

# Refactoring

comece agora

# Entao você é o famoso

@tarsisazevedo

desenvolvedor globo.com

#cobrateam member

# E vocês?

Tdd?

Dojo?

Python?

Refactoring?

# esse é voce?

- vou fazer essa gambiarra aqui, e **depois eu volto...**
- queria fazer isso aqui direito, mas **nao tenho tempo...**
- esse **codigo é tao simples** que nao merece um teste
- vou fazer de qualquer jeito porque **nunca mais vou voltar** nesse codigo mesmo...

Friday, July 1, 2011

ja deu essa desculpa? uma? todas?

# e por fim...

- "caro nerd, isso foi feito em dois dias, então, nao pertube." #priceless - <http://bit.ly/jzpcxj>

# You Are Doing It Wrong!



Friday, July 1, 2011

e agora, quem poderá  
nos defender...

# Chaplin?!





# Refactoring #wtf

melhorar o design do  
codigo de um programa  
funcional, não  
alterando seu  
comportamento. -  
Martin Fowler



Thereifixedit.com

Friday, July 1, 2011

apresentação formal



# HÃ?!



Friday, July 1, 2011



# Arrumar a bagunça



Friday, July 1, 2011

arrumar um design mal feito, remover código duplicado...  
código fácil de mudar, de ler, de reusar e extremamente flexível.  
zerar débitos técnicos, aceleração constante(física),  
fácil achar bugs



# Fix broking windows



Friday, July 1, 2011

Prag-Prog

a ruina começa com uma pequena janela quebrada.  
se vc deixa codigo ruim, logo seu software estará todo cheio de lixo!  
conserte o mais rapido possivel!

legal, #comofaz?

# comece certo



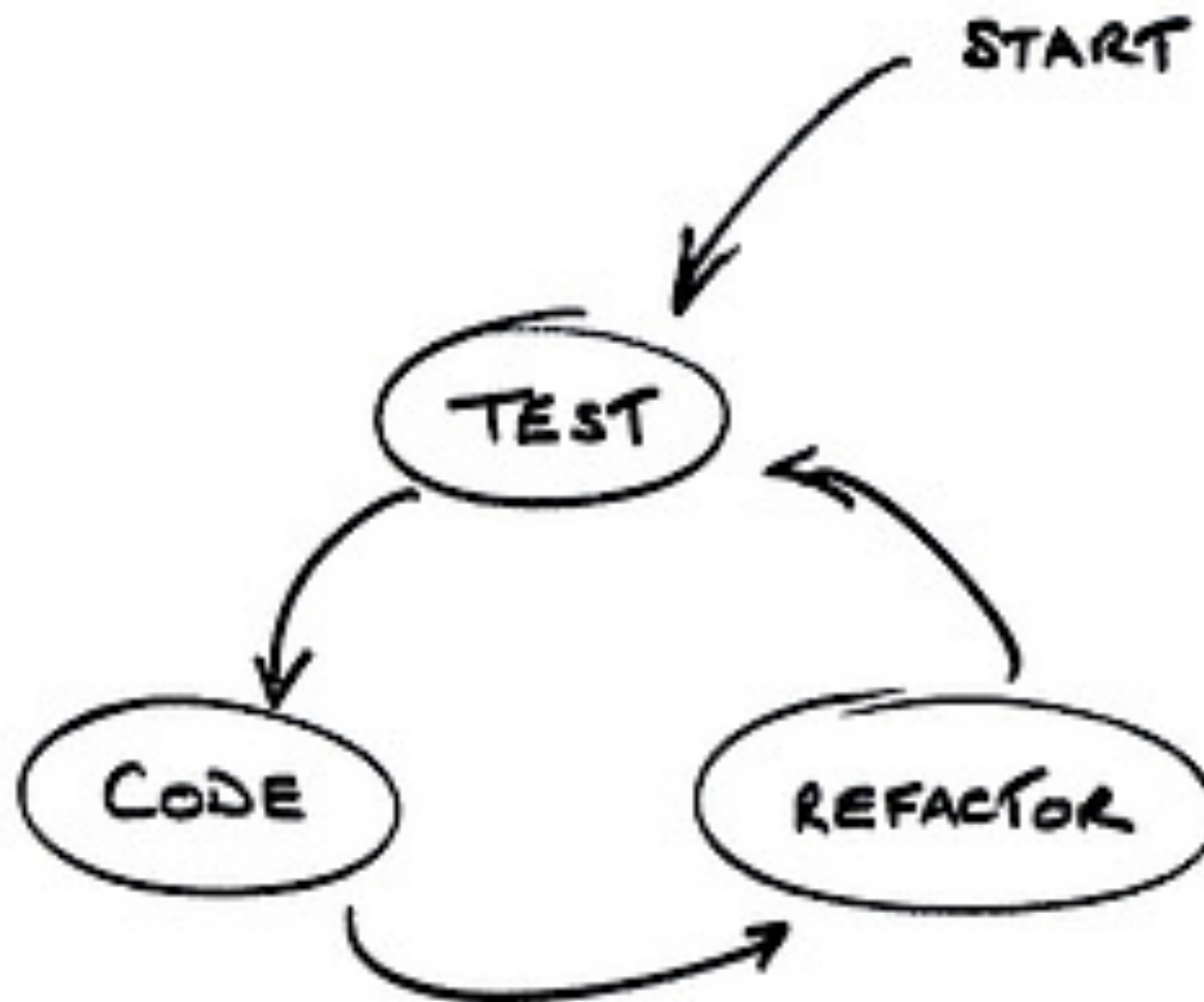
CODESMACK

CODESMACK

Friday, July 1, 2011

testar antes, design enxuto, evita repetição  
testes te fornecem segurança para refatorar sem medo!

# todo o tempo!



Friday, July 1, 2011

tem que ser parte do processo!  
nao tirar um dia, ou uma hora pra refatorar!  
tem de ser feito sempre!



# foco



Friday, July 1, 2011

saber o que refatorar  
nao se perder procurando codigo que voce nao está mexendo



# disciplina



Friday, July 1, 2011

quando refatorar  
ate onde ir – nao ficar polindo muito  
ou refatora, ou adiciona  
nao ser fanatico

don't repeat yourself

**DRY**

*Don't Repeat Yourself*

**CODESMACK**

# seja expressivo

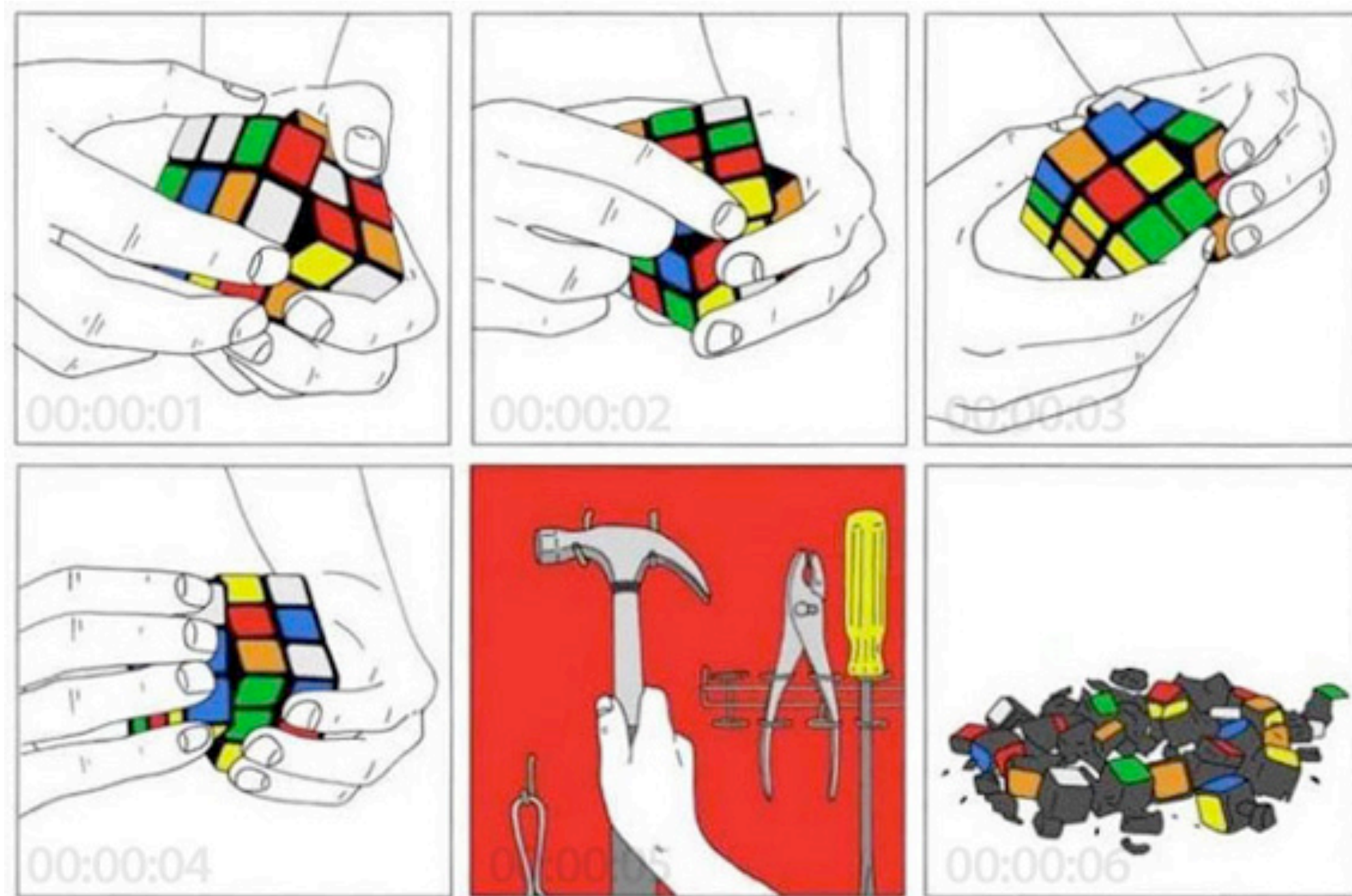


Friday, July 1, 2011

nomes que fazem sentido  
organização do código é importante



# ferramenta certa



Friday, July 1, 2011

conheça a linguagem/framework que voce trabalha; – o que ela pode fazer, – o que nao pode, – frameworks tem magicas  
use as ferramentas para analise estatica de codigo

# va devagar



Friday, July 1, 2011

vá devagar, faça uma alteração de cada vez. NAO CORRA!!!

o que meu código  
ganha com isso?!

Friday, July 1, 2011

legal, mas porque fazer isso se meu sistema já funciona?  
porque mexer em time que tá ganhando?

# facil de entender

Friday, July 1, 2011

o entendimento do resto da equipe é essencial para um projeto de sucesso.  
aumenta a produtividade

# facil de manter

Friday, July 1, 2011

manter um software é algo muito dificil, e se voce tem um codigo bagunçado e feio, se torna quase impossivel.



# flexível

Friday, July 1, 2011

tornando seu software flexível fica muito mais fácil responder a mudanças mais rápido.

# reuso

Friday, July 1, 2011

eliminando a duplicação de código, você escreve muito menos, e faz muito mais.  
isso é lucro \$\$

# show me the code



Friday, July 1, 2011

# nomes expressivos

```
def primes(n):  
    s=[1,]+range(4,n+1,2)+range(6,n+1,3)+range(10,n  
+1,5)+range(14,n+1,7)  
    for x in (2,3,5,7):  
        s+=range(x*2,n+1,x)  
    l=range(1,n+1)  
    s.sort()  
    primes=list(set(l)-set(s))  
    return primes
```

# e o teste?!

```
from unittest import TestCase
from primes import primes_to

class PrimesTest(TestCase):
    def test_primes_to_30(self):
        self.assertEqual([2, 3, 5, 7, 11, 13, 17, 19, 23, 29],
                          primes(30))
```

# funciona xD

```
$ speccloud teste.py
```

```
Primes
```


```
- primes to 30
```

```
-----
```

```
Ran 1 test in 0.001s
```

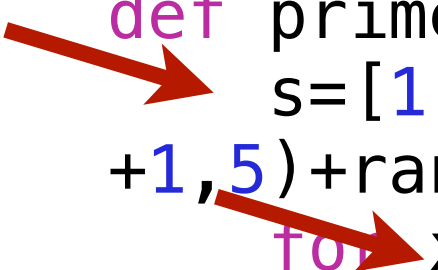
```
OK
```

# nomes expressivos



```
def primes(n):  
    s=[1,]+range(4,n+1,2)+range(6,n+1,3)+range(10,n  
+1,5)+range(14,n+1,7)  
    for x in (2,3,5,7):  
        s+=range(x*2,n+1,x)  
    l=range(1,n+1)  
    s.sort()  
    primes=list(set(l)-set(s))  
    return primes
```

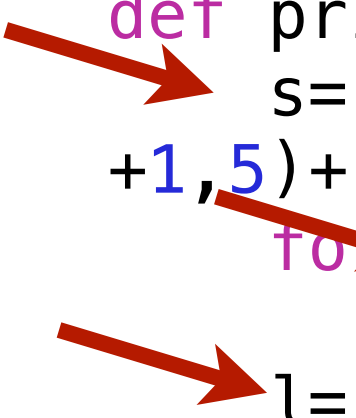
# nomes expressivos



```
def primes(n):  
    s=[1,]+range(4,n+1,2)+range(6,n+1,3)+range(10,n  
+1,5)+range(14,n+1,7)  
    for x in (2,3,5,7):  
        s+=range(x*2,n+1,x)  
    l=range(1,n+1)  
    s.sort()  
    primes=list(set(l)-set(s))  
    return primes
```

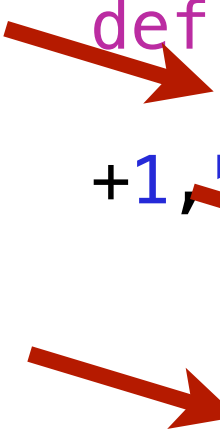


# nomes expressivos



```
def primes(n):  
    s=[1,]+range(4,n+1,2)+range(6,n+1,3)+range(10,n  
+1,5)+range(14,n+1,7)  
    for x in (2,3,5,7):  
        s+=range(x*2,n+1,x)  
    l=range(1,n+1)  
    s.sort()  
    primes=list(set(l)-set(s))  
    return primes
```

# nomes expressivos



```
def primes(n):  
    sieve=[1,]+range(4,n+1,2)+range(6,n+1,3)+range(10,n  
+1,5)+range(14,n+1,7)  
    for prime in (2,3,5,7):  
        sieve+=range(prime*2,n+1,prime)  
    limit=range(1,n+1)  
    sieve.sort()  
    primes=list(set(limit)-set(sieve))  
    return primes
```

# e ainda funciona xD

```
$ speccloud teste.py
```

```
Primes
```

```
- primes to 30
```

```
-----
```

```
Ran 1 test in 0.001s
```

```
OK
```

# pep8

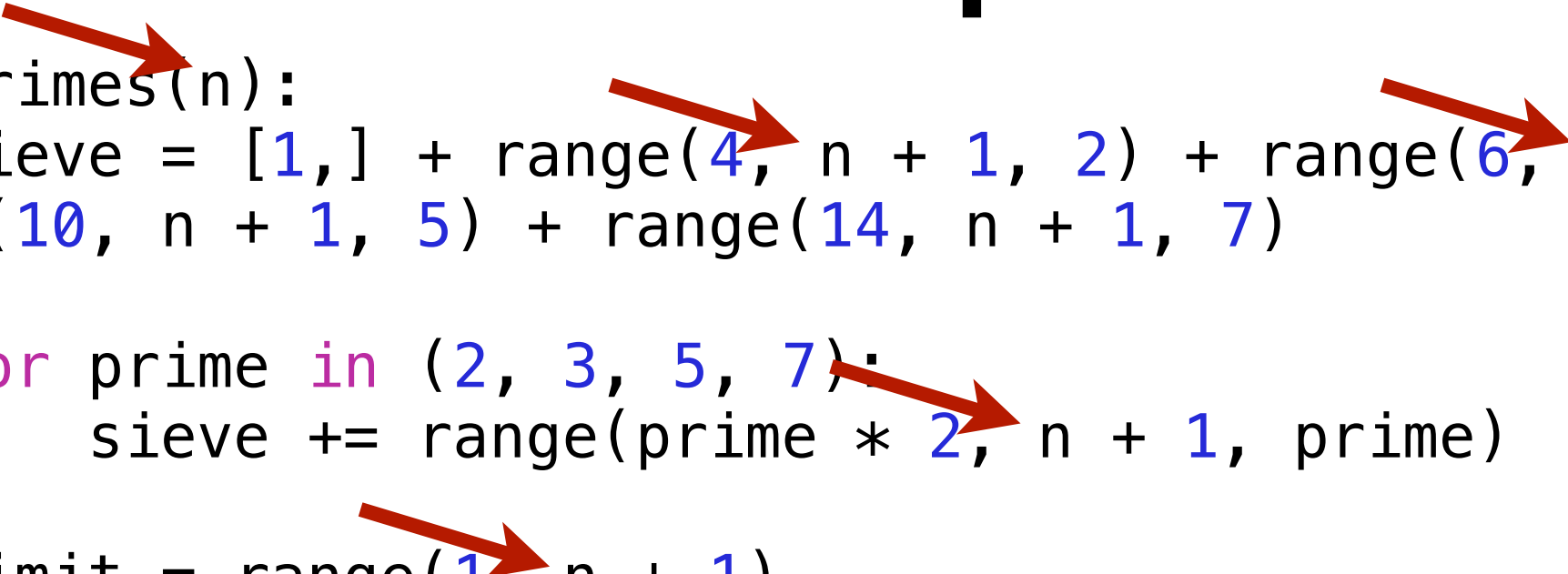
```
def primes(n):  
    sieve = [1,] + range(4, n + 1, 2) + range(6, n + 1, 3) +  
    range(10, n + 1, 5) + range(14, n + 1, 7)  
  
    for prime in (2, 3, 5, 7):  
        sieve += range(prime * 2, n + 1, prime)  
  
    limit = range(1, n + 1)  
    sieve.sort()  
    primes = list(set(limit) - set(sieve))  
    return primes
```

Friday, July 1, 2011

pep8 – <http://www.python.org/dev/peps/pep-0008/>  
guia de estilo para programadores python – Siga ele e seja feliz!!!

# nomes expressivos

```
def primes(n):  
    sieve = [1,] + range(4, n + 1, 2) + range(6, n + 1, 3) +  
    range(10, n + 1, 5) + range(14, n + 1, 7)  
  
    for prime in (2, 3, 5, 7):  
        sieve += range(prime * 2, n + 1, prime)  
  
    limit = range(1, n + 1)  
    sieve.sort()  
    primes = list(set(limit) - set(sieve))  
    return primes
```



# nomes expressivos

```
def primes(number):  
    sieve = [1,] + range(4, number + 1, 2) + range(6, number  
+ 1, 3) + range(10, number + 1, 5) + range(14, number + 1,  
7)  
  
    for prime in (2, 3, 5, 7):  
        sieve += range(prime * 2, number + 1, prime)  
  
    limit = range(1, number + 1)  
    sieve.sort()  
    primes = list(set(limit) - set(sieve))  
    return primes
```

# e continua funcionando

```
$ speccloud teste.py
```

```
Primes
```

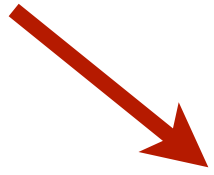
```
- primes to 30
```

```
-----
```

```
Ran 1 test in 0.001s
```

```
OK
```

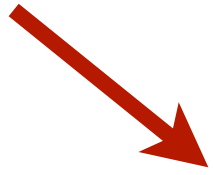
# nomes expressivos



```
def primes(number):  
    sieve = [1,] + range(4, number + 1, 2) + range(6, number  
+ 1, 3) + range(10, number + 1, 5) + range(14, number + 1,  
7)  
  
    for prime in (2, 3, 5, 7):  
        sieve += range(prime * 2, number + 1, prime)  
  
    limit = range(1, number + 1)  
    sieve.sort()  
    primes = list(set(limit) - set(sieve))  
    return primes
```



# nomes expressivos



```
def primes_to(number):  
    sieve = [1,] + range(4, number + 1, 2) + range(6, number +  
1, 3) + range(10, number + 1, 5) + range(14, number + 1, 7)  
  
    for prime in (2, 3, 5, 7):  
        sieve += range(prime * 2, number + 1, prime)  
  
    limit = range(1, number + 1)  
    sieve.sort()  
    primes = list(set(limit) - set(sieve))  
    return primes
```

# e o teste?

- runTest (ERROR)

```
=====
ERROR: Failure: ImportError (cannot import name primes)
-----
```

Traceback (most recent call last):

File "/Users/tarsis/.virtualenvs/conbertura-eventos/lib/python2.6/site-packages/nose/loader.py", line 390, in loadTestsFromName  
 addr.filename, addr.module)

File "/Users/tarsis/.virtualenvs/conbertura-eventos/lib/python2.6/site-packages/nose/importer.py", line 39, in importFromPath  
 return self.importFromDir(dir\_path, fqname)

File "/Users/tarsis/.virtualenvs/conbertura-eventos/lib/python2.6/site-packages/nose/importer.py", line 86, in importFromDir  
 mod = load\_module(part\_fqname, fh, filename, desc)

File "/Users/tarsis/Projects/PalestraFisl/teste.py", line 2, in <module>  
 from primes import primes

ImportError: cannot import name primes

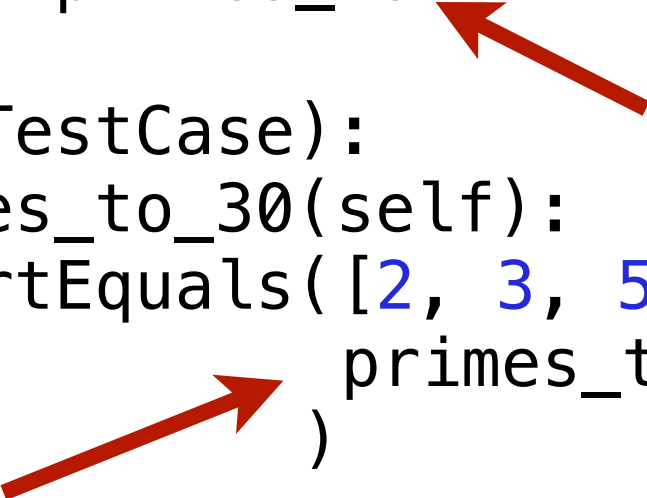
```
-----
Ran 1 test in 0.001s
```

FAILED (errors=1)

# ops...

```
from unittest import TestCase
from primes import primes_to

class PrimesTest(TestCase):
    def test_primes_to_30(self):
        self.assertEqual([2, 3, 5, 7, 11, 13, 17, 19, 23, 29],
                          primes_to(30))
```



# aew lol/

```
$ speccloud teste.py
```

```
Primes
```

```
- primes to 30
```

```
-----
```

```
Ran 1 test in 0.001s
```

```
OK
```

# Zen of Python

Beautiful is better than ugly.

```
def primes_to(number):  
    sieve = [1,] + range(4, number + 1, 2) + range(6, number +  
1, 3) + range(10, number + 1, 5) + range(14, number + 1, 7)  
  
    for prime in (2, 3, 5, 7):  
        sieve += range(prime * 2, number + 1, prime)  
  
    limit = range(1, number + 1)  
    sieve.sort()  
    primes = list(set(limit) - set(sieve))  
    return primes
```

Friday, July 1, 2011

zen of python – <http://www.python.org/dev/peps/pep-0020/>  
como tornar seu código Pythonico xD

# Zen of Python

Beautiful is better than ugly.

```
def is_prime(number):  
    if number <= 1:  
        return False  
  
    for i in xrange(2,number):  
        if number % i == 0:  
            return False  
    return True  
  
def primes_to(number):  
    primes = []  
  
    for num in range(number):  
        if is_prime(num):  
            primes.append(num)  
  
    return primes
```

# duvido que funcione!

```
$ speccloud teste.py
```

```
Primes
```

```
- primes to 30
```

```
-----
```

```
Ran 1 test in 0.001s
```

```
OK
```

ta lindo, mas e se eu  
quiser saber todos os  
primos até 2.000.000?!



# depuis de 13 minutos...

```
from unittest import TestCase
from primes import primes_to

class PrimesTest(TestCase):
    def test_primes_to_30(self):
        self.assertEqual([2, 3, 5, 7, 11, 13, 17, 19, 23, 29],
            primes_to(30))

    def test_primes_to_2000000(self):
        self.assertEqual(["?"], primes_to(2000000))
```

```
$ time speccloud teste.py
```

```
Primes
```

# depois de 13 minutos....

```
ux | grep specloud  
58285 0.0 0.2 2454904 7104 s003 S+ 5:11PM 0:00.13 /Users/tarsis/.virtualenvs/conbertura-eventos/bin/python /Users/  
d teste.py
```

```
$ time specloud teste.py
```

```
Primes
```

Friday, July 1, 2011

as vezes um algoritmo é suficiente para suprir sua necessidade, porem, quando necessario uma otimização o refactoring mostra seu valor.

**e agora?!**

# crivo de erastosthenes

Friday, July 1, 2011

crivo – [http://pt.wikipedia.org/wiki/Crivo\\_de\\_Erat%C3%B3stenes](http://pt.wikipedia.org/wiki/Crivo_de_Erat%C3%B3stenes)  
é um algoritmo para encontrar numeros primos ate um numero x

# crivo de erastostenes

```
from unittest import TestCase
```

```
from primes import primes_to
```

```
from primos_ate_2000000 import primos_ate_2000000
```

```
class PrimesTest(TestCase):
```

```
    def test_primes_to_30(self):
```

```
        self.assertEqual([2, 3, 5, 7, 11, 13, 17, 19, 23, 29],  
                           primes_to(30)  
                           )
```

```
    def test_primes_to_2000000(self):
```

```
        self.assertEqual(primos_ate_2000000,  
                           primes_to(2000000)  
                           )
```

# crivo de erastostenes

```
import math

def primes_to(number):
    sieve = [index for index in xrange(2, number + 1)]
    limit = int(math.sqrt(number))

    for index1 in xrange(0, limit):
        if not sieve[index1]:
            continue

        for index2 in xrange(index1 + 1, number - 1):
            if sieve[index2] and (not (sieve[index2] % sieve[index1])):
                sieve[index2] = 0

    return [index for index in xrange(2, number + 1)]
```

# agora sim!

```
$ time speccloud teste.py
```

```
Primes
```

```
- primes to 2000000
```

```
- primes to 30
```

```
-----
```

```
Ran 2 tests in 57.371s
```

```
OK
```

```
real    0m58.370s
```

```
user    0m50.646s
```

```
sys     0m0.636s
```

# acabou?!

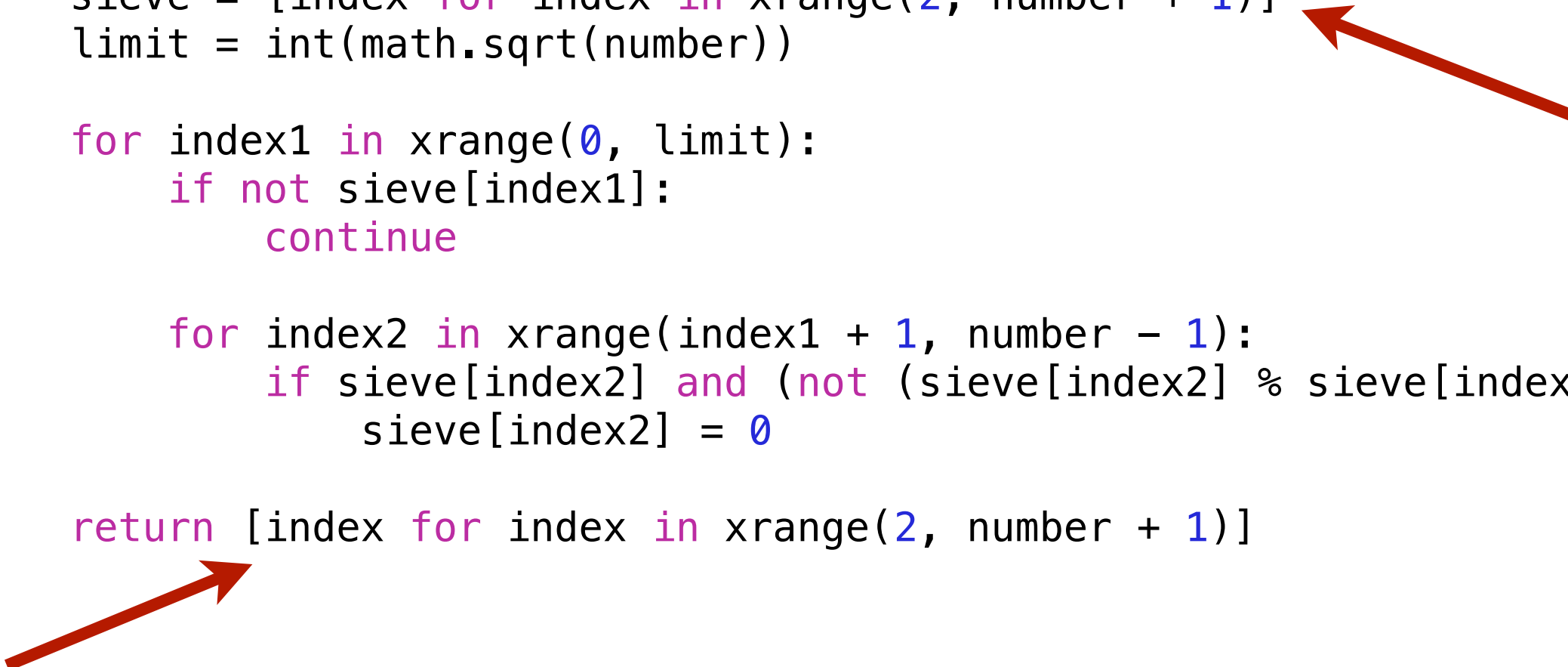
```
import math

def primes_to(number):
    sieve = [index for index in xrange(2, number + 1)]
    limit = int(math.sqrt(number))

    for index1 in xrange(0, limit):
        if not sieve[index1]:
            continue

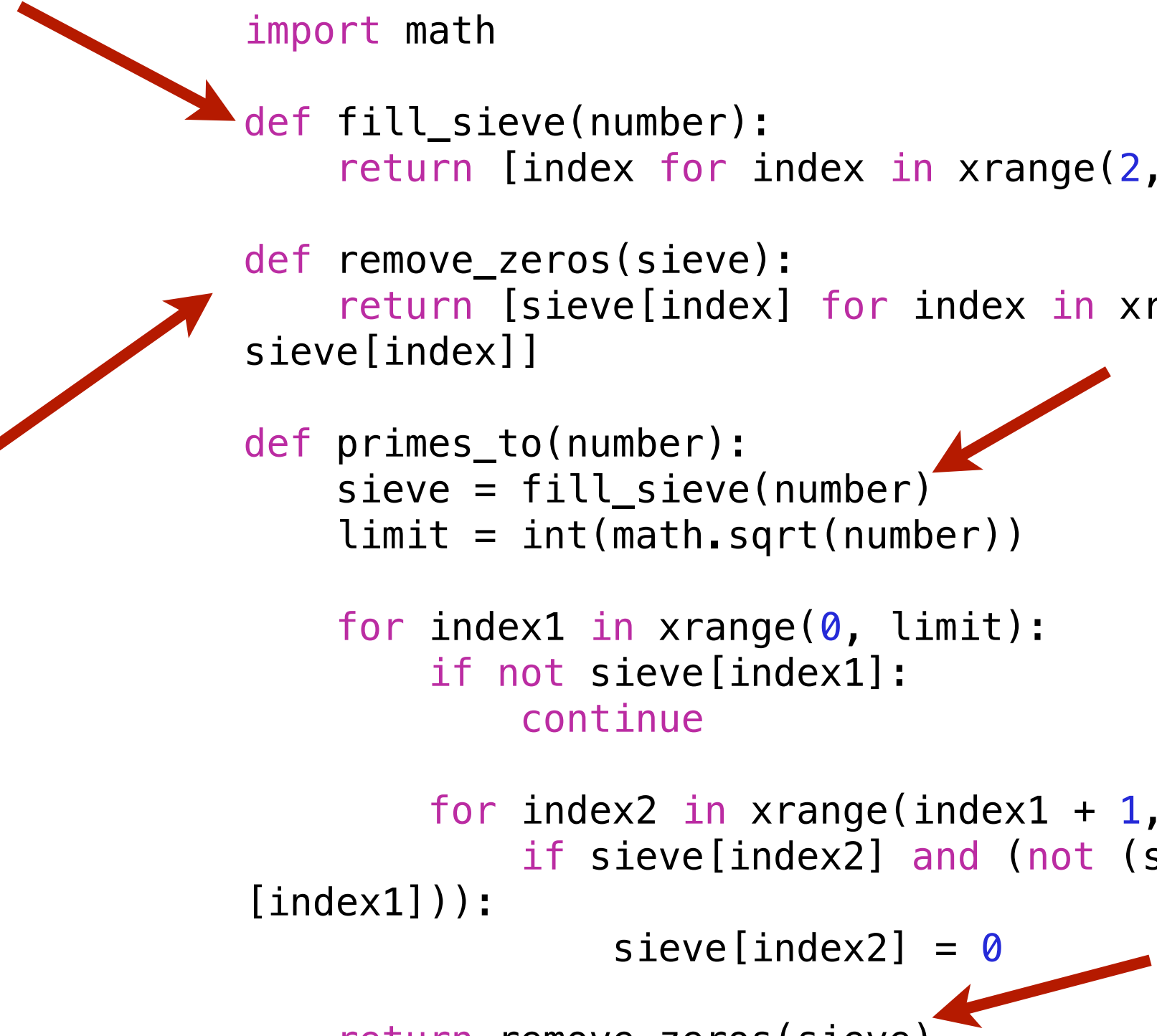
        for index2 in xrange(index1 + 1, number - 1):
            if sieve[index2] and (not (sieve[index2] % sieve[index1])):
                sieve[index2] = 0

    return [index for index in xrange(2, number + 1)]
```

Two red arrows are present. One arrow points from the right towards the line `sieve = [index for index in xrange(2, number + 1)]`. The other arrow points from the bottom-left towards the line `return [index for index in xrange(2, number + 1)]`.



# mais claro!



```
import math

def fill_sieve(number):
    return [index for index in xrange(2, number + 1)]

def remove_zeros(sieve):
    return [sieve[index] for index in xrange(len(sieve)) if
sieve[index]]

def primes_to(number):
    sieve = fill_sieve(number)
    limit = int(math.sqrt(number))

    for index1 in xrange(0, limit):
        if not sieve[index1]:
            continue

        for index2 in xrange(index1 + 1, number - 1):
            if sieve[index2] and (not (sieve[index2] % sieve
[index1])):
                sieve[index2] = 0

    return remove_zeros(sieve)
```

# mas...

```
import math

def fill_sieve(number):
    return [index for index in xrange(2, number + 1)]


def remove_zeros(sieve):
    return [sieve[index] for index in xrange(len(sieve)) if
sieve[index]]

def primes_to(number):
    sieve = fill_sieve(number)
    limit = int(math.sqrt(number))

    for index1 in xrange(0, limit):
        if not sieve[index1]:
            continue

        for index2 in xrange(index1 + 1, number - 1):
            if sieve[index2] and (not (sieve[index2] % sieve
[index1])):
                sieve[index2] = 0

    return remove_zeros(sieve)
```



# agora sim!

```
import math

def fill_sieve(number):
    return [index for index in xrange(2, number + 1)]

def remove_zeros(sieve):
    return [sieve[index] for index in xrange(len(sieve)) if
sieve[index]]

def sieve_of_erastostenes(number):
    sieve = fill_sieve(number)
    limit = int(math.sqrt(number))

    for index1 in xrange(0, limit):
        if not sieve[index1]:
            continue

        for index2 in xrange(index1 + 1, number - 1):
            if sieve[index2] and (not (sieve[index2] % sieve
[index1])):
                sieve[index2] = 0

    return sieve

def primes_to(number):
```

# e o teste!?

```
$ time speccloud teste.py
```

```
Primes
```

```
- primes to 2000000
```

```
- primes to 30
```

---

```
Ran 2 tests in 57.371s
```

```
OK
```

```
real    0m58.370s
```

```
user    0m50.646s
```

```
sys     0m0.636s
```



# agora é com voce!



Friday, July 1, 2011

estude, ganhe experiencia, refatore codigos open source!



# referencias

- Livros
  - - Refactoring - Martin Fowler
  - - Clean Code

# referencias

- slides
  - - <http://www.slideshare.net/caikesouza/refactoring-5412194>
  - - <http://www.slideshare.net/nashjain/refactoring-fest>

# referencias

- artigos
  - - <http://pragprog.com/the-pragmatic-programmer/extracts/software-entropy>
  - - <http://myadventuresincoding.wordpress.com/2009/03/24/working-with-legacy-code-lessons-learned/>

# referencias

- catalogo de refatorações
  - - <http://www.refactoring.com/catalog/index.html>
- site
  - - <http://refactormycode.com/>

# 2 palavras

*Away*