

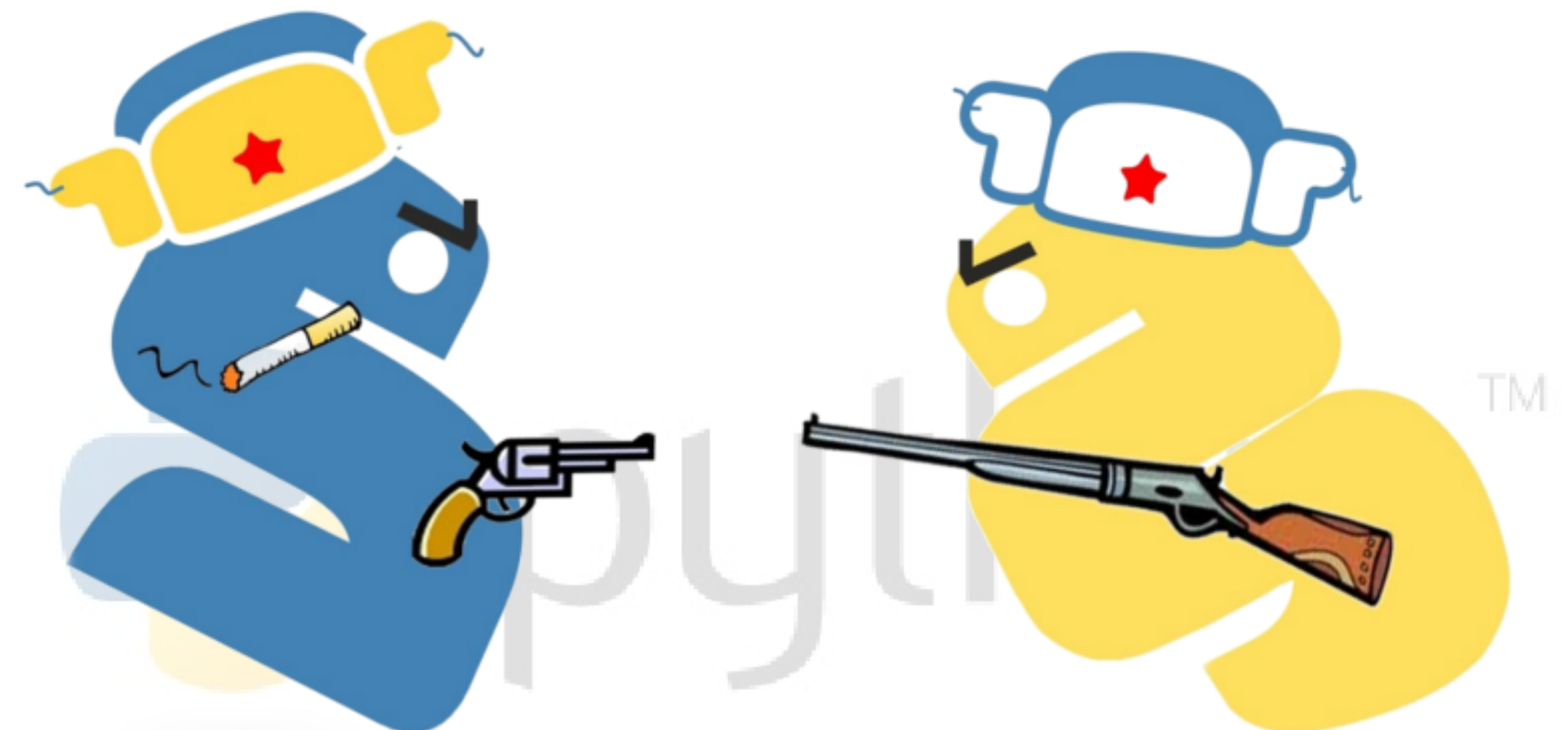


Y solo tenemos unos ~40 minutos

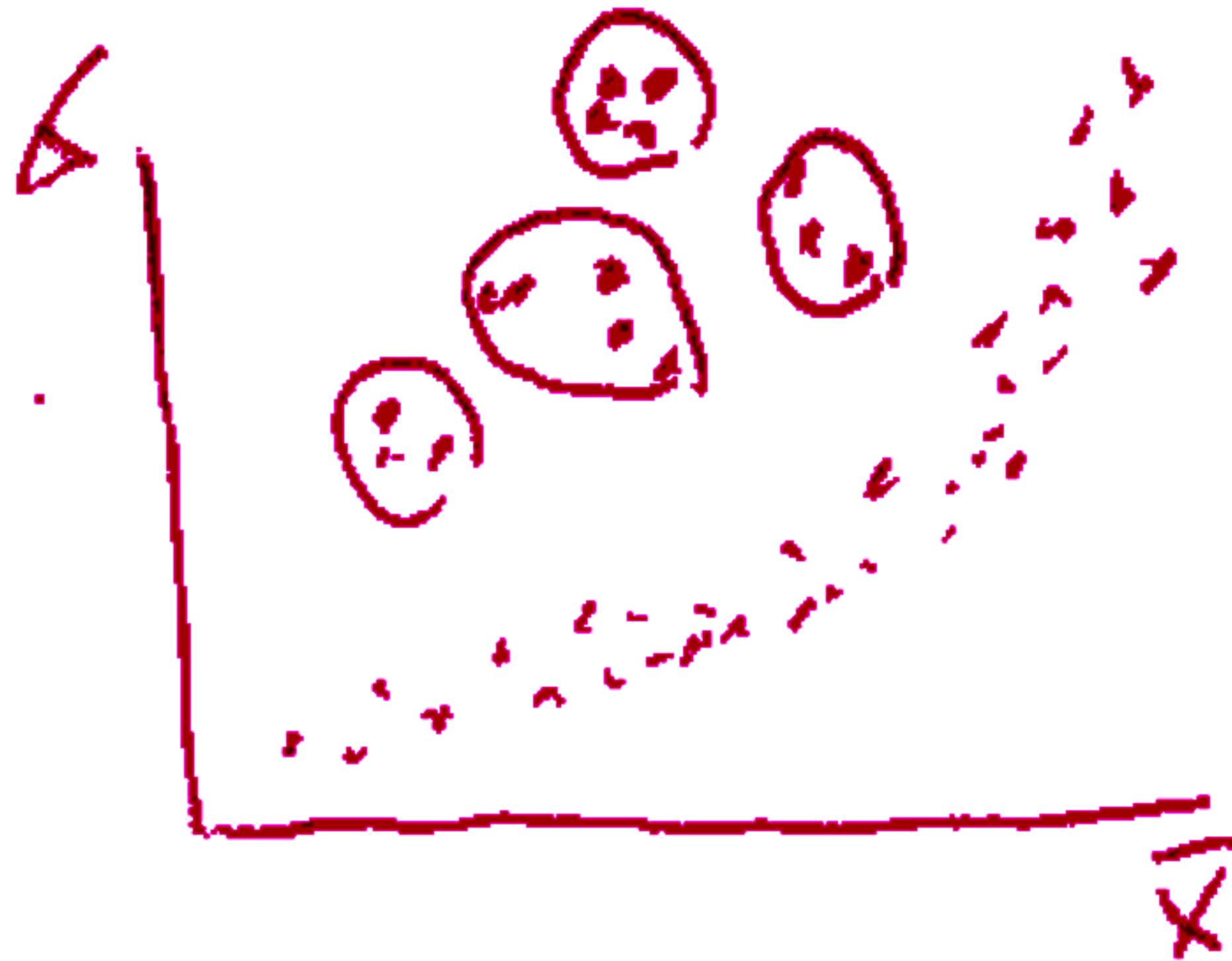
Basado en <http://goo.gl/2Tj7yc>

::Origen

- Sucesor del lenguaje ABC (no, a mi tampoco me suena)
- Lleva desde 1991 dando vueltas
- Guido Van Rossum es el **BDFL**
- Gracias a **Google** y **Django** ahora es un lenguaje supertrendy



::Me dijeron que era así



Hay estrellas que varían, y si gráficas el promedio de brillo contra su desviación puedes agruparlas

::La cruda realidad

::Gracias Python

::Python Zen

Mejor explícito que implícito

Mejor simple que complejo

Mejor plano que anidado

In the face of ambiguity, refuse the temptation to guess.

There should be one and preferably only one obvious way to do it.

import this



::Implementaciones

Implementation	Virtual Machine	Language
CPython	CPythonVM	C
Jython	JVM	Java
IronPython	CLR	C#
Brython	Javascript Interpreter(V8)	Javascript
RubyPython	RubyVM	Ruby



::Lenguaje

- Lenguaje de alto nivel
- Fuertemente tipado
- No hace comprobación de tipos estática
- No hay que declarar variables
- Indentación obligatoria (haciendo amigos)
- Multiparadigma
 - Funcional
 - Orientado a objetos
 - Imperativo

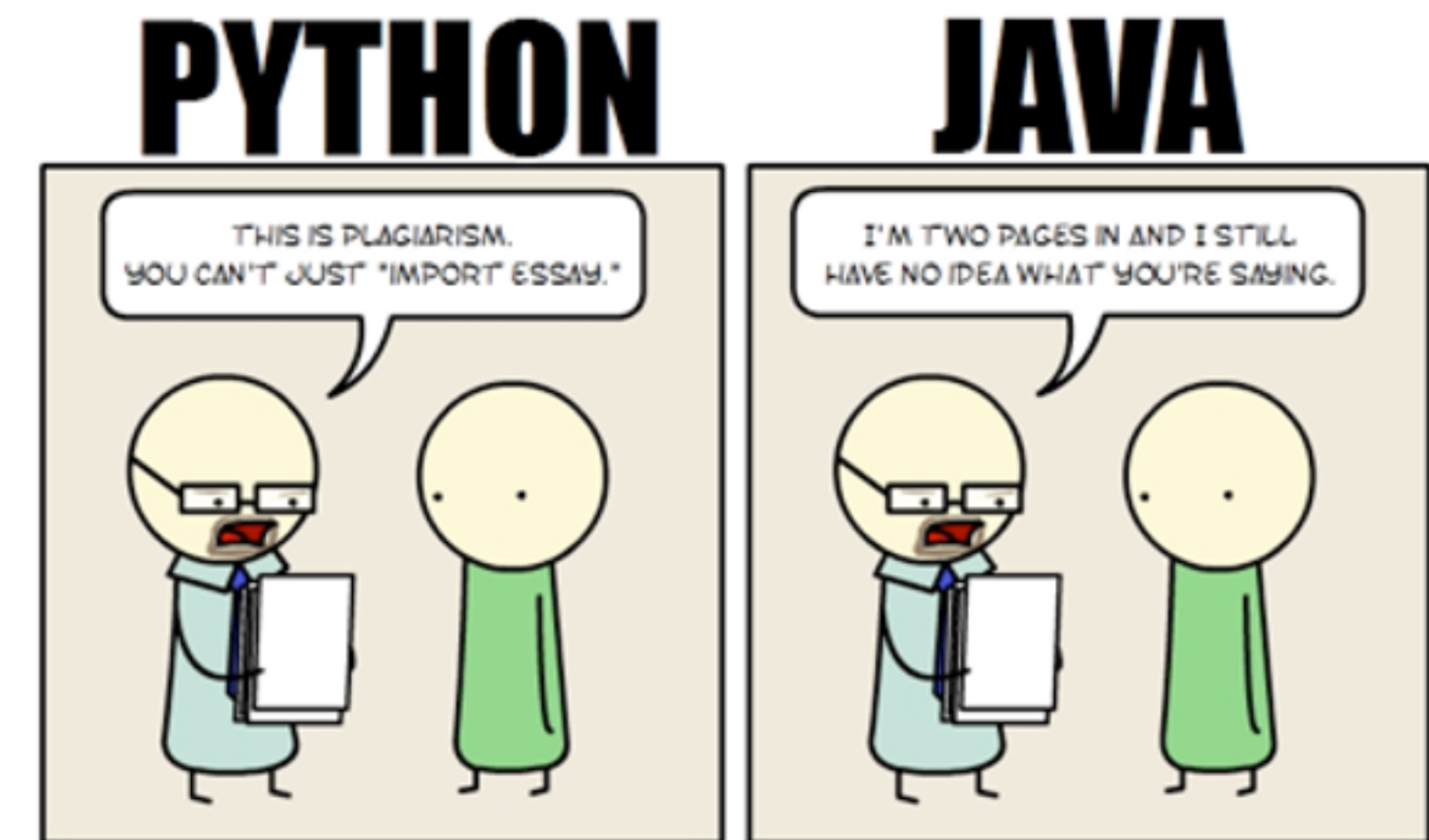


::Batteries included

- Sockets
- DB-API (mysql, postgresql, sqlite, oracle)
- urllib/urllib2
- re (gular expressions)
- Threading
- gettext
- pdb
- pickle



::Ecosistema



- gui: PyGTK, PyQt, wxPython
- juegos: Pygame
- web: Django / Turbogears / Pylons / Zope
- net: Twisted, Orbited
- ciencia: NumPy, SciPy, Biopython
- DVCS: Mercurial



::Hello World!

```
print "Hello world"
```



::Sintaxis

comentarios

happy = True

py = 3.14159

s1 = "cadena"

fib = [0,1,1,2,3,5,8,13]

*h = {'nombre': "Pijus", "apellido":
"Magnificus"}*

*lista = [x*2 for x in fib]*

s1,s2 = s2,s1 # asignación múltiple



::Tipos

- Enteros / flotantes

100

3.1

- Booleanos

True / False

- Cadenas (inmutables)

"cadena"

'cadena'

"""cadena"""



::Tipos

- Tuplas (inmutables)

`(1, 2, 3, 4)`

- Listas

`l = ['a', 1, None, lambda x: x]`
`l[1:3]`

- Diccionarios

`{'min': 250, 'max': 400}`

- Conjuntos

- Clases

- Funciones (objetos de primer orden)



::Control

***if** expresion:*

do()

***elif** otra_expresion:*

dont()

else:

omg()



::Control

```
while True:
```

```
    do_more()
```

```
    break
```

```
for i in range(10):
```

```
    print i
```

```
for k,v in diccionario:
```

```
    print "%s=%s" % (k, str(v))
```



::Funciones

```
def awake_neo():  
    pass
```

```
def fact(n):  
    """ Otro ejemplo de factorial no...  
    """  
    if n == 1:  
        return 1  
    else:  
        return n * fact(n-1)
```



::Excepciones

try:

kill(chuck)

except RoundKickException, rke:

argh(rke)

finally:

you_die()

- Las excepciones son clases



::Módulos

- Sirven para organizar el código
- Pueden definirse variables, clases y funciones
- Se organizan en una estructura jerárquica
- *dir(modulo)* saca una lista de propiedades

import modulo

import modulo.otromodulo as om

from modulo import otromodulo



::Clases

- Lenguaje OO
- Herencia múltiple
- No hay propiedades privadas (convención)
- Sintaxis un poco rara para los métodos
- Métodos mágicos

`__init__`

`__iter__`

`__get__`



::Classes

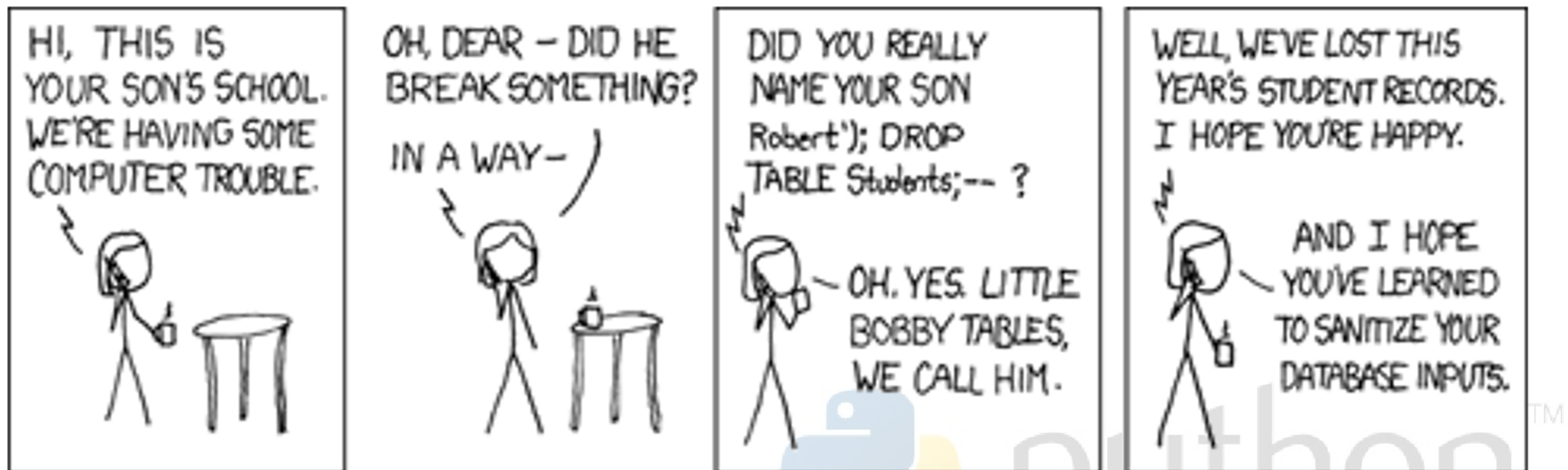
```
class RoundKicker(object):  
    """ Aquel que gira """  
    def __init__(self, name):  
        self.name = name  
    def unkillable(self):  
        return self.name == 'chuck'
```

```
chuck = RoundKicker("chuck")  
chuck.unkillable()
```



::Un poco de acción

- Pequeño script de ejemplo
- Extrae texto de un PDF y lo convierte en un CSV



::Y ahora qué

- Python tutorial

<http://docs.python.org/tut/>

- Dive into Python

<http://diveintopython.org/>

- How to Think Like a Computer Scientist:
Learning with Python

<http://openbookproject.net/thinkCSpy/>



::Be pythonic, my friend

EOF

