
print fred.age
```

Okay, enough code.

Packages

PyPy (formerly "Cheese Shop"), Python's answer to CPAN

http://pypi.python.org/pypi

Package Installs

easy_install
Python's "apt-get"

http://pypi.python.org/pypi/setuptools/

ORM

SQL Alchemy

http://www.sqlalchemy.org/

Networking

Twisted
easy client/server apps & more

http://twistedmatrix.com/trac/

Web Frameworks

Django*

http://www.djangoproject.com/

*TurboGears, CherryPy, & others exist, but Django has the largest community and best documentation.

Cross-Platform (os)

Linux, Mac, Windows, BSD, etc., etc.,

http://python.org/

JVM Support

Who wants to use Java when Python's an option?

http://www.jython.org/

Microsoft .NET*

Who wants to use VB.NET or C# when Python's an option?

http://www.codeplex.com/IronPython

*Microsoft hired IronPython's creator, Jim Hugunin, and pays him to develop IronPython

Wide Adoption

Google*
NASA
National Geographic

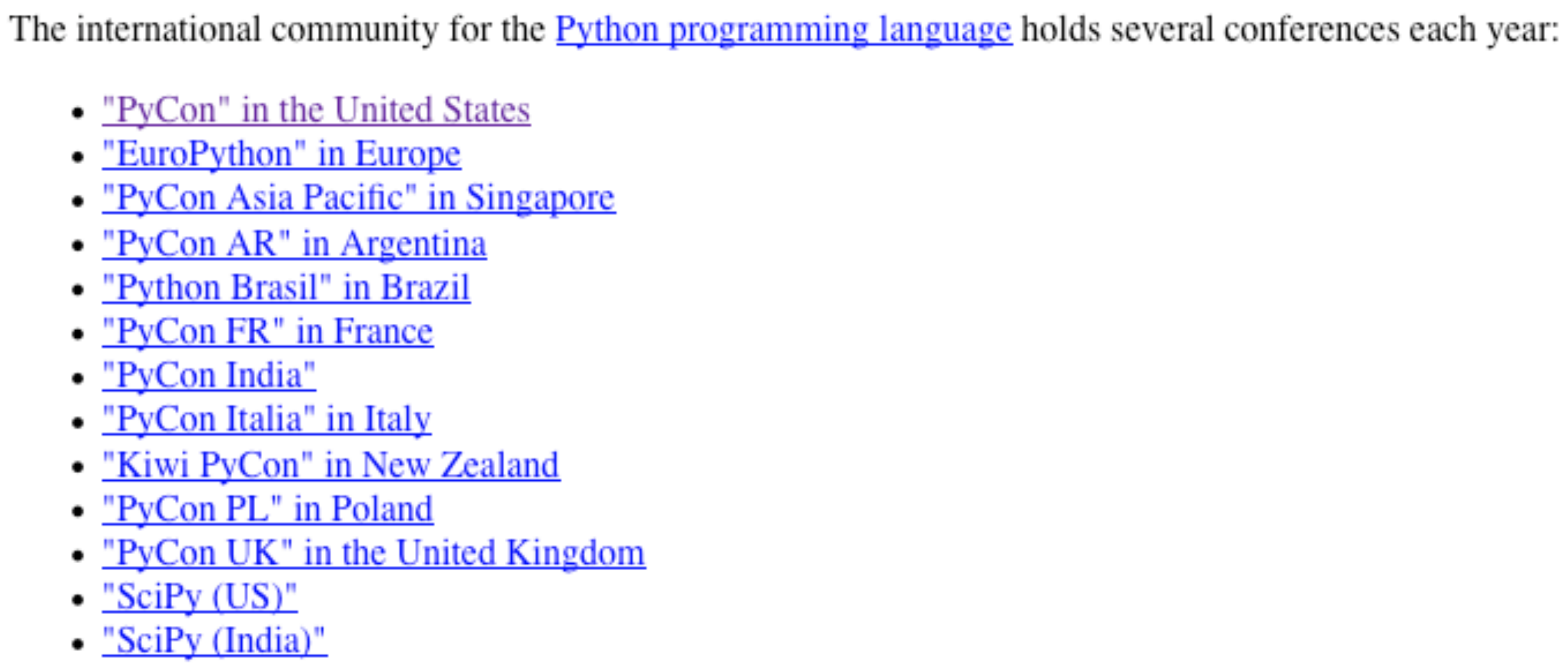
*Google hired Python’s creator, Guido van Rossum, and pays him to develop Python

Google!

Unladen Swallow

PyCon

http://www.pycon.org/


The international community for the Python programming language holds several conferences each year:

- "PyCon" in the United States
- "EuroPython" in Europe
- "PyCon Asia Pacific" in Singapore
- "PyCon AR" in Argentina
- "Python Brasil" in Brazil
- "PyCon FR" in France
- "PyCon India"
- "PyCon Italia" in Italy
- "Kiwi PyCon" in New Zealand
- "PyCon PL" in Poland
- "PyCon UK" in the United Kingdom
- "SciPy (US)"
- "SciPy (India)"

DjangoCon

http://www.djangocon.org/

US & Europe

Contact

Shawn@Milochik.com
@ShawnMilo

(Yes, I will answer Python questions.)

Mailing List

http://mail.python.org/mailman/listinfo/python-list