Python is weird.

Me

- Jedi / jedi-vim (both ~ 2,500 stars on Github)
- Working @ cloudscale.ch

Python 2 Job Security

```
Python 2.7:

>>> __builtins__.True = False
>>> True
False
```

How Python Works

- Source Code
- → Tokenizer
- → → Parser / AST
- → → → Bytecode
- → → → → Functions and Dicts

The Tokenizer

- Is a collection of regular expressions
- Separates 1.0 from “asdf” and names
- Not needed in a lot of languages

bar = 1 or “”

```
$ python -m tokenize foo.py
1,0-1,3:        NAME            'bar'
1,4-1,5:        OP              '='
1,6-1,7:        NUMBER          '1'
1,8-1,10:       NAME            'or'
1,11-1,13:      STRING          '""'
1,13-1,14:      NEWLINE         '\n'
2,0-2,0:        ENDMARKER       ''
```

The Tokenizer

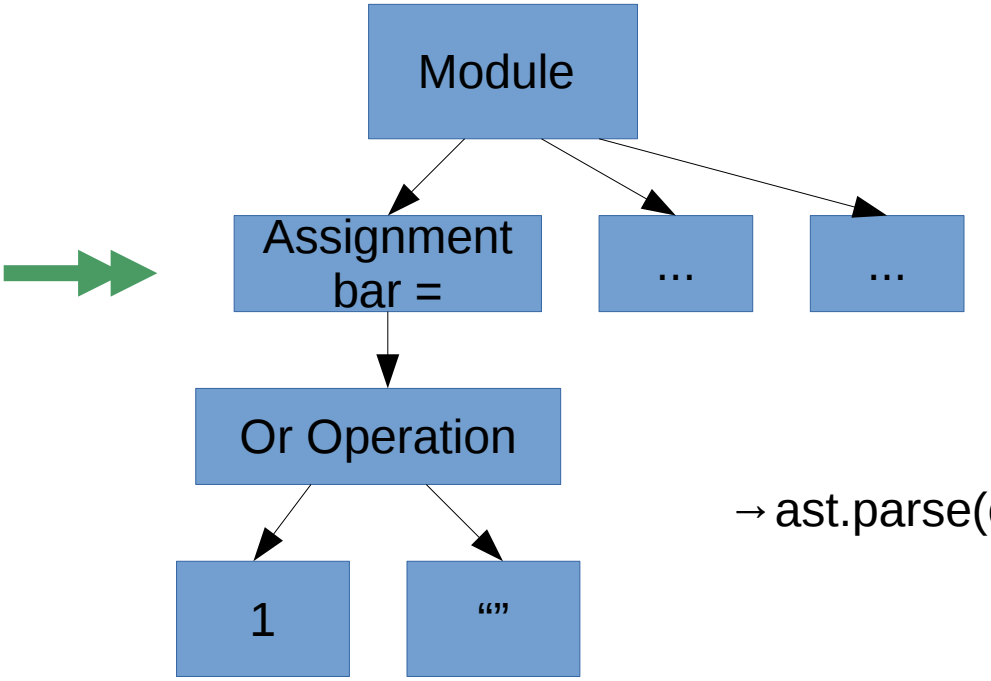
Is this valid Python code?

```
br''or.0jif.1else-...
```

The Parser

- Translates tokens into a parser tree

```
$ python -m tokenize foo.py
1,0-1,3:        NAME            'bar'
1,4-1,5:        OP              '='
1,6-1,7:        NUMBER          '1'
1,8-1,10:       NAME            'or'
1,11-1,13:      STRING          '""'
1,13-1,14:      NEWLINE         '\n'
2,0-2,0:        ENDMARKER       ''
```



→ ast.parse(code)

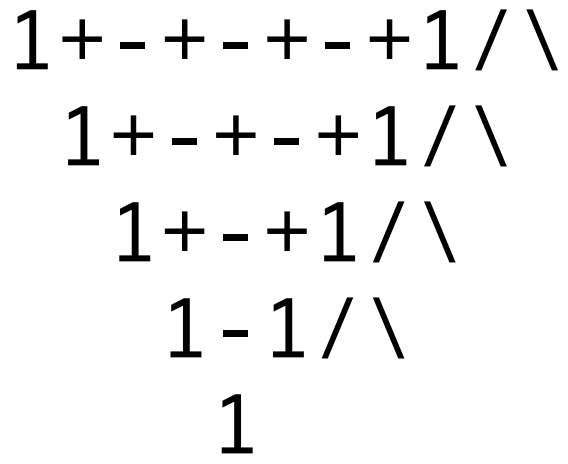
Odd Parser Stuff #1

Is this valid Python code?

```
++1;
--1;
```

Odd Parser Stuff #2

Ascii art yaayy!



Odd Parser Stuff #3

How about lambda generators?

```
lambda x: (yield x)
lambda x: (yield from x)
```

Odd Parser Stuff #4

It can always get worse!

```
def x():
  yield lambda x=(yield): (yield x)
```

Execution

Let’s insert that bytecode into dictionaries.

```
$ python -m dis foo.py
  1           0 LOAD_CONST               0 (1)
              3 JUMP_IF_TRUE_OR_POP      9
              6 LOAD_CONST               1 ('')
        >>    9 STORE_NAME               0 (bar)

  [..]
```

Dictionaries

C again...

```
while (c = getchar(), c != 'x') {
    printf()
}
```

Dictionaries

C again...

```
while (c = getchar(), c != 'x') {
    printf()
}
```

Python:

```
while globals().__setitem__('z', input()) or z != 'x':
   print(z)
```

Classes Are Actually Functions

```
from dave import utils

def func():
    foo = 'yay'
    bar = 'arrrr'
    return foo

func = utils.convert_byte_code(func)

Cls = __build_class__(func, 'MyClass')

Cls.bar # Returns ‘arrrr’
Func()  # Returns ‘yay’
```

Me

Github: davidhalter
Twitter: jedidjah_ch
Work: dave@cloudscale.ch

Bonus: A Riddle

Github: davidhalter
Twitter: jedidjah_ch
Work: dave@cloudscale.ch

```
lambda:br''or.6*.3+.4jif.1else-...();float.__int__(eval('_')().imag*100);chr(int(str(_),16)+1)
```