Desenvolvimento Web com Python & Django

Apresentação

- Osvaldo Santana Neto
  - Programador amador desde 1988
  - Programador profissional desde 1991
  - Programador Python desde 2000
  - Programador Django desde 2008

triveos

Introdução e História

Python

- Surgiu em 1989
- Criada por Guido van Rossum
- *Monty Python and the flying circus*
- Licença compatível com Software Livre
- Linguagem de altíssimo nível (VHLL)
- Tipagem dinâmica
- Multiparadigma (OO, funcional e procedural)
- Compilada + Interpretada

Versões de Python

- Python 2.7
  - Mantido até que a versão 3 esteja estável e com boa parte das bibliotecas portadas
- Python 3.2
  - Evolução da linguagem
  - Quebra compatibilidade retroativa
- Usaremos Python 2.7
  - O Django (e muitas outras bibliotecas) ainda não foi portado para o Python 3.0

Versões de Python

- Python 2.7
  - Mantido até que a versão 3 esteja estável e com boa parte das bibliotecas portadas
- Python 3.2
  - Evolução da linguagem
  - Quebra compatibilidade retroativa
- Usaremos Python 2.7
  - O Django (e muitas outras bibliotecas) ainda não foi portado para o Python 3.0

Trataremos as diferenças sempre que necessário!

Primeiro Programa

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# prog1.py - Primeiro programa
"""
Importa o módulo random e sorteia
um número inteiro entre 1 e 100
"""
import random
numero = random.randint(1, 100)

escolha, tentativas = 0, 0
while escolha != numero:
    escolha = input("Escolha um número entre 1 e 100:")
    tentativas += 1
    if escolha < numero:
        print u"O número", escolha, u"é menor que o sorteado."
    elif escolha > numero:
        print u"O número", escolha, u"é maior que o sorteado."
print u"Parabéns! Você acertou com", tentativas, "tentativas."
```

Primeiro Programa

Que interpretador será usado?

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# prog1.py - Primeiro programa
"""
Importa o módulo random e sorteia
um número inteiro entre 1 e 100
"""
import random
numero = random.randint(1, 100)

escolha, tentativas = 0, 0
while escolha != numero:
    escolha = input("Escolha um número entre 1 e 100:")
    tentativas += 1
    if escolha < numero:
        print u"O número", escolha, u"é menor que o sorteado."
    elif escolha > numero:
        print u"O número", escolha, u"é maior que o sorteado."
print u"Parabéns! Você acertou com", tentativas, "tentativas."
```

Primeiro Programa

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# prog1.py - Primeiro programa
"""
Importa o módulo random e sorteia
um número inteiro entre 1 e 100
"""
import random
numero = random.randint(1, 100)

escolha, tentativas = 0, 0
while escolha != numero:
    escolha = input("Escolha um número entre 1 e 100:")
    tentativas += 1
    if escolha < numero:
        print u"O número", escolha, u"é menor que o sorteado."
    elif escolha > numero:
        print u"O número", escolha, u"é maior que o sorteado."
print u"Parabéns! Você acertou com", tentativas, "tentativas."
```

Qual conjunto de caracteres (charset) será usado no arquivo?

Primeiro Programa

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# prog1.py - Primeiro programa
"""
Importa o módulo random e sorteia
um número inteiro entre 1 e 100
"""
import random
numero = random.randint(1, 100)

escolha, tentativas = 0, 0
while escolha != numero:
    escolha = input("Escolha um número entre 1 e 100:")
    tentativas += 1
    if escolha < numero:
        print u"O número", escolha, u"é menor que o sorteado."
    elif escolha > numero:
        print u"O número", escolha, u"é maior que o sorteado."
print u"Parabéns! Você acertou com", tentativas, "tentativas."
```

Usa-se "#" para comentários em Python. Também pode-se usar docstrings.

Primeiro Programa

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# prog1.py - Primeiro programa
"""
Importa o módulo random e sorteia
um número inteiro entre 1 e 100
"""
import random
numero = random.randint(1, 100)

escolha, tentativas = 0, 0
while escolha != numero:
    escolha = input("Escolha um número entre 1 e 100:")
    tentativas += 1
    if escolha < numero:
        print u"O número", escolha, u"é menor que o sorteado."
    elif escolha > numero:
        print u"O número", escolha, u"é maior que o sorteado."
print u"Parabéns! Você acertou com", tentativas, "tentativas."
```

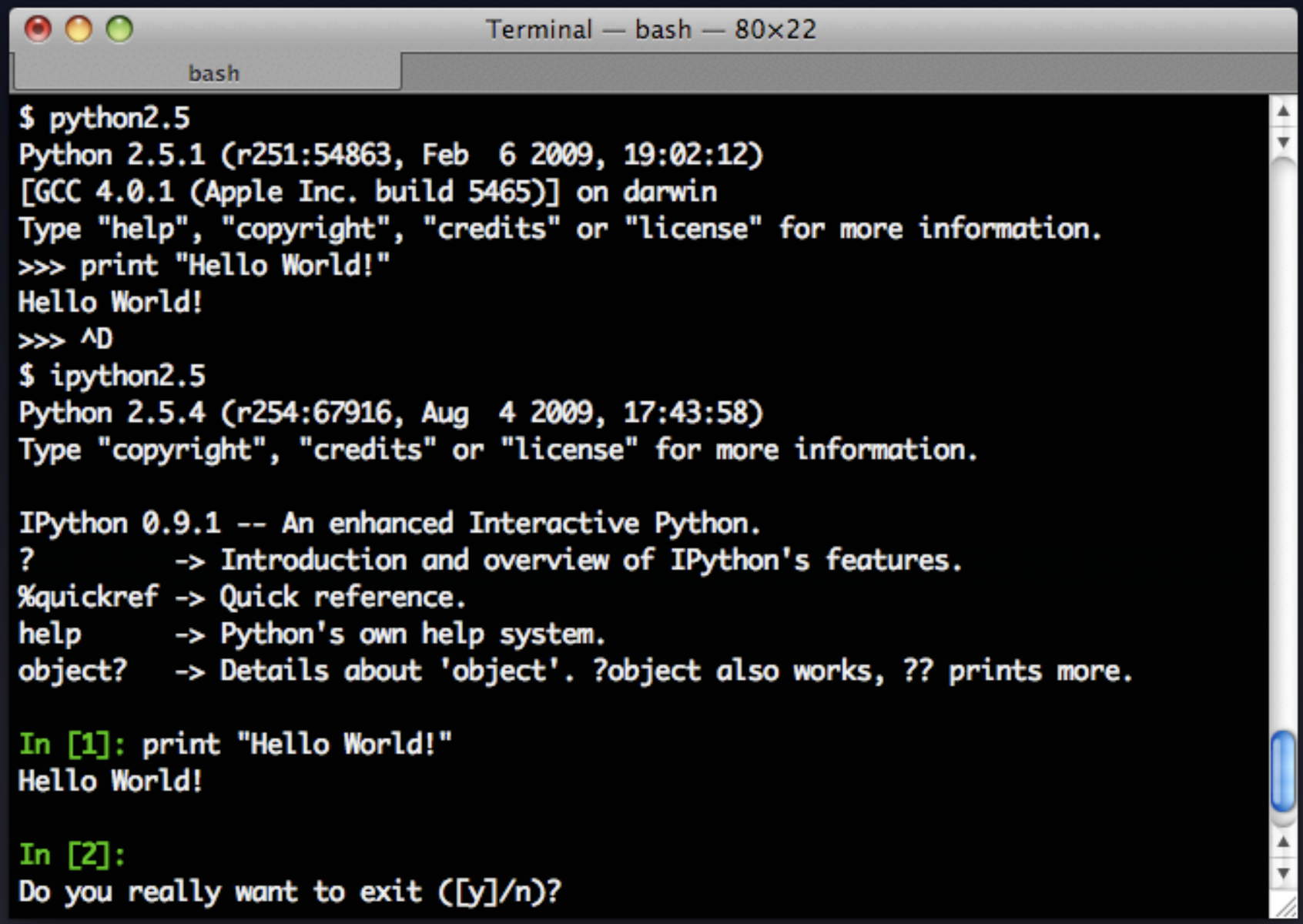
Python usa espaços para delimitar os blocos de código

Demonstração

Criando e executando nosso primeiro programa...

O interpretador

O interpretador



triveos

O intepretador

Parâmetros	
Parâmetro	Descrição
-h	Exibe uma relação de todos os parâmetros e uma breve descrição de funcionalidades.
-O / -OO	Liga a opção de otimização do código *bytecode* compilado. O arquivo gerado passa a ter a extensão `.pyo` no lugar de `.pyc`.
`-c cmds`	Executa os comandos `cmds` Python passados por parâmetro.
-u	Desliga o *buffer* de saída para o terminal. Isso faz com que todo o conteúdo impresso no terminal seja exibido imediatamente.
-m	Executa o `módulo` como um programa

O intepretador

Variáveis de ambiente	
Variável	Descrição
PYTHONHOME	Define o diretório inicial onde foi instalado o Python. Python utilizará essa variável para montar o caminho de diretórios que ele irá percorrer para encontrar a sua biblioteca padrão. O caminho de busca dessas bibliotecas, por padrão, será `PYTHONHOME/lib/python<versão>`.
PYTHONPATH	Define a ordem de procura de diretórios usados para importar os módulos Python. Os diretórios devem ser separados por ":". Diretórios incorretos ou que não existam serão ignorados.
PYTHONSTARTUP	Nome de um arquivo com código Python que será executado pelo interpretador antes de exibir o prompt do ambiente interativo. Neste arquivo você pode definir algumas configurações para o interpretador.

A linguagem

Características

- Um comando por linha
  - Usar ; para mais de uma linha
  - Usar \ para continuar em outra linha
- Bloco de comando por indentação
  - Não misture Tabs e espaços
- Comentários
  - Caracter # ou """*strings* multilinhas"""
- Diferencia maiúsculas de minúsculas

Números

- Números inteiros (int), inteiros longos (long), ponto flutuante (float) e complexos (complex)
- Operadores
  - Aritméticos: +, -, *, /, //, **, %
  - Bit a bit: &, |, ^, ~, >>, <<

```
>>> 2 ** 32, 2 + 5j
(4294967296L, (2+5j)
>>> 5.35 / 2.45
2.1836734693877546
>>> 5 + 2, 5 - 2, 5 * 2, 5 / 2
(7, 3, 10, 2)
>>> 5 / 2.0, 5 // 2.0
(2.5, 2.0)
>>> 5 ** 2, 5 % 2
(25, 1)
>>> 1 & 0, 1 & 1, 1 | 0, 1 | 1
(0, 1, 1, 1)
>>> 1 ^ 0, 1 ^ 1
(1, 0)
>>> ~1, ~0, 1 << 2, 32 >> 2
(-2, -1, 4, 8)
```

Números

- Números inteiros (int), inteiros longos (long), ponto flutuante (float) e complexos (complex)
- Operadores
  - Aritméticos: +, -, *, /, //, **, %
  - Bit a bit: &, |, ^, ~, >>, <<
- Divisão:
  - / » ponto flutuante
  - // » inteiro

3

```
>>> 2 ** 32, 2 + 5j
(4294967296L, (2+5j)
>>> 5.35 / 2.45
2.1836734693877546
>>> 5 + 2, 5 - 2, 5 * 2, 5 / 2
(7, 3, 10, 2)
>>> 5 / 2.0, 5 // 2.0
(2.5, 2.0)
>>> 5 ** 2, 5 % 2
(25, 1)
>>> 1 & 0, 1 & 1, 1 | 0, 1 | 1
(0, 1, 1, 1)
>>> 1 ^ 0, 1 ^ 1
(1, 0)
>>> ~1, ~0, 1 << 2, 32 >> 2
(-2, -1, 4, 8)
```

triveos

Lógica

- bool()
- True, False
- 0, 0.0, [], (), {}, "", set(), None, ... - Falso
- ==, !=, >, >=, <, <=, is, is not, and, or, in e not in
- Usar "is None" para comparações com None
- Prefira "if valor: ..." no lugar de "valor == True" ou "valor != 0".

```
>>> bool( 3 > 2 ), bool( 2 > 3)
(True, False)
>>> bool( 3 <= 2 ), bool( 2 < 3)
(False, True)
>>> bool( 3 == 2 ), bool( 2 == 3)
(False, False)
>>> bool( 0 ), not bool( None )
(False, True)
>>> bool( ["a"] ), not bool( "abc" )
(True, False)
>>> bool( ('a','b') ) and \
    not bool( "abc" )
False
>>> bool( ('a','b') ) or \
    not bool( "abc" )
True
```

Palavras reservadas

and	as	assert	break	class
continue	def	del	elif	else
except	exec	False	finally	for
global	if	import	in	is
lambda	None	nonlocal	not	or
pass	print	raise	return	True
try	while	with	yield	

3.0

triveos

Identificadores

- Diferença entre maiúscula e minúsculas
- Nome deve iniciar com letra ou "_"
- Restante do nome pode conter letras, números e "_"
- Não é permitido o uso de palavras reservadas mas quando necessário costuma-se usar um "_" no fim do nome (ex. "from_")

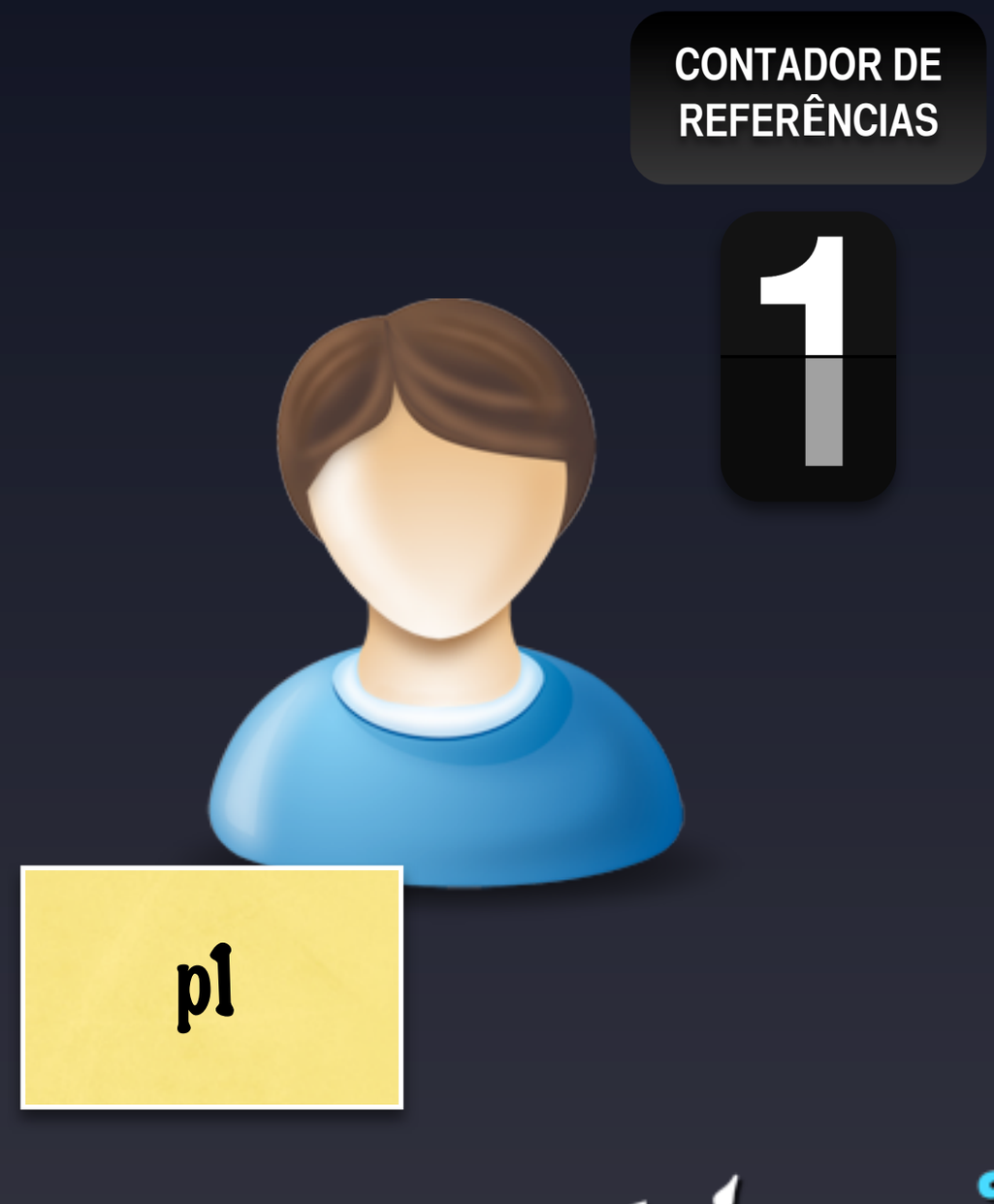
Atribuição

- Atribuição simples (=)
- Atribuição "aumentada": +=, -=, *=, /=, //=, **=, %=, |=, &=, ^=, <<= e >>=
  - Funcionam como nos operadores já explicados
- Atribuição por tupla:
  - a, b = b, a
  - (a, b) = (b, a)

```
>>> a = "a"
>>> b = "b"
>>> c, d = "cd"
>>> e, f = "e", "f"
>>> print a, b, c, d, e, f
a b c d e f
>>> e, f = f, e
>>> print a, b, c, d, e, f
a b c d f e
>>> a += b
>>> print a
ab
>>> a *= 5
>>> print a
ababababab
```

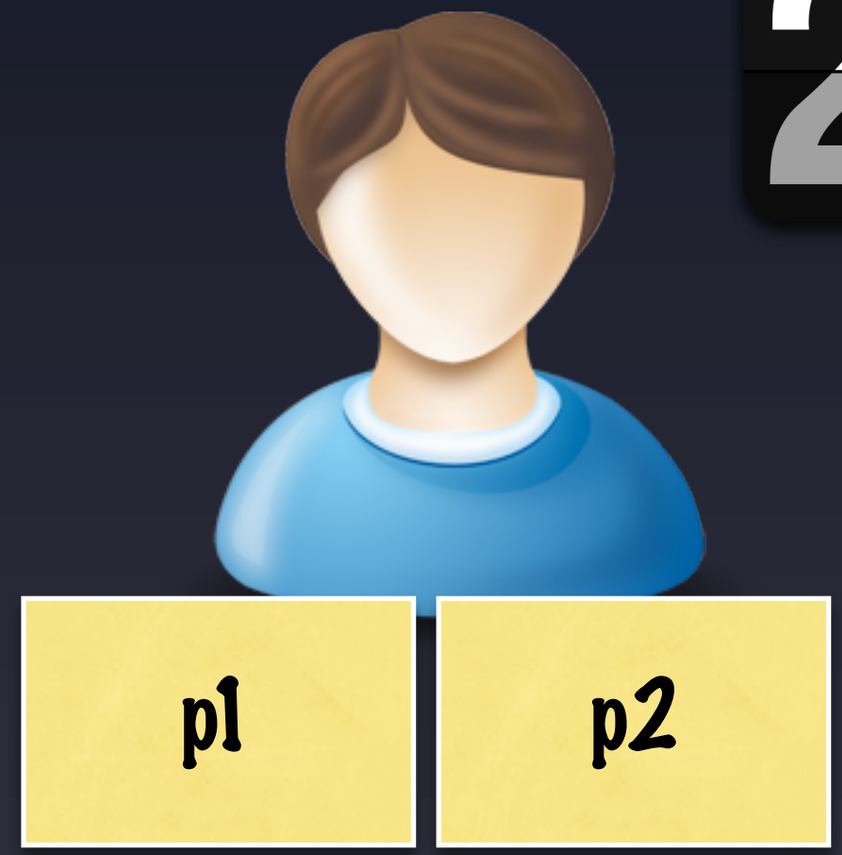
Referências

- p1 = PessoaFisica()



Referências

- p1 = PessoaFisica()
- p2 = p1



triveos

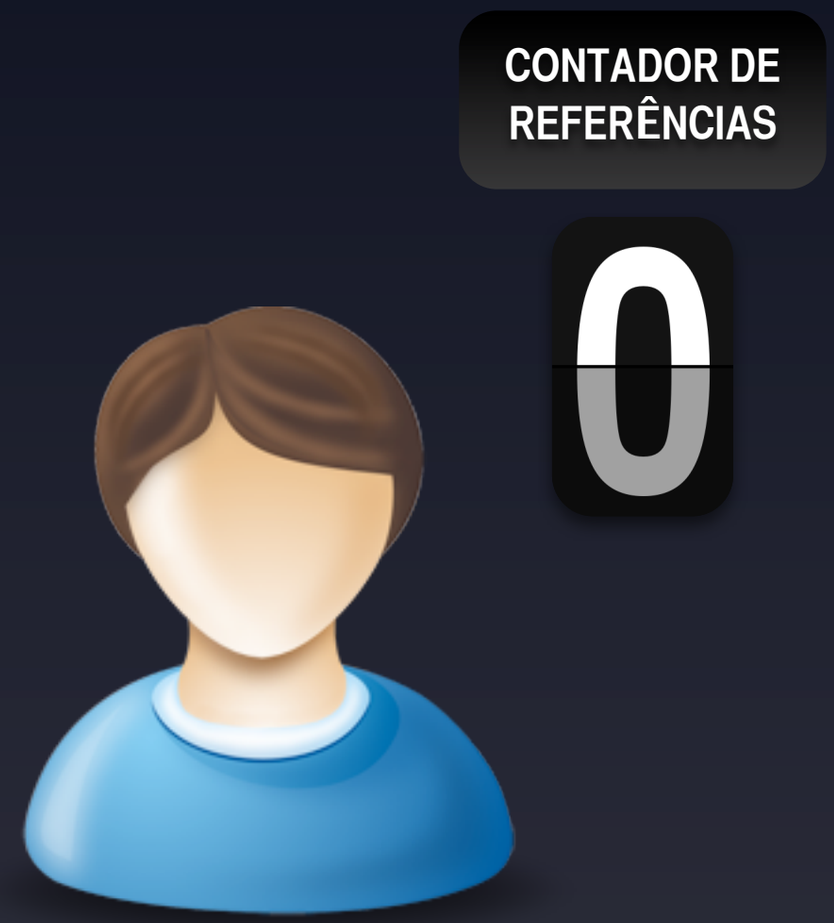
Referências

- p1 = PessoaFisica()
- p2 = p1
- del p1

triveos

Referências

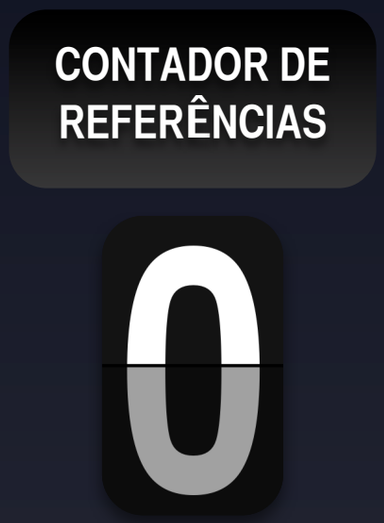
- p1 = PessoaFisica()
- p2 = p1
- del p1
- del p2



triveos

Referências

- p1 = PessoaFisica()
- p2 = p1
- del p1
- del p2
- *Destruição do objeto*



Comandos

Comando if/elif/else

- Comando de decisão.
- Executa o bloco de código em if ou elif caso a condição for verdadeira.
- Se nenhuma condição for verdadeira executa o bloco else
- Expressão if usada entre parênteses

```
a = input("A:")
b = input("B:")

if a > b:
    print "A é maior que B"
elif b > a:
    print "B é maior que A"
else:
    print "A e B são iguais"

print ("A é maior" if a > b \
        else "A não é maior")
```

Comando while

- Executa o bloco de código, em *loop*, enquanto a condição for verdadeira
- A cláusula 'else' pode ser usada e será executada caso o *loop* termine normalmente (sem break)

```
import sys, time

d = raw_input("Despertar (HH:MM): ")
while time.strftime("%H:%M") != d:
    try:
        print "\b\b\b\b\btick",
        sys.stdout.flush()
        time.sleep(0.5)
        print "\b\b\b\b\btack",
        sys.stdout.flush()
        time.sleep(0.5)
    except KeyboardInterrupt:
        break
else:
    print "\n\nTRIM!\a\a\a"
    sys.exit(0)
print "\n\nInterrompido!"
```

Comando for

- Itera sobre os elementos de um objeto (iterador, seqüência, ...)
- Pode ser usado na forma: 'for key, valor in dic.items(): ...' onde ele faz atribuição por tupla

```
import sys

for linha in sys.stdin:
    if not linha.strip(): continue
    if "FIM" in linha: break
    print "#", linha.rstrip()
else:
    print "# FIM DO ARQUIVO"
```

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

Comando continue

- Força a iteração de um *loop* (e a verificação de uma condição no caso do *loop* while)

```
import sys

for linha in sys.stdin:
    if not linha.strip(): continue
    if "FIM" in linha: break
    print "#", linha.rstrip()
else:
    print "# FIM DO ARQUIVO"
```

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

Comando break

- Interrompe o *loop*
- Um *loop* interrompido com break não executa os comandos da cláusula 'else'

```
import sys

for linha in sys.stdin:
    if not linha.strip(): continue
    if "FIM" in linha: break
    print "#", linha.rstrip()
else:
    print "# FIM DO ARQUIVO"
```

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

Comando print

- Imprime informações na tela
- Também pode enviar informações para um arquivo

```
import sys

s1, s2 = "String1", "String2"
obj = object()
f = open("/tmp/out.txt", "w")

print s1     # String1\n
print s1, s2 # String1 String2
print s1,    # String1␣
print s2     # String2
print obj    # <object .. 0xXX>
print >>f, s1, s2 # >out.txt
print >>sys.stderr, "ERRO!"
f.close()

$ cat /tmp/out.txt
String1 String2
```

Comando pass

- Não executa nada
- Necessário para definir blocos de código vazios já que Python não utiliza *braces* para delimitar blocos.

```
class Abstrata:
    def metodo_abstrato(self):
        pass

try:
    print 1 / 0
except ZeroDivisionError:
    pass # ignora exceção
```

Comando del

- Remove a referência a um objeto
- Remove elementos de *collections*
- Quando as referências chegam a zero o objeto entra na fila do *garbage collector* que irá liberar os recursos desse objeto

```
>>> a = [1,2,3]
>>> b = a
>>> a.append(4)
>>> a, b
([1, 2, 3, 4], [1, 2, 3, 4])
>>> del b
>>> b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'b' is not defined
>>> a
[1, 2, 3, 4]
>>> del a
>>> a, b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
```

Comando exec

- Executa o código objeto (compilado com a função *builtin* `compile()`) ou a *string* com código passada como parâmetro.
- Cuidado no uso desse comando com dados fornecidos pelo usuário

```
>>> exec "a = 1"
>>> a
1
>>> exec "if a: print 'verdadeiro'"
verdadeiro
```

Comando assert

- Verifica se uma condição é verdadeira. Caso contrário levanta uma AssertionError
- Removido nos módulos .pyo (executados com o parâmetro -O/-OO do interpretador python)

```
print "1 == 1?",
assert 1 == 1
print "sim."

try:
    print "2 == 1?",
    assert 2 == 1, "2 != 1"
except AssertionError, ex:
    print str(ex)

$ python prog_assert.py
1 == 1? sim.
2 == 1? 2 != 1
$ python -O prog_assert.py
1 == 1? sim.
2 == 1?
```

Tipos de dados

Tipos de dados

- Em Python, todos os tipos de dados são objetos
- Quando fazemos algo como: int("0") estamos, na verdade instanciando um objeto do tipo int() passando "0" para seu construtor.

Inteiros – int / long

```
>>> type(50)
<type 'int'>
>>> type(0xFF) # Hexadecimal
<type 'int'>
>>> type(034) # Octal
<type 'int'>
>>> 50, 0xFF, 034
(50, 255, 28)
>>> type(50L)
<type 'long'>
>>> type(0xFFFFFFFFF) # Hexadecimal
<type 'long'>
>>> 0xFFFFFFFFF # Hexadecimal
1099511627775L
>>> 2147483647 + 1
2147483648L
```

Inteiros – int / long

- Inteiros – int(x, base)
  - -2147483648 e 2147483647 (32b)
  - -NUMERÃO e +NUMERÃO (64b)

```
>>> type(50)
<type 'int'>
>>> type(0xFF) # Hexadecimal
<type 'int'>
>>> type(034) # Octal
<type 'int'>
>>> 50, 0xFF, 034
(50, 255, 28)
>>> type(50L)
<type 'long'>
>>> type(0xFFFFFFFFF) # Hexadecimal
<type 'long'>
>>> 0xFFFFFFFFF # Hexadecimal
1099511627775L
>>> 2147483647 + 1
2147483648L
```

Inteiros – int / long

- Inteiros – int(x, base)
  - -2147483648 e 2147483647 (32b)
  - -NUMERÃO e +NUMERÃO (64b)
- Longos – long(x, base)
  - Promoção automática de tipo
  - "L" identificando. Prefira o "L" maiúsculo
  - Memória é o limite

```
>>> type(50)
<type 'int'>
>>> type(0xFF) # Hexadecimal
<type 'int'>
>>> type(034) # Octal
<type 'int'>
>>> 50, 0xFF, 034
(50, 255, 28)
>>> type(50L)
<type 'long'>
>>> type(0xFFFFFFFFF) # Hexadecimal
<type 'long'>
>>> 0xFFFFFFFFF # Hexadecimal
1099511627775L
>>> 2147483647 + 1
2147483648L
```

Ponto flutuante – float

- Ponto flutuante – float(x)
  - 0.12, 1.25, 2e-5, ...

```
>>> type(1.5)
<type 'float'>
>>> type(7e-5)
<type 'float'>
>>> type(1e10)
<type 'float'>
>>> 7e-5
6.999999999999994e-05
>>> print 7e-5 - 7e-3
-0.00693
>>> 7e-5 - 7e-3
-0.006930000000000004
```

Complexos – complex

- Complexos – complex(i, j)
  - 2+2j, 5j

```
>>> type(5j)
<type 'complex'>
>>> type(2+5j)
<type 'complex'>
>>> (2+3j) * 5
(10+15j)
```

Booleanos – bool

- Booleano – bool(x)
- True e False
- Valores Falsos:
  - (), [], {}, set([]), None, 0, 0.0 e ""
  - __nonzero__() ou __len__() retornam 0

```
>>> type(True)
<type 'bool'>
>>> type(False)
<type 'bool'>
>>> int(True)
1
>>> int(False)
0
```

Seqüências

- São seqüências:
  - *Strings* (imutável)
  - Listas (mutável)
  - Tuplas (imutável)
- Indexáveis
- Iteráveis
- "Fatiáveis" (*slice*)
- *List comprehension*

Indexação e *Slicing*

- Indice inicial: 0 (zero)
- Indexação:
  - "abcde"[2] – 'c' / ['a', 'b', 'c', 'd'][-1] – 'd'
- *Slicing*
  - seq[início:fim:intervalo]
  - "abcde"[2:] – 'cde' / "abc"[:-1] – 'ab'
  - "abcde"[::2] – 'ace' / "abcde"[1:-1:2] – 'bd'
  - "abcde"[:] – 'abcde' (cópia 'rasa')

Iteração

- for c in "abcde": ... – percorre todos os caracteres
- for n in [1, 2, 3]: ... – percorre 1, 2 e 3
- for x in "": ... – não entra no *loop*

List Comprehension

- Simplifica *loops* que percorrem seqüências (conforme exemplo que imprime o quadrado dos números positivos)

```
seq = [ -1, 3, -5, 4, 8, 9 ]

# loop
res = []
for n in seq:
    if n > 0: res.append(n * n)
print res

# list comprehension
print [ x*x for x in seq if x > 0 ]
print [ x*x for x in seq ] # todos

# ['nome: fulano', 'idade: 35']
d = { 'nome': 'fulano',
      'idade': '35' }
print [ "%s: %s" % (k,v) for \
            k,v in d.items() ]
```

Strings – str / unicode

- *String* – str(x)
  - 'spam', "spam", """spam""" ou '''spam'''
- Unicode – unicode(x[, codificacao[, erros]])
  - u'eggs', u"eggs", u"""eggs""" ou u'''eggs'''
- *Regex strings* – r"(\w)"
- *docstrings*
- Seqüência de caracteres – Comporta-se como uma seqüência imutável
  - *Slice*, indexação, iterável

Strings – str / unicode

Caracteres especiais	
Caracter	**Descrição**
`\<newline>`	*Newline* será ignorado. Usado para continuar *string* em outra linha
`\\`	Caracter de contra-barra
`\'`	Apóstrofo
`\"`	Aspas
`\a`	Bell (aviso sonoro)
`\b`	Backspace (volta cursor)
`\e`	Caracter de escape (Esc)
`\0`	Caracter nulo. Não identifica término da *string* como em C
`\t`	Tab horizontal
`\r`	Retorno de carro (*Carriage Return*)
`\n`	Avanço de linha
`\0xx \N{nome} \uxxxx`	Especifica o caracter, em octal, por nome ou em hexadecimal.

Strings – str / unicode

Códigos de formatação	
`%[(chave)][atributos][largura][.precisão]código`	
Código	**Descrição**
%s	*String* (método __str__())
%c	Caracter
%d	Decimal inteiro (trunca números reais)
%i	Inteiro com sinal
%o	Inteiro em base octal
%x %X	Inteiro em base hexadecimal (letras em minúscula ou maiúscula)
%r	*String* via método __repr__()
%f %e %g	Notação científica: nunca, sempre, ou de acordo com expoente
%%	Caracter %

Strings – str / unicode

- Métodos úteis:
  - "a b c".split(' ') – ['a', 'b', 'c']
  - '\n'.join(['a', 'b', 'c']) – "a\nb\nc"
  - "xyz".strip("xz") – "y"
  - 'teste1'.startswith('teste') – True
  - 'func_test'.endswith('test') – True
  - '_' in 'func_test' – True
  - 'a'.find('b') – -1 / 'a'.index('b') – Exception
  - 'abracadabra'.count('a') – 5
  - 'a'.upper() – 'A' / 'A'.lower() – 'a'

Para listagem completa:
dir(str)

Strings – str / unicode

```
>>> "ola", u"ola"
('ola', u'ola')
>>> "Ola mundo"[5] # index
'u'
>>> "Ola mundo"[5:] # slice
'mundo'
>>> "Ola mundo"[2:5]
'a m'
>>> "Ola mundo"[:-1]
'Ola mund'
>>> "Ola mundo"[::2]
'Oamno'
>>> for c in "Spam":
...     print c * 2,
...
SS pp aa mm
```

```
>>> print "\tx"
        x
>>> print r"\tx"
\tx
>>> a = "Olá"
>>> print a
Olá
>>> b = a.decode("latin1")
>>> print b
OlÃ¡
>>> print b.encode("latin1")
Olá
>>> f = ru"C:\Diretório"
>>> print f
C:\Diretório
```

triveos

Strings – str / unicode

```
>>> print "Ola %s" % ("mundo",)
Ola mundo
>>> print """Spam: %(spam)d;
... Eggs: %(eggs)d""" % (\
...     {'spam': 1, 'eggs': 2})
Spam: 1; Eggs: 2
>>> "Dólar: %1.2f" % (2.1,)
'Dólar: 2.10'
>>> "Dólar: %1.2f" % (2.156,)
'Dólar: 2.16'
>>> "Desconto: %03d%%" % (35,)
Desconto: 035%
>>> print "|%d|%5d|%-5d|%05d|" %\
...         (3, 3, 3, 3)
|3|    3|3    |00003|
>>> print "%+d e %d" % (3, 3)
+3 e 3
```

```
>>> print "%g e %g" % (0.0005,
0.00005)
0.0005 e 5e-05
>>> print "%015.5f" % (5.015,)
000000005.01500
>>> a = "spam"
>>> print "%s" % a
spam

>>> def func(x):
...     """func(x) -> inteiro"""
...     return int(x)
...
>>> func.__doc__
'func(x) -> inteiro'
```

Strings – str / unicode

```
>>> "a\nb c\nd".split()
['a', 'b', 'c', 'd']
>>> "a; b; c; ".split("; ")
['a', 'b', 'c', '']
>>> "".split(), "abc".split()
([], ['abc'])
>>> ":".join([ 'a', 'b', 'c' ])
'a:b:c'
>>> "\t  abc  \t\n".strip()
'abc'
>>> "xyz".strip("xz")
'y'
>>> 'teste'.startswith('test')
True
>>> 'test' in 'um teste'
True
```

```
>>> 'abc'.find('z'), 'abc'.find('a')
(-1, 0)
>>> 'abc'.find('c')
2
>>> 'abc'.index('z')
ValueError: substring not found
>>> 'abc'.index('c')
2
>>> 'aba'.count('a')
2
>>> 'accbcc'.count('cc')
2
>>> 'abcde'.upper()
'ABCDE'
>>> 'ABCDE'.lower()
'abcde'
```

Listas – list

- Lista – list(x)
  - [1, 2, 3], list('seq')
- Mutável
  - lista[2] = 'spam'
  - lista + ['eggs']

Listas – list

- Métodos úteis:
  - `['a'].append('b')` – `['a', 'b']` (inline)
  - `['a'].extend(['b', 'c'])` – `['a','b','c']` (inline)
  - `['a'].insert(0, 'x')` – `['x', 'a']`
  - `['a', 'b'].pop()` – `'b'` (lista fica `['a']`)
  - `['b', 'a'].sort()` – `['a', 'b']` (inline)
  - `['b', 'a'].reverse()` – `['a', 'b']` (inline)
  - `in`, `.index()`, e `.count()` funcionam como em *String*

Para listagem completa:
`dir(list)`

Listas – list

```
>>> ['a', 'b']
['a', 'b']
>>> ['a', 'b'] + ['c', 'd']
['a', 'b', 'c', 'd']

>>> a = ['a', 'b', 'c', 'd']
>>> a
['a', 'b', 'c', 'd']

>>> a[0] = 'X'
>>> a
['X', 'b', 'c', 'd']

>>> a += "efg"
>>> a
['X', 'b', 'c', 'd', 'e', 'f', 'g']
```

```
>>> a + "ghi"
TypeError: can only concatenate list
           (not "str") to list

>>> ['a', 'b', 'c', 'd'][0]
'a'

>>> ['a', 'b', 'c', 'd'][0:2]
['a', 'b']
```

Listas – list

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
>>> a.extend([5, 6])
>>> a
[1, 2, 3, 4, 5, 6]
>>> a.insert(2, 2.5)
>>> a
[1, 2, 2.5, 3, 4, 5, 6]
>>> a.pop()
6
>>> a
[1, 2, 2.5, 3, 4, 5]
>>> a.reverse()
>>> a
[5, 4, 3, 2.5, 2, 1]
>>> a.sort()
>>> a
[1, 2, 2.5, 3, 4, 5]
>>> ['err', 'ok', 'err'].count('err')
2
```

Tuplas – tuple

- Tupla – tuple(x)
  - (1, 2, 3), tuple([1,2,3])
- Imutável
  - tupla[2] = 'spam' # Erro!
  - tupla + ['eggs'] # Erro!
- Consome menos recursos que list()

Dicionários – dict

- Dicionário – dict(x)
  - { 'chave1': 'valor1', 'chave2': 'valor2' }
  - dict(k1=v1, k2=v2)
- Mutável
  - dicionario['chave1'] = 'valor novo'
  - dicionario['chave nova'] = 1
- *Hashmap* (não preserva ordem das chaves)
- Python usa dicionários em toda sua implementação

Dicionários – dict

Para listagem completa:
`dir(dict)`

- Métodos úteis:
  - `{'a':1}.get('b', 0) – 0`
  - `{'a':1}.setdefault('b', 0) – 0 / d['b'] = 0`
  - `{'a': 1, 'b': 2}.items() – [('a', 1), ('b', 2)]`
  - `{'a': 1, 'b': 2}.keys() – ['a', 'b']`
  - `{'a': 1, 'b': 2}.values() – [1, 2]`
  - `{'a': 1}.update({'b': 2}) – {'a': 1, 'b': 2} (inline)`
  - `{'a': 1, 'b': 2}.pop('a') – 1 / del d['a']`
  - `'chave' in {'chave': 1} – True`

Dicionários – dict

```
>>> d = { "c1": "v1" }
>>> dic["c2"] = "v2"
>>> d
{'c1': 'v1', 'c2': 'v2'}
>>> d[1] = "chave numerica"
>>> d
{'c1': 'v1', 1: 'chave numerica',
'c2': 'v2'}
>>> tuplachave = (1, 2, 3)
>>> d[tuplachave] = "objeto chave"
>>> d
{'c1': 'v1', 1: 'chave numerica',
 'c2': 'v2',
 (1, 2, 3): 'objeto chave'}
>>> d.update({'c3':'v3', 'c4':'v4'})
>>> d
{1: 'chave numerica', 'c3': 'v3',
 'c2': 'v2', 'c1': 'v1', 'c4': 'v4',
 (1, 2, 3): 'objeto chave'}
>>> d.pop('c4')
'v4'
>>> d
{1: 'chave numerica', 'c3': 'v3',
 'c2': 'v2', 'c1': 'v1',
 (1, 2, 3): 'objeto chave'}
>>> d.items()
[(1, 'chave numerica'), ('c3', 'v3'),
 ('c2', 'v2'), ('c1', 'v1'),
 ((1, 2, 3), 'objeto chave')]
>>> d.keys()
[1, 'c3', 'c2', 'c1', (1, 2, 3)]
>>> d.values()
['chave numerica', 'v3', 'v2', 'v1',
 'objeto chave']
```

triveos

Dicionários – dict

```
>>> 'c1' in d, 'c4' in d
(True, False)
>>> d.get('c4', 'v4')
'v4'
>>> 'c4' in d
False
>>> d.setdefault('c4', 'v4')
'v4'
>>> d['c4']
'v4'
>>> d.setdefault('c4', 'X')
'v4'
>>> d['c4']
'v4'
```

Dicionários – dict

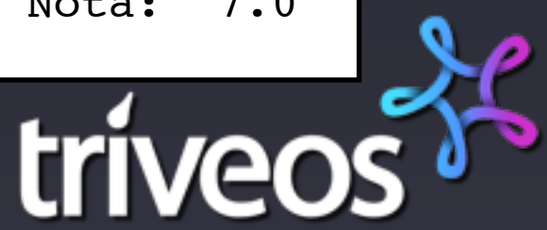
```
notas = {
   "Graham Chapman": 5.5,
   "John Cleese": 7.0,
   "Terry Gilliam": 4.5,
   "Terry Jones": 4.5,
   "Eric Idle": 10,
   "Michael Palin": 3.5,
}

print "Conteudo dicionario:", \
      notas

print

for aluno in notas:
   print aluno

print

for nota in notas.values():
   print nota

print

for aluno, nota in notas.items():
   print "Aluno: %-20s  Nota: %4.1f" % \
         (aluno, nota)
```

Dicionários – dict

```
Conteudo do dicionario: {'Terry Gilliam': 4.5, 'Eric Idle': 10, 'Graham Chapman': 5.5, 'Michael Palin': 3.5, 'Terry Jones': 4.5, 'John Cleese': 7.0}

Terry Gilliam
Eric Idle
Graham Chapman
Michael Palin
Terry Jones
John Cleese
```

```
4.5
10
5.5
3.5
4.5
7.0
```

```
Aluno: Terry Gilliam    Nota:  4.5
Aluno: Eric Idle        Nota: 10.0
Aluno: Graham Chapman   Nota:  5.5
Aluno: Michael Palin    Nota:  3.5
Aluno: Terry Jones      Nota:  4.5
Aluno: John Cleese      Nota:  7.0
```



Arquivos – file

- Arquivo – file(x) / open(x)
  - file(nome, modo, buffer)
  - file("/etc/passwd")
  - *Buffer* (f.flush() para esvaziar):
    - 0 – Sem *buffer*
    - 1 – Com *buffer*
    - +N – Tamanho do *buffer*
  - Iterável linha-a-linha (for linha in arquivo: ...)

Arquivos – file

Modos			
r			Leitura
w			Escrita (trunca)
a			Escrita (adiciona)
	b		Binário
	+		Leitura e Escrita
		U	Quebra de linha uni

Não pode ser usado com w ou +

Arquivos – file

Para listagem completa:
`dir(file)`

- Métodos úteis:
  - `f.close()` – Fecha o arquivo
  - `f.flush()` – Esvazia o buffer
  - `f.read(5)` – `'linha'`
  - `f.seek(0)` – Posiciona no início do arquivo
  - `f.tell()` – `0` (início do arquivo)
  - `f.write('linha final')` – Escreve 'linha final'
  - `f.readline()` – `'linha 1'`
  - `f.readlines()` – `['linha 1', 'linha 2', ...]`

Arquivos – file

Conteúdo do arquivo:

```
linha 1
linha 2
linha 3
```

```
>>> f = file("arq.txt", "a")
>>> f.write("linha 4\n")
>>> f.close()
>>> f = file("arq.txt")
>>> for l in f: print l,
...
linha 1
linha 2
linha 3
linha 4
>>> f.seek(0)
>>> f.read(5)
'linha'
>>> f.tell()
5L
>>> f.readlines()
[' 1\n', 'linha 2\n', 'linha 3\n',
'linha 4\n']
>>> f.seek(0)
>>> f.readline()
'linha 1\n'
>>> f.close()
```

Arquivos – file

```
>>> f = file("arq.txt", "a")
>>> f.write("linha 4\n")
>>> f.close()
>>> f = file("arq.txt")
>>> for l in f: print l,
...
linha 1
linha 2
linha 3
linha 4
>>> f.seek(0)
>>> f.read(5)
'linha'
>>> f.tell()
5L
>>> f.readlines()
[' 1\n', 'linha 2\n', 'linha 3\n',
'linha 4\n']
>>> f.seek(0)
>>> f.readline()
'linha 1\n'
>>> f.close()
```

Iteradores – iter

- Iterador – iter(x)
  - iter([1,2,3])
- *Generators* também implementam a interface de iteradores
- *Generator expressions* são iguais à uma *list comprehension* mas retorna um iterador no lugar de uma lista
  - Equivalente a list((x for x in seq))

Iteradores – iter

```
>>> a = [1, 2, 3]
>>> i = iter(a)
>>> type(i)
<type 'listiterator'>
>>> i.next(), i.next(), i.next()
(1, 2, 3)
>>> i.next()
Traceback (most recent call last):
StopIteration

>>> def f(n):
...     for i in range(n):
...         yield i
...
>>> r = f(3)
>>> type(r)
<type 'generator'>
>>> r.next(), r.next(), r.next()
(0, 1, 2)
>>> r.next()
Traceback (most recent call last):
StopIteration

>>> g = ( x for x in range(3) )
>>> type(g)
<type 'generator'>
>>> g.next(), g.next(), g.next()
(0, 1, 2)
>>> g.next()
Traceback (most recent call last):
StopIteration
```

Funções *builtin*

- `abs(x)` – Valor absoluto de um número
- `divmod(x, y)` – Divisão inteira e módulo
- `round(x[, d])` – float com x arredondado d casas decimais
- `all(iter)` – True se todos os elementos forem verdadeiros
- `any(iter)` – True se um dos elementos for verdadeiro
- `callable(obj)` – True se o objeto for 'chamável'. Ex. `obj()`
- `chr(ascii)` – caracter com código ASCII informado
- `unichr(código)` – caracter unicode com código informado
- `ord(caracter)` – retorna o código ASCII/Unicode do caracter
- `oct(i)` – retorna o número i no formato octal
- `dir([obj])` – mostra todos os identificadores membros de obj
- `execfile(...)` – executa o arquivo informado
- `hash(obj)` – retorna o *hash* do objeto

Funções *builtin*

- `id(obj)` – retorna o identificador do objeto
- `len(obj)` – retorna o tamanho do objeto
- `raw_input(p)` – solicita a entrada do usuário
- `input(p)` – solicita entrada do usuário e avalia com eval()
- `globals()` – dicionário com identificadores globais
- `locals()` – dicionário com identificadores locais
- `vars([obj])` – identificadores do objeto
- `max(s|x1, x2, ...)` – maior valor da seqüência
- `min(s|x1, x2, ...)` – menor valor da seqüência
- `range(s, e, i)` – retorna uma seqüência de s até e intervalo i
- `xrange(...)` – mesmo que `range()` mas retorna um iterador
- `enumerate(iter)` – retorna tuplas (indice, elemento)
- `open(...)` – atalho para file()

Funções

- Existem 2 tipos de funções:
  - Funções "convencionais" (def)
  - Funções anônimas (lambdas)
- Todo objeto que implementa o método __call__() é executável.

Chamando uma função

```
ret = funcao(
          param1,
          param2,
          param3=valor3,
          param4=valor4,
          *seq_parametros,
          **dic_parametros,
      )
```

Definindo uma função

- Função:

```
def funcao(param1, param2, param3="default",
           *params, **dicparams): ...
```

- Função anônima:

```
lambda p1, p2, p3=1, *ps, **dps: ...
```

- Sempre nessa ordem: parâmetros normais, opcionais (*default*), sequencia e dicionário.

Parâmetros

- `param` – nome da variável local que receberá o valor
- `param=valor` – mesmo que param mas assume valor default caso não informado
- `*params` – lista dos parâmetros adicionais
- `**dparams` – dicionário com parâmetros adicionais passados na forma param=valor

Demonstração

```
>>> def spam(x, y):
...     return x ** y
...
>>> spam(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: spam() takes exactly 2 arguments (1 given)

>>> def spam(x, y=2):
...     return x ** y
...
>>> spam(5)
25
>>> spam(5, 3)
125
```

Demonstração

```
>>> def spam(x, y=2, *args, **kw):
...     print x, y, args, kw
...
>>> spam(1)
1 2 () {}
>>> spam(1, 3)
1 3 () {}
>>> spam(1, 2, 3, 4, a1=5, a2=6)
1 2 (3, 4) {'a1': 5, 'a2': 6}

>>> def spam(x, *a, **k, y):
  File "<stdin>", line 1
    def spam(x, *a, **k, y):
                         ^
SyntaxError: invalid syntax
```

triveos

Demonstração

```
>>> def spam(x, y=2, z):
...     pass
...
  File "<stdin>", line 1
SyntaxError: non-default argument follows default argument

>>> def spam(x, y=2, *a, **kw):
...     print x, y, a, kw
...

>>> spam(1, z=3)
1 2 () {'z': 3}
>>> spam(1, z=3, 2)
  File "<stdin>", line 1
SyntaxError: no-keyword arg after keyword arg
```

Comando return

- Retorna o valor de uma função
- Pode retornar mais de um valor ao mesmo tempo (tupla)

```
def potencia(x, y=2): return x ** y
def sem_return(): pass

d = {'txt': "Arquivo texto",
     'html': "Arquivo HTML" }

m = {'txt': "text/plain",
     'html': "text/html" }

def arquivo(ext):
    return d[ext], m[ext]

potencia(5) # 25
sem_return() # None
arquivo('txt') # ('Arquivo texto',
               #  'text/plain')
```

Generators

Comando yield

- Espécie de iterador que permite o retorno à função geradora para que ela produza o novo elemento da iteração
- Pode-se enviar mensagens para a função geradora via método .send()
- Pode-se gerar uma exceção na função com o método .throw()

```
db = (((2009, 01, 05), "rec1"),
       :
      ((2009, 06, 05), "rec6"))

def busca(dt_ini, dt_fim):
    for rec in db:
        if rec[0] >= dt_ini and \
           rec[0] <= dt_fim:
            msg = yield rec[1]
            if msg:
                print rec[1], msg

b = busca((2009, 03, 01),
          (2009, 05, 01))
print b.next()
b.send("OK")
```

Escopo

- Busca-se os identificadores no escopo local, escopo das funções externas, escopo global, e então no escopo __bultins__.
- Atribuições, por padrão, criam/alteram identificadores do escopo local.
- O comando global faz com que os identificadores sejam criados/alterados no escopo global.

Comando global

- Diz que um identificador foi declarado no escopo global (e não no local)

```
def l():
    x = "L"
    print "l:", x # l(): L

def g():
    global x
    x = "G"
    print "g():", x # g(): G

x = "X"
print "X1:", x # X1: X
l()            # l(): L
print "X2:", x # X2: X
g()            # g(): G
print "X3:", x # X3: G
```

Closures

- Funções que constroem funções
- Linguagem deve ter funções como objetos de primeira classe
- Linguagem deve permitir acessar identificadores de funções externas

```
def fib(n):
    if n < 2: return 1
    return fib(n-1) + fib(n-2)

def memoize(fn):
    memo = {}
    def memoizer(key):
        if key not in memo:
            memo[key] = fn(key)
        return memo[key]
    return memoizer

print fib(35) # muuuito mais lento
fib = memoize(fib)
print fib(35)
```

Decorators

- Sintaxe nova para substituir as atribuições: x = dec(x)

Com *decorator*	Sem *decorator*

`@dec1`
`def f(x):`
`  return x * 2`
	`def f(x):`
	`  return x * 2`
	`f = dec1(f)`

`@dec1`
`@dec2`
`def f(x):`
`  return x * 2`
	`def f(x):`
	`  return x * 2`
	`f = dec1(dec2(f))`

`@dec1(arg)`
`def f(x):`
`  return x * 2`
	`def f(x):`
	`  return x * 2`
	`f = dec1(arg)(f)`

```
def memoize(fn):
    memo = {}
    def memoizer(key):
        if not key in memo:
            memo[key] = fn(key)
        return memo[key]
    return memoizer

@memoize
def fib(n):
    if n < 2: return 1
    return fib(n-1) + fib(n-2)

print fib(35)
```

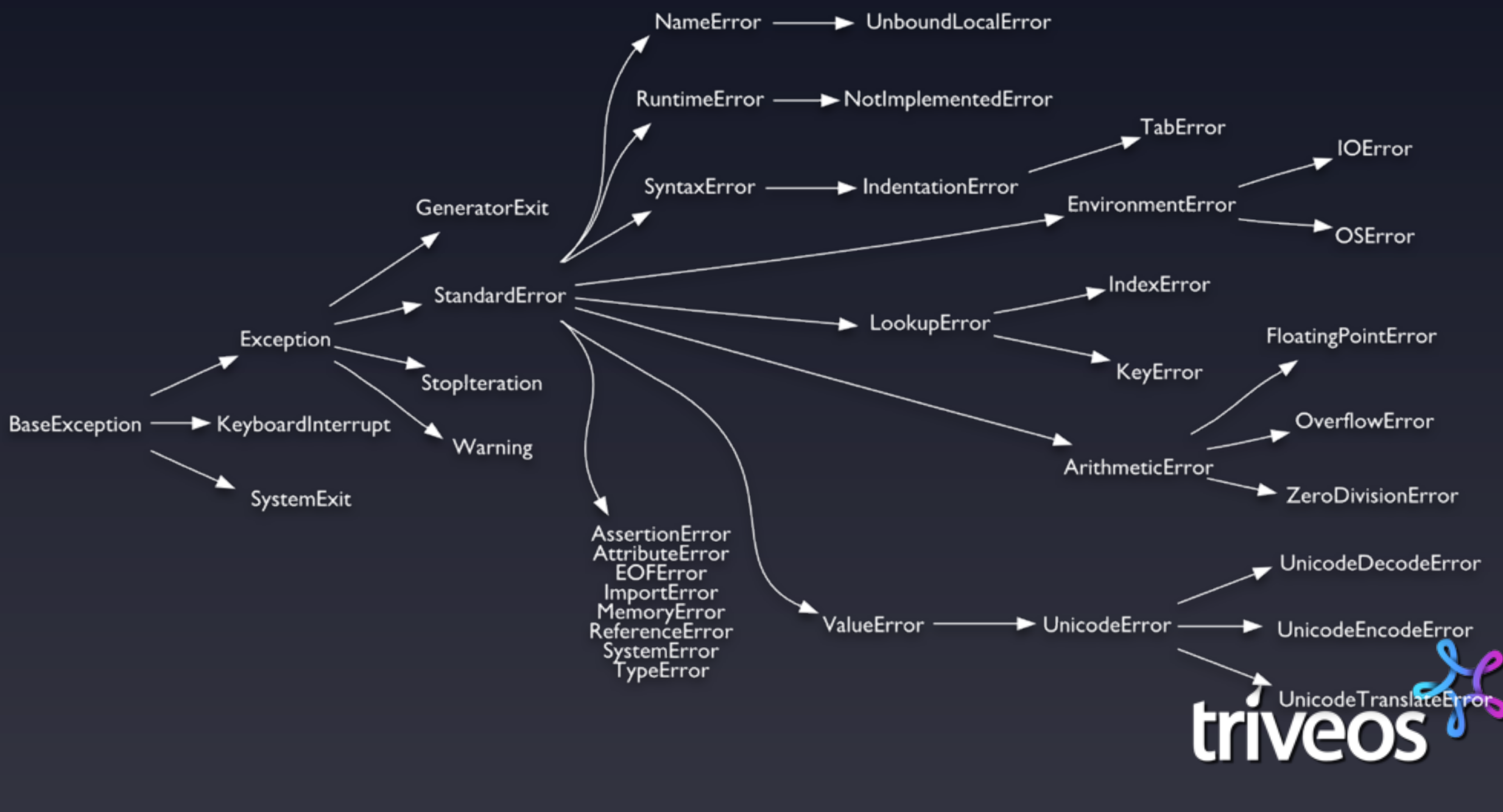
triveos

Exceções

Exceções

- Comandos:
  - try:/except:/else:/finally: – tratamento de exceções.
  - raise
- Exceções builtin (mais comuns):
  - `Exception`, `AttributeError`, `IOError`, `ImportError`, `IndexError`, `KeyError`, `KeyboardInterrupt`, `NotImplementedError`, `UnicodeError` e `ValueError`.

Exceções


NameError
UnboundLocalError
RuntimeError
NotImplementedError
SyntaxError
IndentationError
TabError
IOError
GeneratorExit
EnvironmentError
OSError
StandardError
IndexError
LookupError
Exception
FloatingPointError
KeyError
StopIteration
BaseException
KeyboardInterrupt
OverflowError
Warning
ArithmeticError
SystemExit
ZeroDivisionError
AssertionError
AttributeError
EOFError
ImportError
MemoryError
ReferenceError
SystemError
TypeError
UnicodeDecodeError
ValueError
UnicodeError
UnicodeEncodeError
UnicodeTranslateError
triveos

Exceções

```
class NotFound(Exception): pass
class InvalidKey(Exception): pass

def find(collection, elem):
   try:
      return collection[elem]
   except (KeyError, IndexError), ex:
      raise NotFound("%s not found (%s)" % \
                     (elem, ex))
   except TypeError:
      raise InvalidKey("%s is invalid" % \
                       (elem,))

a, b = [0], {}

try:
    find(a, 1) # IndexError
except Exception, ex:
    print "Exception: %s" % (ex,)
```

```
try:
    find(b, 'spam') # KeyError
except Exception, ex:
    print "Exception: %s" % (ex,)

try:
    find(a, "spam") # TypeError
except Exception, ex:
    print "Exception: %s" % (ex,)

try:
    find(b, 0) # No error
except Exception, ex:
    print "Exception: %s" % (ex,)
else:
    print "Element found."

try:
    find(a, "spam") # TypeError again
except Exception, ex:
    raise # re-raise exception
```

triveos

Exceções

```
def readfile(filename, bytes):
    try:
        f = file(filename)
    except IOError:
        return ""

    try:
        return f.read(bytes)
    except IOError:
        return ""
    finally:
        f.close()

print "Nao existe:", readfile("no.txt", 5)
print "Existe:", readfile("yes.txt", 5)
```

Programação Orientada a Objetos

Objetos

Instâncias

Classe

triveos

Classes

- Definida com o comando class
- Suporte a herança múltipla
- Método inicializador ("construtor"):
  def __init__(self, ...): ...
- Instanciar: identificador = Pessoa()

Classes

	Classe	Instância

Atributos	`class Pessoa(object):`
	`  atributo = 1`
		`class Pessoa(object):`
		`  def __init__(self, attr):`
		`  self.attr = attr`

Métodos	`class Pessoa(object):`
	`  @classmethod`
	`  def metodo(cls, arg):`
	`  pass`
		`class Pessoa(object):`
		`  def metodo(self, arg):`
		`  pass`

Método estático	`class Pessoa(object):`
	`  @staticmethod`
	`  def metodo(arg):`
	`  pass`	

Objetos

```
class Pessoa(object):
  def __init__(self, nome):
    self.nome = nome

  def valida(self):
    raise NotImplementedError()


class PF(Pessoa):
  tabela_db = "pessoa_fisica"
  def __init__(self, nome, cpf):
    super(PF, \
          self).__init__(nome)
    self.cpf = ''.join(\
      d for d in cpf if d.isdigit()
    )
  def valida(self):
    return len(self.cpf) == 11
```

```
class PJ(Pessoa):
  tabela_db = "pessoa_juridica"
  def __init__(self, nome, cnpj):
    super(PJ, \
          self).__init__(nome)
    self.cnpj = ''.join(\
      d for d in cnpj if d.isdigit()
    )
  def valida(self):
    return len(self.cnpj) == 14

f = PF("Fulano", "123.456.789-01")
j = PJ("ACME", "12345678/0001-90")

print f.nome, f.cpf, f.valida()
print j.nome, j.cnpj, j.valida()
```

Segurança

- Todos os métodos e atributos são públicos
- Por convenção os métodos com nomes iniciados por "_" são privados
- Adicionar "__" (duplo) renomeia o identificador para "_Classe__metodo"

Propriedades

- Não é necessário *getters* e *setters*
- Acesse os atributos diretamente
- Para validar as entradas ou calcular as saídas transforme-o em uma propriedade
- Use a função property()

Propriedades

```
class Forno(object):
    def __init__(self):
        self.temp = 0

forno = Forno()
print forno.temp # 0

forno.temp = 120
print forno.temp # 120

forno.temp = 300 # FORNO DERRETIDO!
```

Propriedades

```
class Forno(object):
    def __init__(self):
        self._temp = 0

    def _get_temp(self):
        return self._temp

    def _set_temp(self, temp):
        if temp > 250:
            raise ValueError(\
                  "Maxima: 250")
        self._temp = temp

    temp = property(_get_temp,
                    _set_temp)
```

```
forno = Forno()
print forno.temp # 0

forno.temp = 120
print forno.temp # 120

forno.temp = 300 # ValueError
```

Módulos, pacotes e bibliotecas

Pacotes e Módulos

- Importar um módulo ou pacote:
  - import pacote.modulo [as nome]
    - Importa para o *namespace* do módulo
  - from pacote.modulo import ident|*
    - Importa para o *namespace* corrente
- Módulo é "executado" ao ser importado
- Se necessário módulo é recompilado para *bytecode* (.pyc/.pyo)

Pacotes e Módulos

- Pacotes: diretórios com arquivo __init__.py
- Módulo:
  - .py – módulo fonte
  - .pyc / .pyo – *bytecode* (com e sem *asserts*)
  - .pyd / .so – extensão em C para Windows e Unix/Linux
- Módulos e pacotes são procurados no PYTHONPATH:
  - import sys; print sys.path

Pacotes e Módulos

- Atributos do módulo:
  - __name__ – nome do módulo ou __main__ se for executado diretamente
  - __file__ – caminho completo para o arquivo importado
- Para proteger código da execução durante o import:
  - if __name__ == "__main__": ...

Biblioteca padrão

anydbm	bz2	calendar	cgi	CGIHTTPServer	cgitb
cmd	codecs	collections	ConfigParser	copy	csv
ctypes	datetime	decimal	distutils	doctest	DocXMLRPCServer
email	ftplib	gc	getpass	glob	gzip
hashlib	htmlentitydefs	HTMLParser	httplib	imaplib	locale/gettext
logging	mailbox	math	mmap	optparse	os
os.path	pickle/cPickle	platform	poplib	pprint	Queue
random	re	shelve	shlex	shutil	signal
SimpleHTTPServer	SimpleXMLRPCServer	smtpd	smtplib	socket	SocketServer
sqlite3	ssl	stat	string	StringIO/cStringIO	struct
subprocess	sys	tarfile	tempfile	thread	threading
time	traceback	unittest	urllib	urllib2	urlparse
uuid	weakref	webbrowser	wsgiref	xml.dom.minidom	xml.etree
xml.sax	xmlrpclib	zipfile	zlib		

PIP

- Utilitário de instalação de bibliotecas
- Baixa e instala pacotes do PyPI (Python Package Index)
- Comando: pip install nome_do_pacote
  - Necessário permissão de administrador
- Não vem instalado por padrão, necessário baixar em:
  - http://pypi.python.org/pypi/pip

Virtualenv

- Cria ambientes Python isolados
- Permite modificação de um ambiente sem causar interferência em outros
- Instalar virtualenv
  - sudo pip install virtualenv
- Criar um ambiente virtual
  - virtualenv nome_do_ambiente
- Ativando um ambiente
  - source nome_do_ambiente/bin/activate

Django

Django

- Criado por Jacob Kaplan-Moss, Adrian Holovaty e Simon Willison em 2003
- Criado como um CMS para a World Online (divisão Web do Lawrence Journal-World em Lawrence, Kansas)
- Tornado opensource logo após
- Framework Web ágil
- DRY - *Don't Repeat Yourself*

Componentes

- *Object-relation Mapper* – mapeador objeto-relacional
- *Template System* – linguagem de templates
- *URL dispatcher* – processador de URLs
- *Admin Interface* – Interface de administração
- Internacionalização – regionalização
- Outros: gerador de formulário, serializador de dados, autenticação e segurança, ...

MVC vs. MTV

- *Pattern* MVC
  - *Model* – lógica de negócios
  - *View* – camada(s) de visualização
  - *Controller* – manipulação de Model
- *Pattern* MTV
  - *Model* – mesmo que no MVC
  - *Template* – mesmo que a *View* do MVC
  - *View* – mesmo que a *Controller* do MVC
- A essência continua a mesma

Instalação

- Criar ambiente com virtualenv e ativá-lo
- Instalando "na mão":
  - Baixar do site:
    - http://djangoproject.com/download/
  - Descompactar
  - No diretório do Django rodar:
    - python setup.py install
- Ou usando o setuptools com o comando:
  - pip install django

Instalação

```
$ su -
# apt-get install python-pip
# pip install virtualenv
# exit
$ virtualenv aula --no-site-packages
$ cd aula
aula $ source bin/activate
(aula) aula$ pip install django
```

O Projeto de aula

- Agenda de eventos
  - Cadastro de evento com data/hora e lugar
  - Cadastro dos usuários que participarão do evento
  - Sistema de autenticação de usuário

Iniciando o projeto

- Utilitário django-admin.py
  - Diversos comandos de gerenciamento
  - django-admin.py help para listá-los
  - django-admin.py help <comando> para ajuda detalhada de cada comando
- Criando um projeto 'gerenciador':
  - django-admin.py startproject gerenciador

Iniciando o projeto

```
$ cd aula
aula$ source bin/activate

(aula) aula$ django-admin.py startproject gerenciador
(aula) aula$ cd gerenciador
(aula) aula/gerenciador$ ls
__init__.py  manage.py    settings.py  urls.py
(aula) aula/gerenciador$ chmod +x manage.py
(aula) aula/gerenciador$ ls
__init__.py  manage.py*   settings.py  urls.py
```

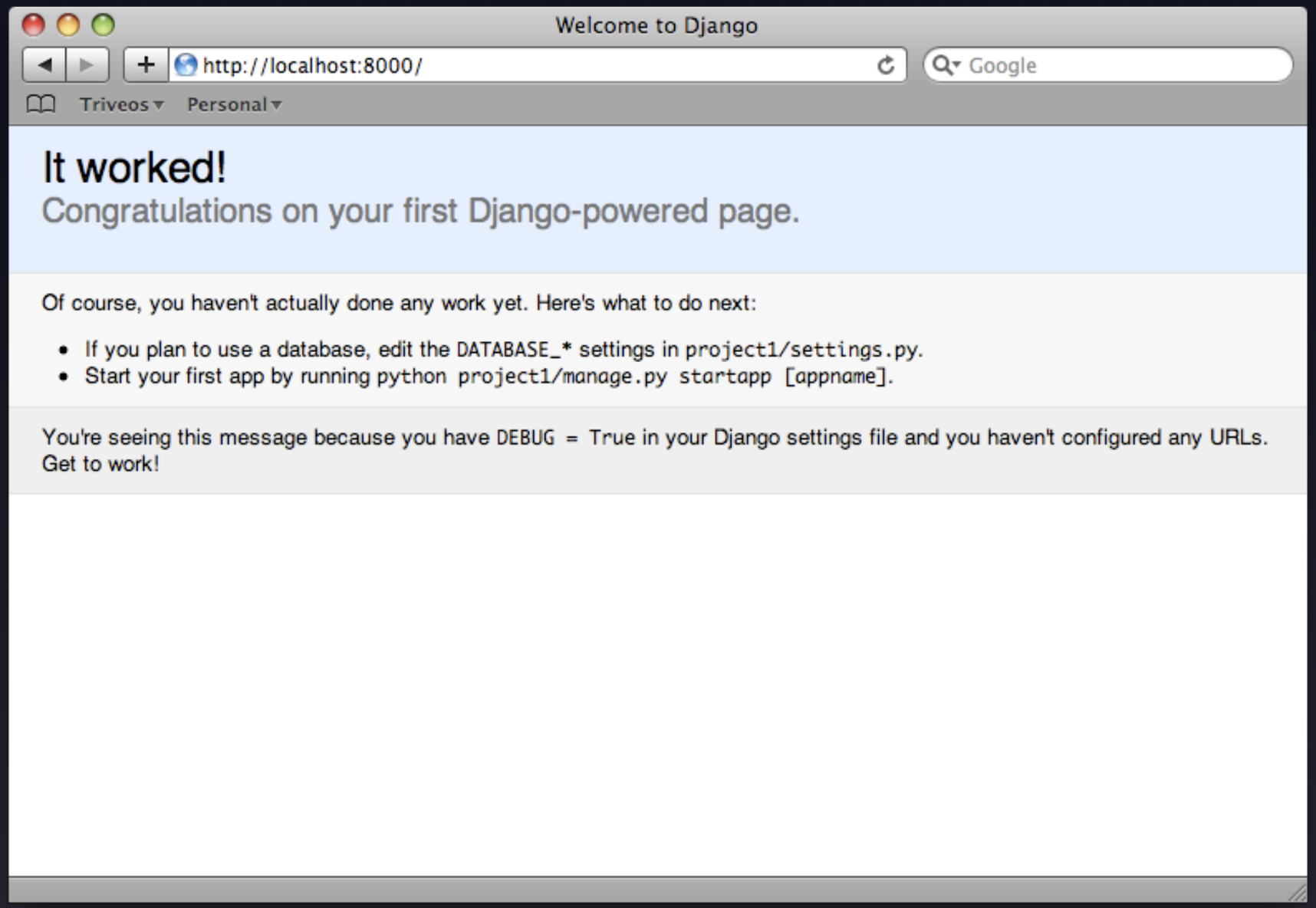
Os arquivos do projeto

- __init__.py – arquivo vazio
- manage.py – utilitário parecido com o django-admin.py. Colocado aqui por comodidade.
  - Dica: chmod +x manage.py
- settings.py – arquivo de configuração do projeto
- urls.py – definições de URLs do projeto

Testando

- Django vem com um servidor Web próprio para ser usado durante o desenvolvimento
  - Recarga automática de módulos
  - NÃO usar este servidor em produção
- O comando para iniciar o servidor é:
  - ./manage.py runserver
- Aponte o navegador para o endereço:
  - http://localhost:8000/

Testando



triveos

Gerenciando

manage.py	
Comando	Descrição
`syncdb`	Cria tabelas no banco de dados
`dumpdata --format F [aplicação]`	Descarrega dados da aplicação em XML ou JSON
`loaddata fixture`	Carrega dados no banco de dados
`shell`	Interpretador Python com modelos de dados
`createsuperuser --username --email`	Cria um usuário administrador
`runserver [endereço:porta]`	Inicia o servidor Web de desenvolvimento
`startapp [aplicação]`	Cria uma nova aplicação no projeto

Aplicações

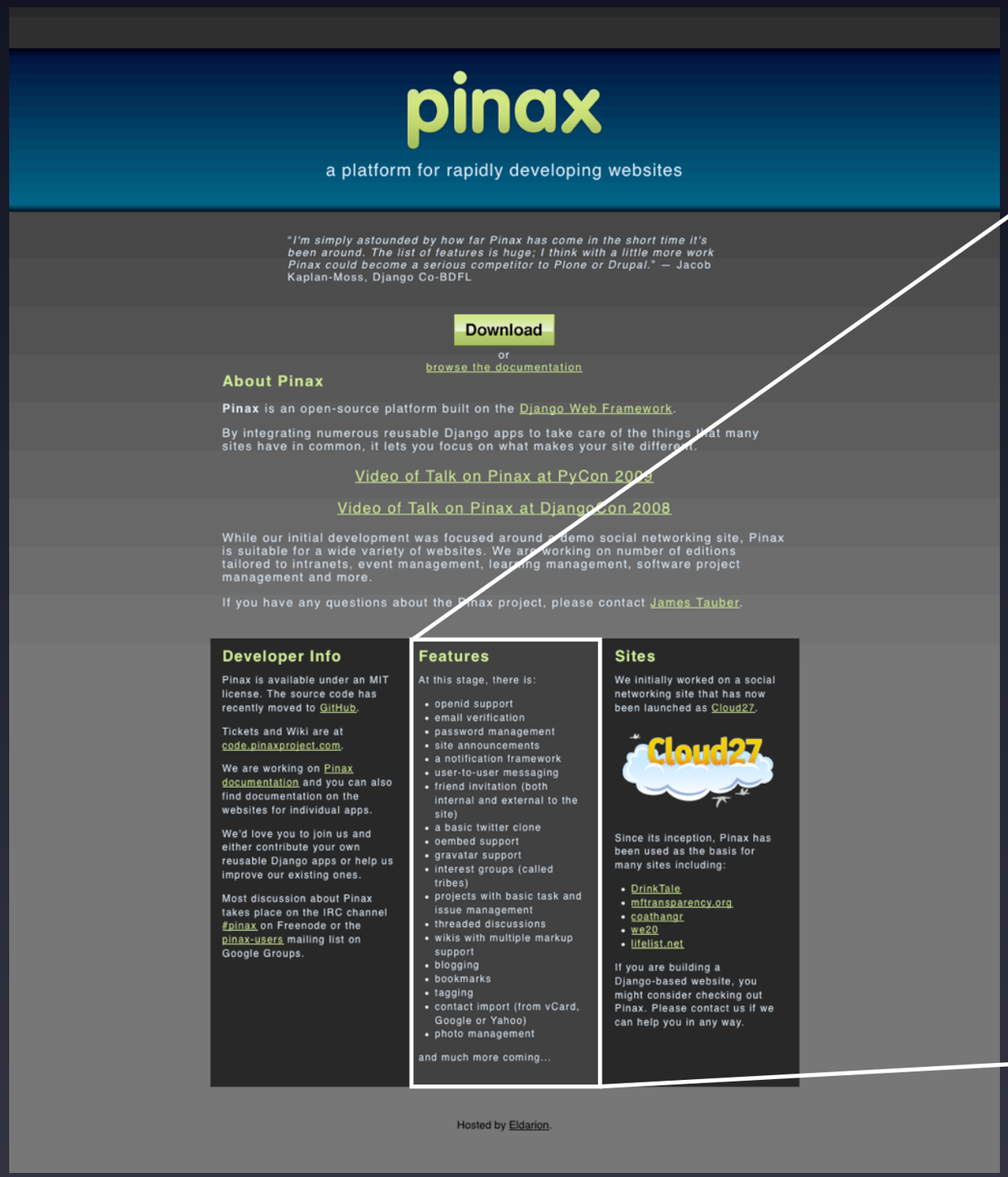
- Aplicações Django são módulos com funcionalidades que podem ser reaproveitadas em vários projetos
- Exemplos:
  - Sistema de comentários
  - blog
  - tags
  - autenticações alternativas
  - etc.

triveos

Aplicações

triveos

Aplicações



Features

At this stage, there is:

- openid support
- email verification
- password management
- site announcements
- a notification framework
- user-to-user messaging
- friend invitation (both internal and external to the site)
- a basic twitter clone
- oembed support
- gravatar support
- interest groups (called tribes)
- projects with basic task and issue management
- threaded discussions
- wikis with multiple markup support
- blogging
- bookmarks
- tagging
- contact import (from vCard, Google or Yahoo)
- photo management

and much more coming...

triveos

Aplicações

```
(aula) aula/gerenciador$ ./manage.py startapp agenda
(aula) aula/gerenciador$ ls
__init__.py  agenda       manage.py*   settings.py  urls.py
(aula) aula/gerenciador$ ls agenda
__init__.py models.py   tests.py    views.py
```

Dizendo "Olá"

Arquivo:
`urls.py`

```
from django.conf.urls.defaults import *
# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()
urlpatterns = patterns('',
    # Example:
    # (r'^gerenciador/', include('gerenciador.foo.urls')),

    (r'^$', 'agenda.views.index'),

    # Uncomment the admin/doc line below and add 'django.contrib.admindocs'
    # to INSTALLED_APPS to enable admin documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    # (r'^admin/', include(admin.site.urls)),
)
```

triveos

Dizendo "Olá"

Arquivo:
agenda/
views.py

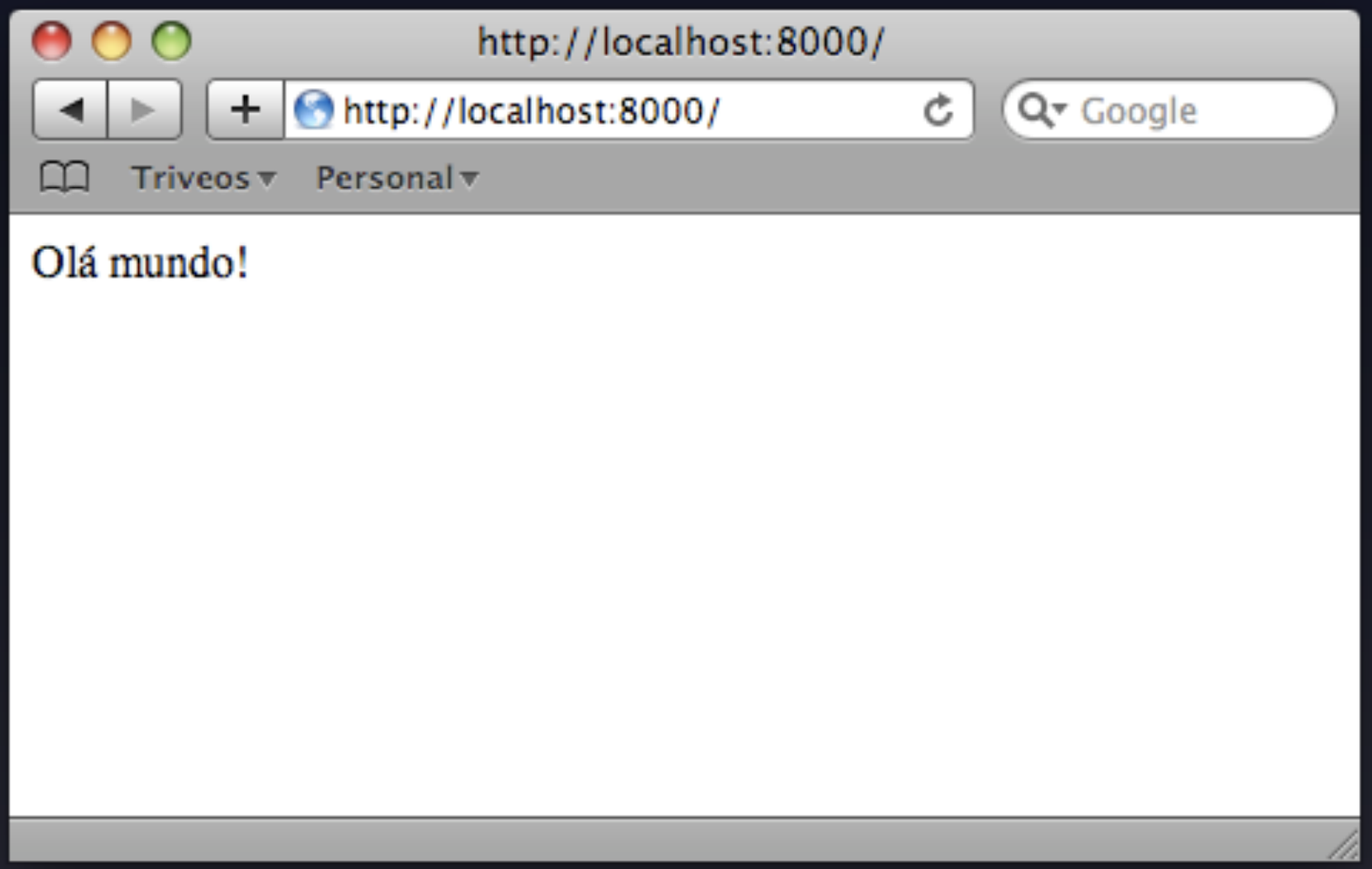
```
# -*- encoding: utf-8 -*-

from django.http import HttpResponse

def index(request):
    return HttpResponse(u"Olá mundo!")
```

triveos

Dizendo "Olá"

```
(aula) aula/gerenciador $ ./manage.py runserver
Validating models...
0 errors found

Django, using settings 'gerenciador.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Dizendo "Olá"



triveos

Models

Arquivo:
agenda/
models.py

```python
from django.db import models

class ItemAgenda(models.Model):
    data = models.DateField()
    hora = models.TimeField()
    titulo = models.CharField(max_length=100)
    descricao = models.TextField()
```

ItemAgenda
data: Date hora: Time titulo: Varchar(100) descricao: Text

Models

Tipos de campos				
BooleanField	EmailField	IntegerField	SmallIntegerField	URLField
CharField	FileField	IPAddressField	PositiveSmallIntegerField	XMLField
DateField	FilePathField	TextField	CommaSeparatedIntegerField	TimeField
DateTimeField	FloatField	SlugField	NullBooleanField	
AutoField	DecimalField	ImageField	PositiveIntegerField	

Opções dos campos			
null	db_index	help_text	unique_for_month
blank	db_tablespace	primary_key	unique_for_year
choices	default	unique	verbose_name
db_column	editable	unique_for_date	

Configurando

- Arquivo de configuração: settings.py
- Já vem preenchido
- Arquivo auto-explicativo
- Repleto de comentários e configurações *default* mais comuns
- Informações sobre todas as opções em:
  - http://docs.djangoproject.com/en/dev/ref/settings/#ref-settings

Configurando

- Configuração da conexão com o Banco de Dados para o ORM
  - Até o momento suporta os bancos: SQLite3, MySQL, PostgreSQL ou Oracle.
- Usaremos o SQLite3
- Recomendado usar outro banco de dados em ambientes de produção

Configurando

```
# Caso 2: SQLite3
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'gerenciador.db',
    'USER': '', 'PASSWORD': '',
    'HOST': '', 'PORT': '',
  }
}

# Caso 1: MySQL
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'gerenciadordb',
    'USER': 'user_name', 'PASSWORD': 'secreta123',
    'HOST': '',  'PORT': '',
  }
}
```

Configurando

Arquivo:
settings.py

```
# Caso 2: SQLite3
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'gerenciador.db',
    'USER': '', 'PASSWORD': '',
    'HOST': '', 'PORT': '',
  }
}

# Caso 1: MySQL
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'gerenciadordb',
    'USER': 'user_name', 'PASSWORD': 'secreta123',
    'HOST': '',  'PORT': '',
  }
}
```

Configurando

Arquivo:
`settings.py`

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'agenda',
)
```

Gerenciando dados

```
(aula) aula/gerenciador$ ./manage.py syncdb
:
Creating table agenda_itemagenda

You just installed Django's auth system, which means you don't have any
superusers defined.
Would you like to create one now? (yes/no): yes
Username (Leave blank to use 'osantana'): dead_parrot
E-mail address: dead_parrot@montypython.co.uk
Password: XXXXX
Password (again): XXXXX
Superuser created successfully.
:
Installing index for agenda.ItemAgenda model
```

Gerenciando dados

```
(aula) aula/gerenciador$ sqlite3 gerenciador.db
SQLite
Enter ".help" for instructions

sqlite> .schema agenda_itemagenda
CREATE TABLE "agenda_itemagenda" (
    "id" integer NOT NULL PRIMARY KEY,
    "data" date NOT NULL,
    "hora" time NOT NULL,
    "titulo" varchar(100) NOT NULL,
    "descricao" text NOT NULL,
);
```

Mostrando Dados

Mostrando dados

```
(aula) aula/gerenciador$ mkdir -p agenda/templates
```

Linguagem de Template

- A linguagem de templates do Django é composta de tags e filtros
- As tags (com comandos) devem ficar entre as marcações {% e %}
- Para imprimir o resultado de uma expressão podemos usar os marcadores {{ e }}

Linguagem de Template

- A linguagem é propositadamente construída para desetimular a inclusão de lógica dentro dos templates
- O conceito de herança de templates permite reaproveitar trechos de outros templates da mesma maneira que a herança o faz no paradigma da programação OO
- É possível criar tags e filtros personalizadas para extender a linguagem de template

Mostrando dados

Arquivo:
templates/
base.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
        "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
<head>
    {% block cabecalho %}
        <title>Agenda</title>
    {% endblock %}
</head>
<body>
    {% block corpo %}
        <!-- body -->
    {% endblock %}
</body>
</html>
```

Mostrando dados

Arquivo:
templates/
lista.html

```
{% extends 'base.html' %}

{% block corpo %}

<a href="/adiciona/">Adicionar novo item</a>
<ul>
{% for item in lista_itens %}
  <li>
    <a href="/item/{{ item.id }}">
        {{ item.data|date:'d/m/Y' }} - {{ item.titulo }}</a>
  </li>
{% empty %}
  <li>Sem itens na lista</li>
{% endfor %}
</ul>

{% endblock %}
```

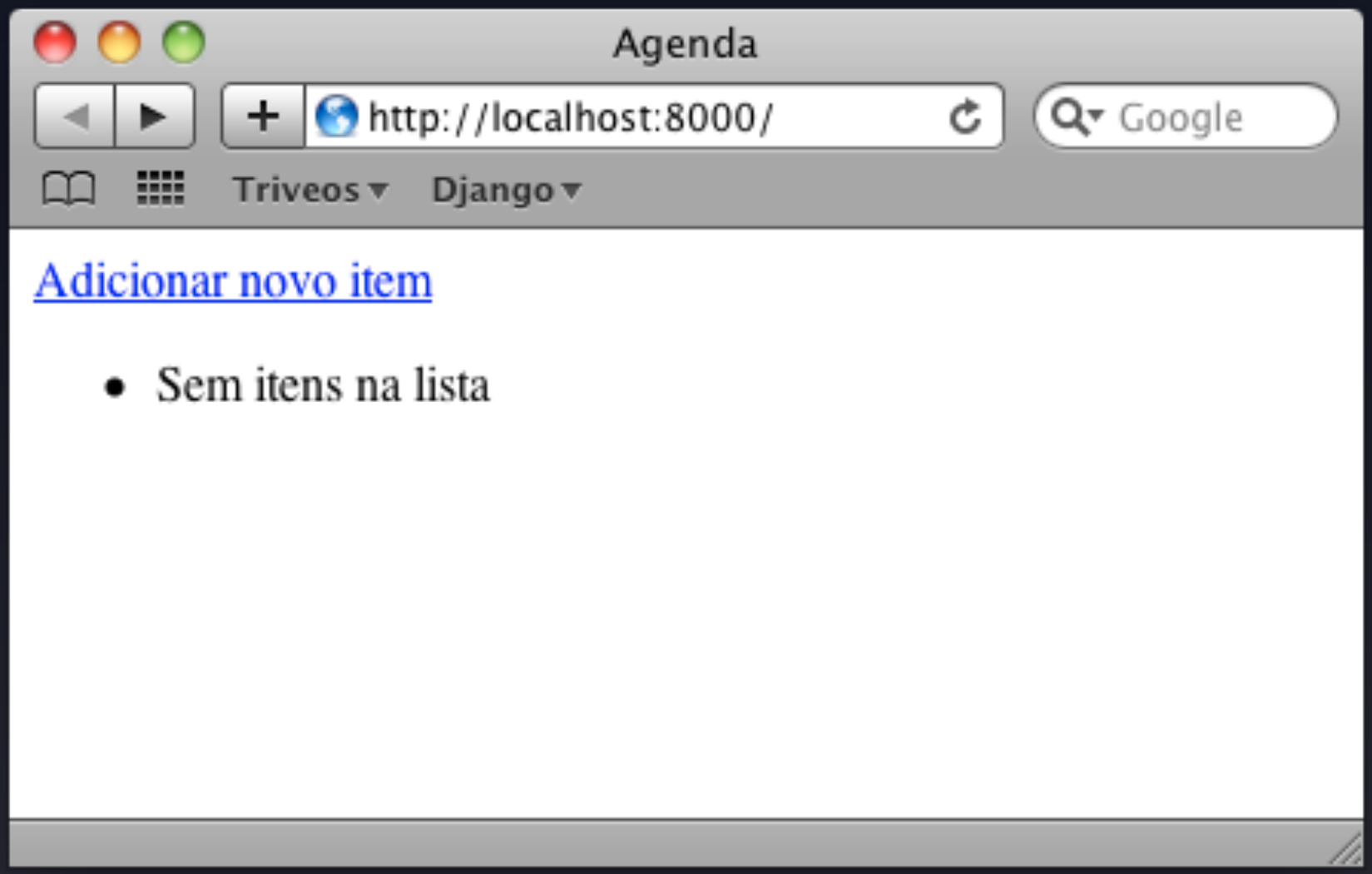
View

Mostrando dados

Arquivo:
agenda/
views.py

```
# -*- encoding: utf-8 -*-

from django.shortcuts import render

from models import ItemAgenda

def index(request):
     lista_itens = ItemAgenda.objects.all()
     return render(request, "lista.html",
                    {'lista_itens': lista_itens})
```

Mostrando dados



triveos

Criando objetos

Formulários

- Uma das formas mais usadas para interação dos usuários com a aplicação
- Django oferece classes geradoras de formulários: forms.Form e forms.ModelForm
- A impressão dos campos de um formulário podem ser feitas com uma instrução no template: form.as_p, form.as_li ou form.as_table

Formulários

- Campos
  - Fazem o mapeamento do formulário recebido via POST já convertendo o valor recebido para um tipo Python
- Widget
  - O visual que o campo terá na página HTML (ex. campo data será um campo texto ou um calendário para escolha da data)

Formulários

Tipos de campos			
BooleanField	DateTimeField	FloatField	NullBooleanField
CharField	DecimalField	ImageField	RegexField
ChoiceField	EmailField	IntegerField	TimeField
TypedChoiceField	FileField	IPAddressField	URLField
DateField	FilePathField	MultipleChoiceField	

Tipos de widgets			
TextInput	DateInput	Select	MultiWidget
PasswordInput	DateTimeInput	NullBooleanSelect	SplitDateTimeWidget
HiddenInput	TimeInput	SelectMultiple	SelectDateWidget
MultipleHiddenInput	Textarea	RadioSelect	
FileInput	CheckboxInput	CheckboxSelectMultiple	

ModelForm

Arquivo:
agenda/
forms.py

```python
from django import forms
from agenda.models import ItemAgenda

class FormItemAgenda(forms.ModelForm):
    data = forms.DateField(
            widget=forms.DateInput(format='%d/%m/%Y'),
            input_formats=['%d/%m/%Y', '%d/%m/%y'])

    class Meta:
        model = ItemAgenda
        fields = ('titulo', 'data', 'hora', 'descricao')
```

Formulários

Arquivo:
templates/
adiciona.html

```
{% extends 'base.html' %}

{% block corpo %}

<form action="" method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Adicionar</button>
</form>

{% endblock %}
```

Formulários

Arquivo:
`urls.py`

```
from django.conf.urls.defaults import *
# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()
urlpatterns = patterns('',
    # Example:
    # (r'^gerenciador/', include('gerenciador.foo.urls')),

    (r'^$', 'agenda.views.index'),
    (r'^adiciona/$', 'agenda.views.adiciona'),

)
```

ModelForm

Arquivo:
agenda/
views.py

```python
from django.shortcuts import render, redirect
from forms import FormItemAgenda

def adiciona(request):
    if request.method == 'POST': # Formulário enviado
        form = FormItemAgenda(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect("/")
    else:
        # Exibe formulário em branco
        form = FormItemAgenda()

    return render(request, "adiciona.html", {'form': form})
```

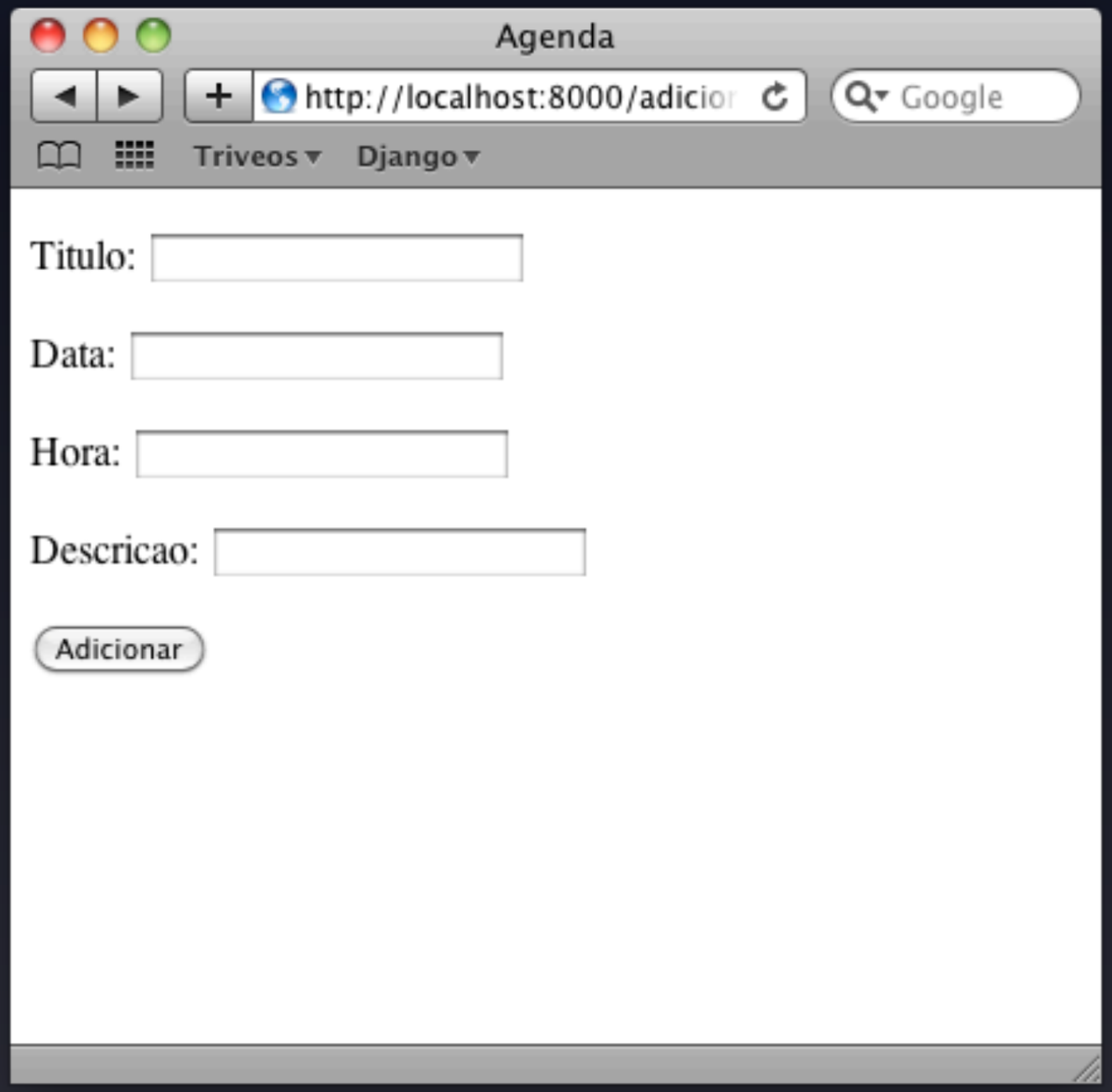
ModelForm

Arquivo:
agenda/
views.py

```
from django.shortcuts import render, redirect
from forms import FormItemAgenda

def adiciona(request):
    form = FormItemAgenda(request.POST or None, request.FILES or None)
    if form.is_valid():
        form.save()
        return redirect("/")

    return render(request, "adiciona.html", {'form': form})
```

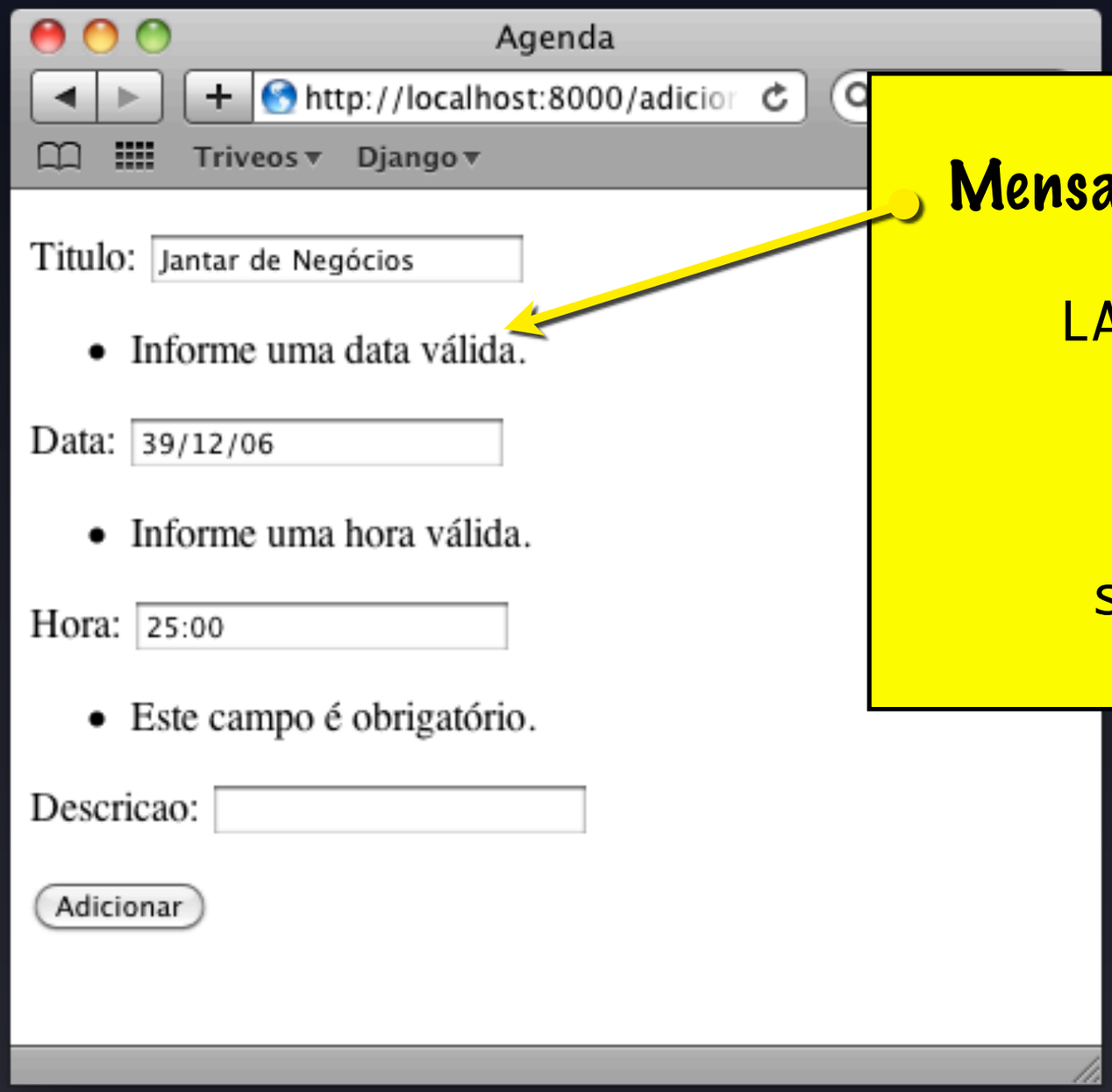
Versão aprimorada

Formulários



triveos

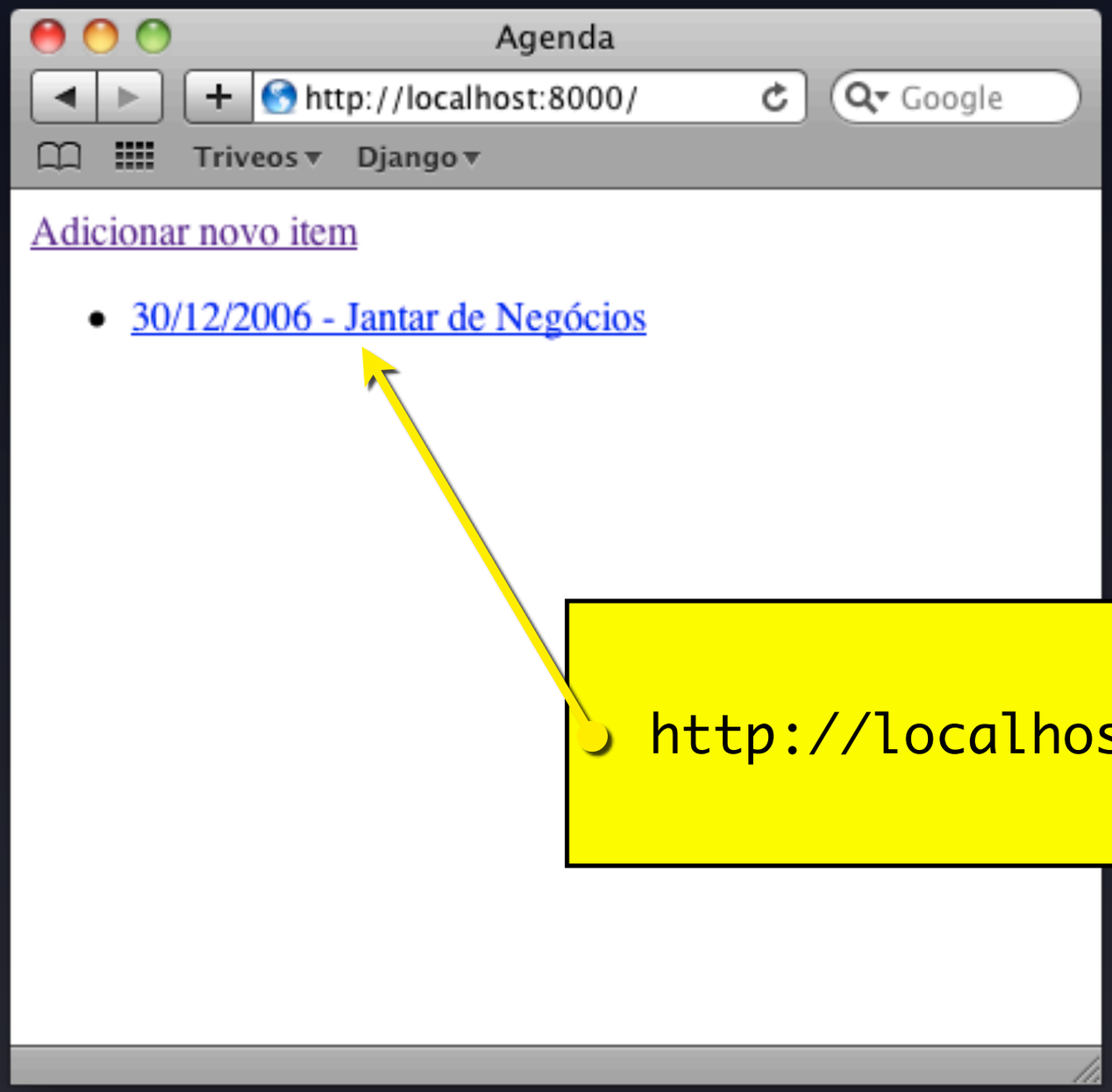
Formulários



Mensagens em português:
Mudar opção
LANGUAGE_CODE
para
`'pt-br'`
no arquivo
`settings.py`

triveos

Editando objetos



triveos

Exibindo objetos

Arquivo:
`urls.py`

```
from django.conf.urls.defaults import *
# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()
urlpatterns = patterns('',
    # Example:
    # (r'^gerenciador/', include('gerenciador.foo.urls')),

    (r'^$', 'agenda.views.lista'),
    (r'^adiciona/$', 'agenda.views.adiciona'),
    (r'^item/(?P<nr_item>\d+)/$', 'agenda.views.item'),
:
)
```

ModelForm

Arquivo:
agenda/
views.py

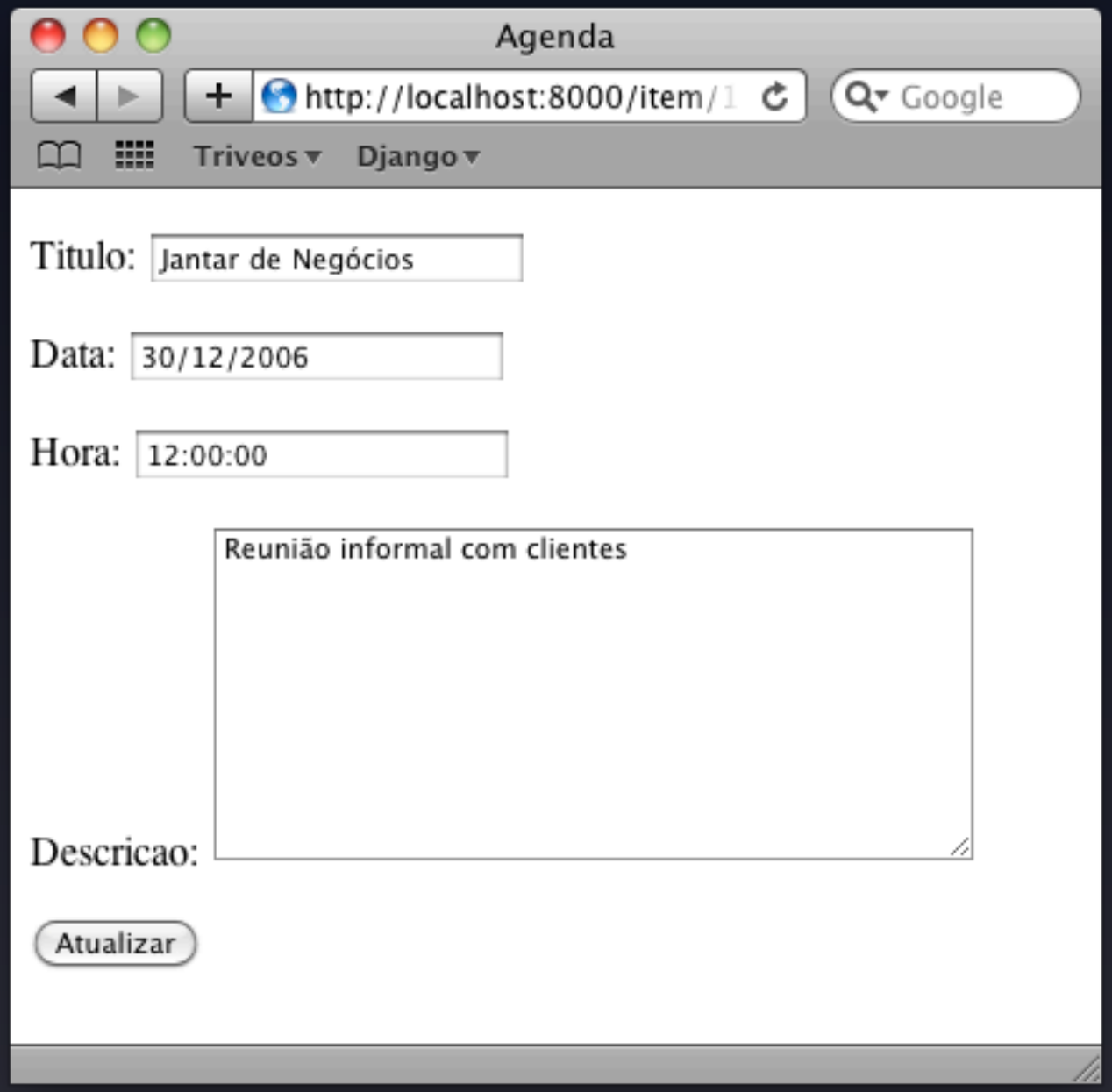
```
def item(request, nr_item):
    item = get_object_or_404(ItemAgenda, pk=nr_item)
    form = FormItemAgenda(request.POST or None,
                          request.FILES or None, instance=item)
    if form.is_valid():
        form.save()
        return redirect("/")

    return render(request, "item.html", {'form': form})
```

ModelForm

Arquivo:
templates/
item.html

```
{% extends 'base.html' %}

{% block corpo %}

<form action="" method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Atualizar</button>
</form>

{% endblock %}
```

ModelForm



triveos

Atividades

- Adicionar a funcionalidade de remoção
- de item com confirmação

Agenda Multiusuário

triveos

Segurança

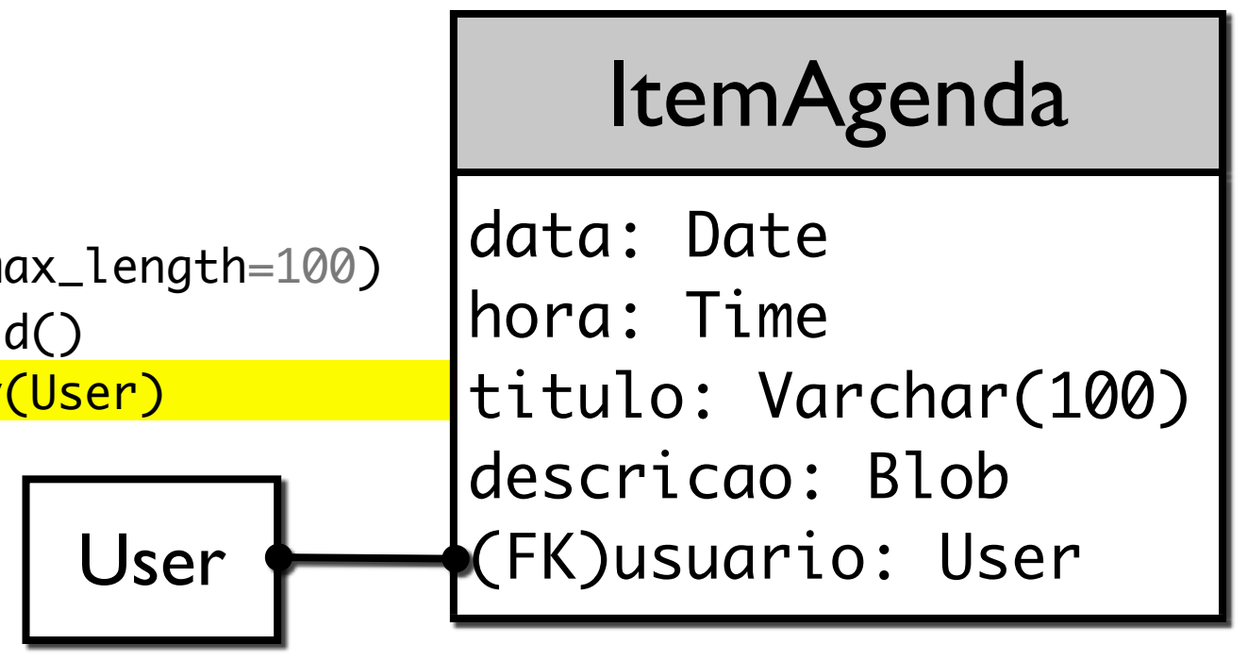
- Aplicação: django.contrib.auth
  - Requer aplicações:
    - django.contrib.sessions
    - django.contrib.contenttypes
- *Models*:
  - User
  - Group
  - Permission
- Sistema de perfil de usuário

triveos

Segurança

Arquivo:
agenda/
models.py

```
from django.db import models

from django.contrib.auth.models import User

class ItemAgenda(models.Model):
    data = models.DateField()
    hora = models.TimeField()
    titulo = models.CharField(max_length=100)
    descricao = models.TextField()
    usuario = models.ForeignKey(User)
```



triveos

Segurança

```
(aula) gerenciador$ rm gerenciador.db
(aula) gerenciador$ ./manage.py syncdb
Creating table auth_permission
Creating table auth_group
Creating table auth_user
Creating table auth_message
Creating table django_content_type
Creating table django_session
:
:
Would you like to create one now? (yes/no): yes
Username (Leave blank to use 'osantana'): dead_parrot
E-mail address: dead_parrot@montypython.co.uk
Password: XXXXX
Password (again): XXXXX
Superuser created successfully.
:
```

Segurança

Arquivo:
agenda/
views.py

```python
# -*- encoding: utf-8 -*-


from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required

from models import ItemAgenda
from forms import FormItemAgenda

@login_required
def lista(request):
    lista_itens = ItemAgenda.objects.filter(usuario=request.user)
    return render(request, "lista.html", {'lista_itens': lista_itens})
```

Segurança

Arquivo:
agenda/
views.py

```
@login_required
def adiciona(request):
    form = FormItemAgenda(request.POST or None,
                          request.FILES or None)
    if form.is_valid():
        item = form.save(commit=False)
        item.usuario = request.user
        item.save()
        return redirect("/")

    return render(request, "adiciona.html", {'form': form})
```

Segurança

Arquivo:
agenda/
views.py

```
@login_required
def item(request, nr_item):
    item = get_object_or_404(ItemAgenda, usuario=request.user, pk=nr_item)
    form = FormItemAgenda(request.POST or None,
                          request.FILES or None, instance=item)
    if form.is_valid():
        form.save()
        return redirect("/")

    return render(request, "item.html", {'form': form})
```

Segurança

Arquivo:
agenda/
views.py

```python
@login_required
def remove(request, nr_item):
    item = get_object_or_404(ItemAgenda, usuario=request.user, pk=nr_item)
    if request.method == "POST":
        item.delete()
        return redirect("/")

    return render(request, "remove.html", {'item': item})
```

Segurança

Arquivo:
`settings.py`

```
LOGIN_URL = "/login/"
LOGOUT_URL = "/logout/"
LOGIN_REDIRECT_URL = "/"
```

Segurança

Arquivo:
urls.py

```
:
urlpatterns = patterns('',
    # Example:
    # (r'^gerenciador/', include('gerenciador.foo.urls')),

    (r'^$', 'agenda.views.lista'),
    (r'^adiciona/$', 'agenda.views.adiciona'),

    (r'^item/(?P<nr_item>\d+)/$', 'agenda.views.item'),
    (r'^remove/(?P<nr_item>\d+)/$', 'agenda.views.remove'),

    (r'^login/$', 'django.contrib.auth.views.login',
                  {'template_name': 'login.html' }),
    (r'^logout/$', 'django.contrib.auth.views.logout_then_login',
                  {'login_url': '/login/'}),
```

triveos

Segurança

Arquivo:
templates/
login.html

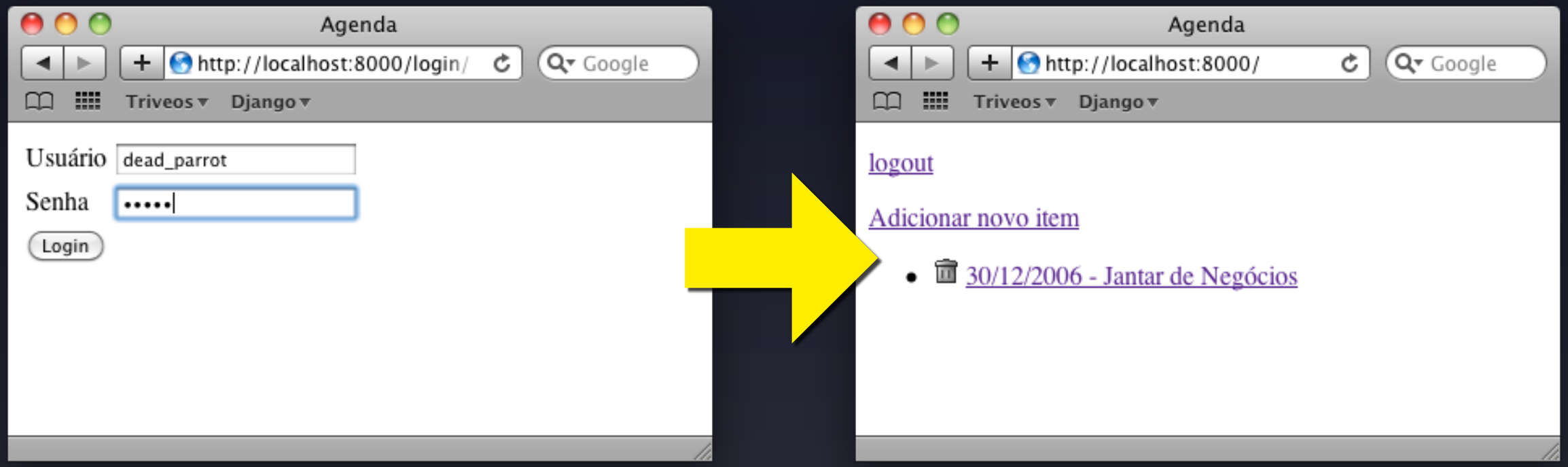
```
{% extends "base.html" %}

{% block corpo %}

{% if form.errors %}
<p>Seu usuário e senha não conferem. Tente novamente.</p>
{% endif %}
<form action="" method="post">
{% csrf_token %}
<table>
  <tr><td>{{form.username.label_tag}}</td><td>{{form.username}}</td></tr>
  <tr><td>{{form.password.label_tag}}</td><td>{{form.password}}</td></tr>
  <tr><td colspan="2"><button type="submit">Login</button></td></tr>
</table>
</form>
{% endblock %}
```

Segurança

Arquivo:
templates/
lista.html

```
{% extends 'base.html' %}
{% block corpo %}
<p><a href="/logout/">logout</a></p>
<p><a href="/adiciona/">Adicionar novo item</a></p>
<ul>
{% for item in lista_itens %}
<li>
<a href="/remove/{{item.id}}">remover</a>
<a href="/item/{{item.id}}">
   {{ item.data|date:"d/m/Y" }} - {{ item.titulo }}
</a>
</li>
{% empty %}
    <li>Sem itens na lista</li>
{% endfor %}
</ul>
{% endblock %}
```

Segurança



triveos

Admin

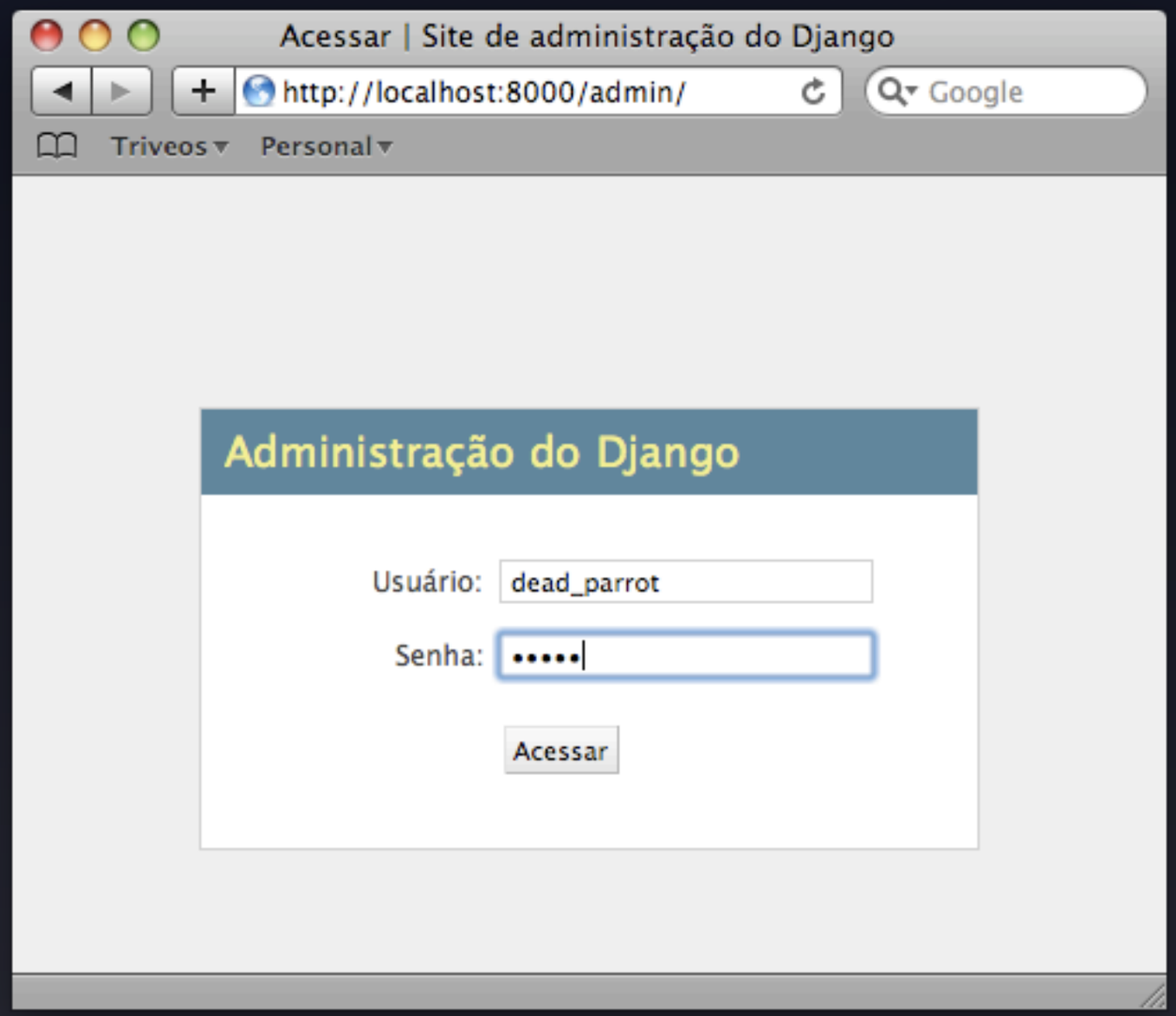
- Adicione `'django.contrib.admin'` na opção `INSTALLED_APPS` do `settings.py`
- Remova comentários relacionados a admin do `urls.py`
- Rode novamente `./manage.py syncdb`
- Crie um arquivo `agenda/admin.py` contendo...
- Desabilite a aplicação staticfiles em `INSTALLED_APPS`

Admin

Arquivo:
agenda/
admin.py

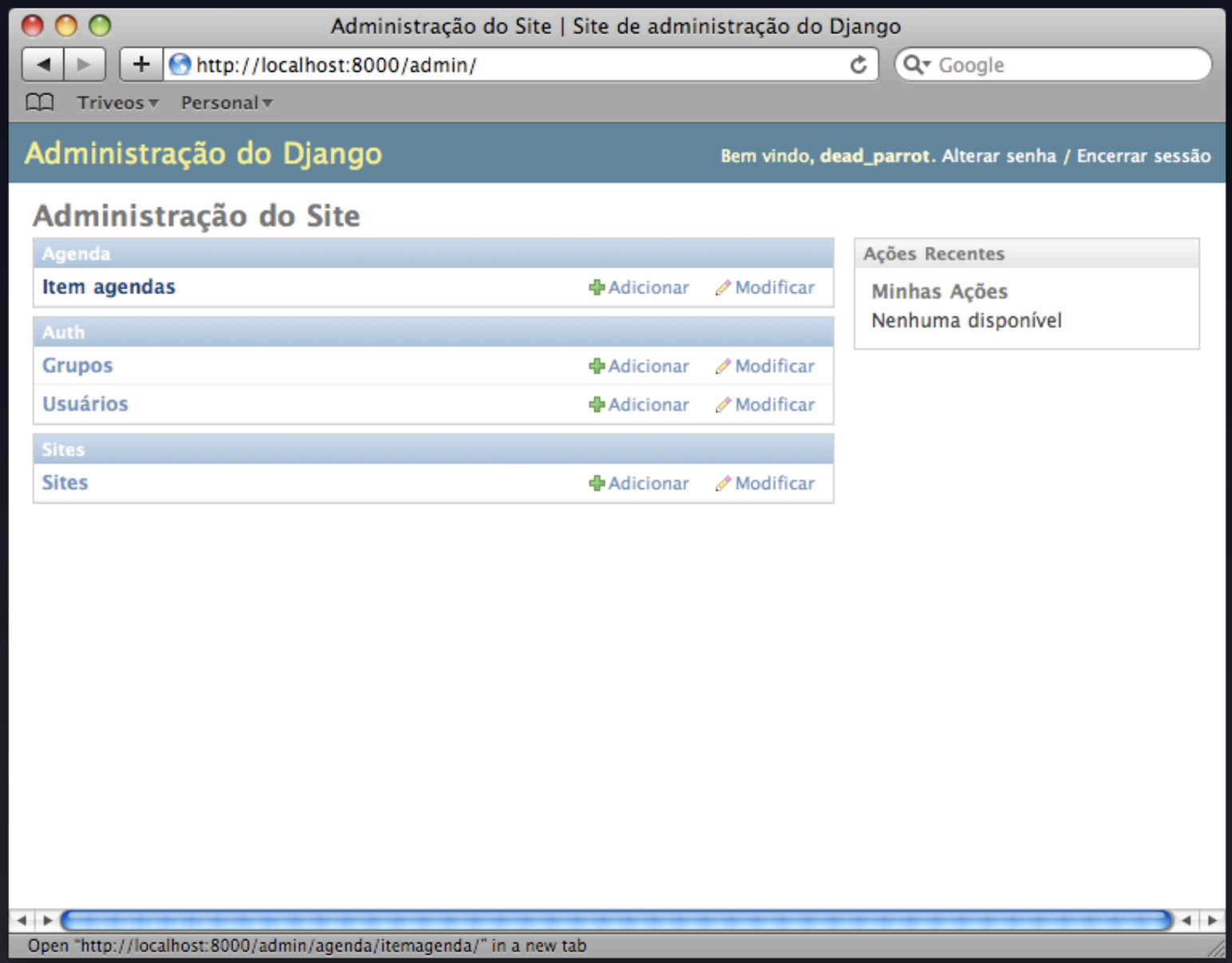
```
from gerenciador.agenda.models import ItemAgenda
from django.contrib import admin

admin.site.register(ItemAgenda)
```

Admin



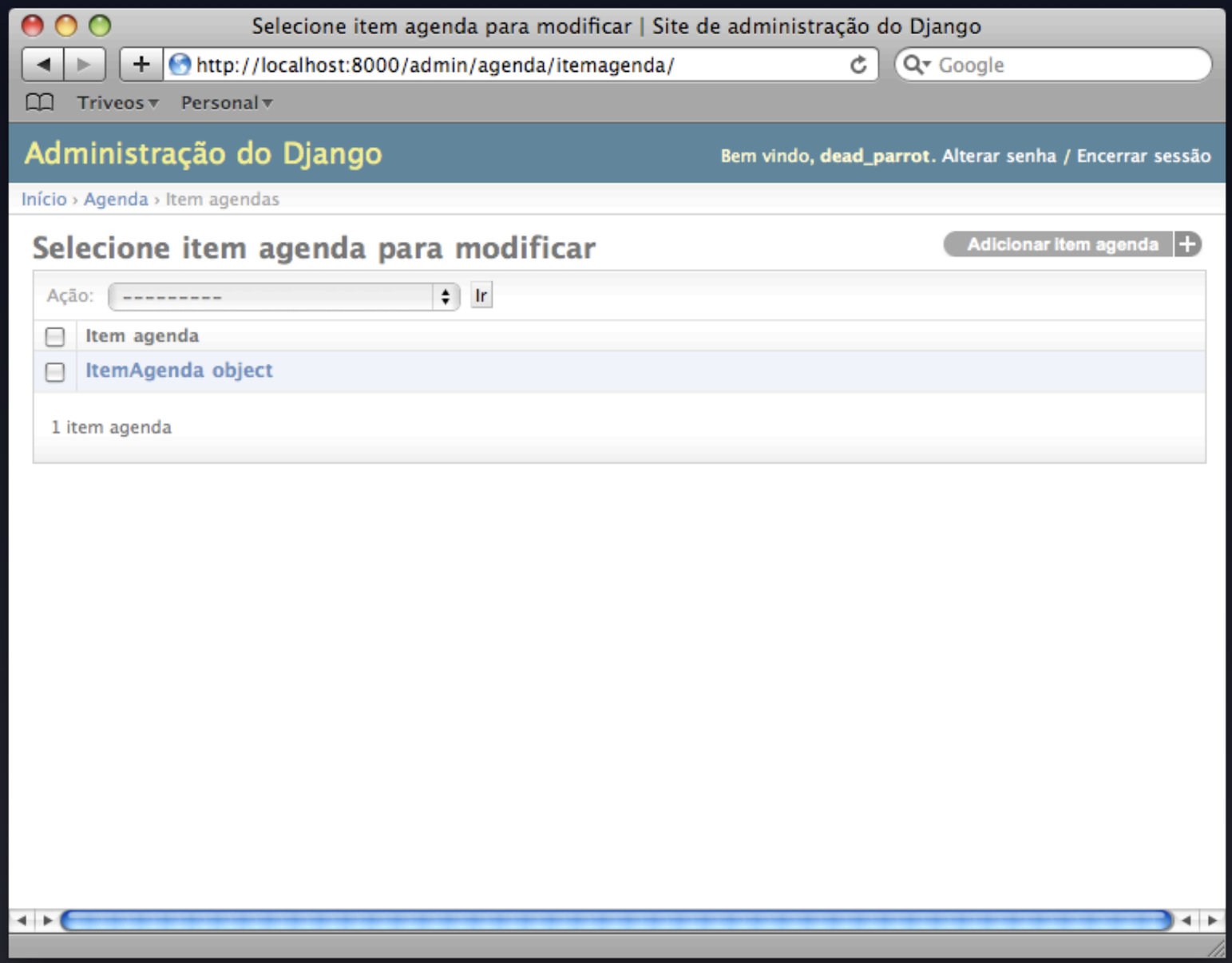
triveos

Admin



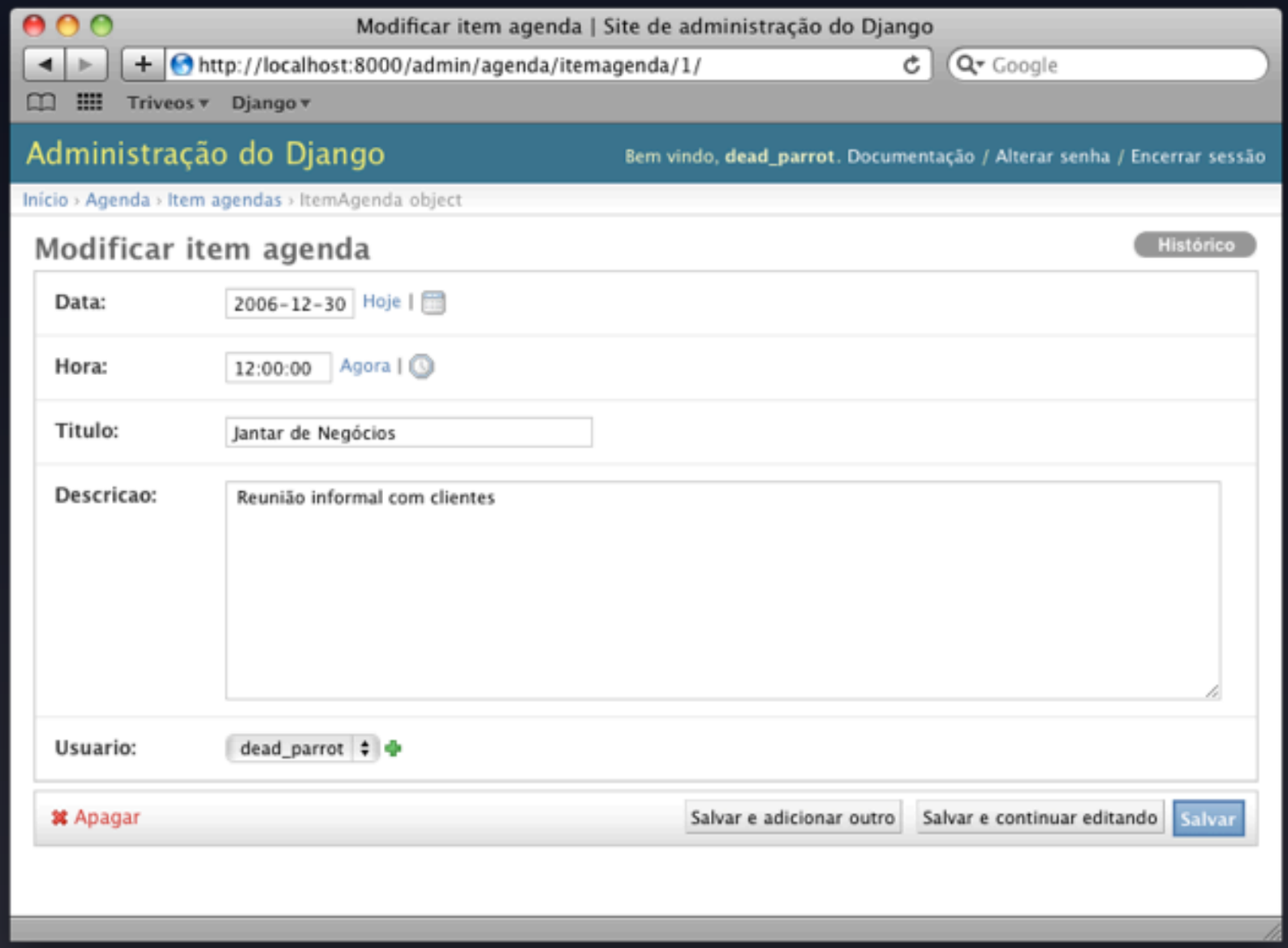
triveos

Admin



triveos

Admin



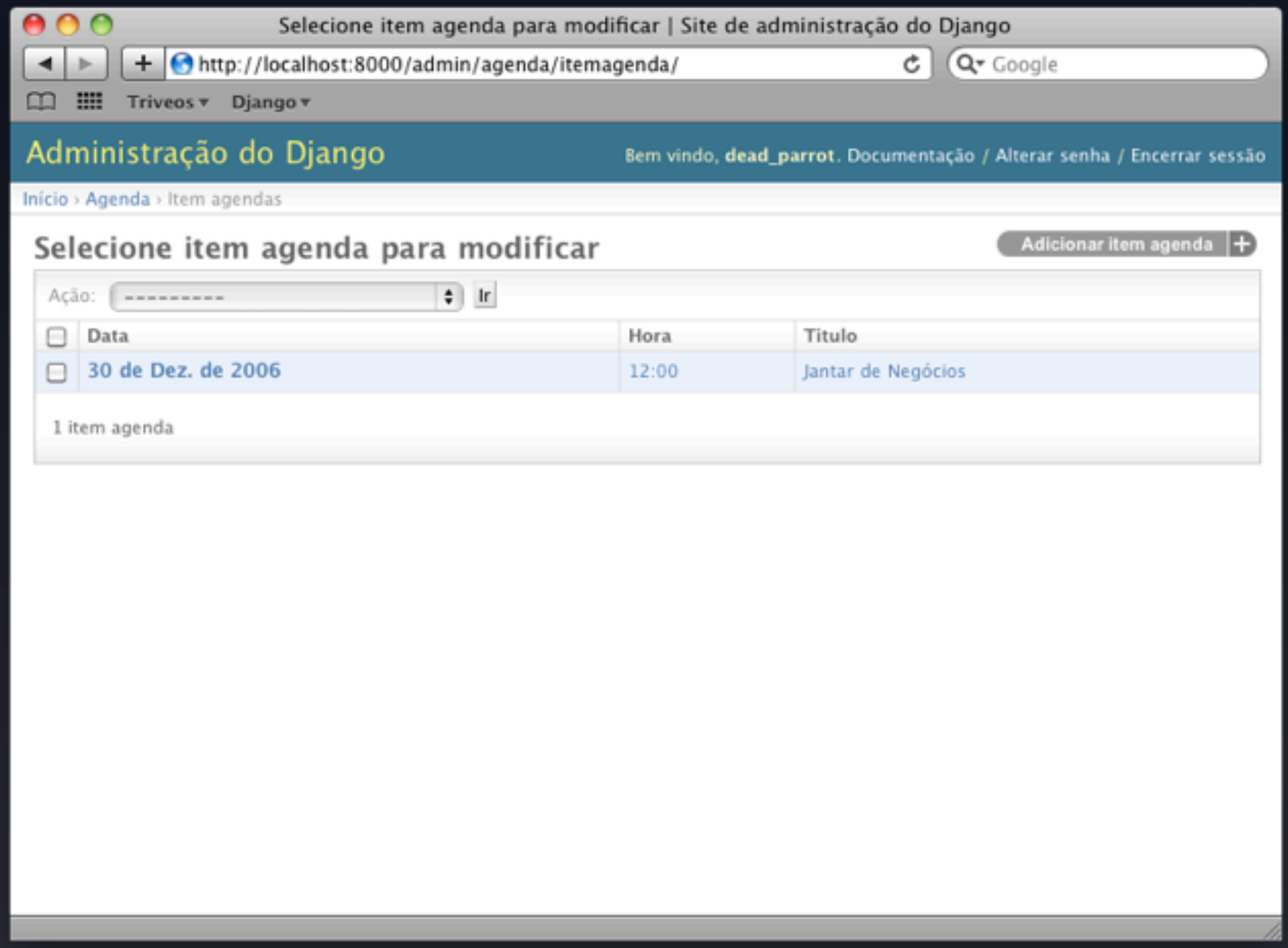
triveos

Admin

Arquivo:
agenda/
admin.py

```python
from gerenciador.agenda.models import ItemAgenda
from django.contrib import admin

class ItemAgendaAdmin(admin.ModelAdmin):
    fields = ('data', 'hora', 'titulo', 'descricao')
    list_display = ('data', 'hora', 'titulo')

    def queryset(self, request):
        qs = super(ItemAgendaAdmin, self).queryset(request)
        return qs.filter(usuario=request.user)

    def save_model(self, request, obj, form, change):
        obj.usuario = request.user
        obj.save()

admin.site.register(ItemAgenda, ItemAgendaAdmin)
```

Admin



triveos

Django ORM

Relacionamentos

Arquivo:
agenda/
models.py

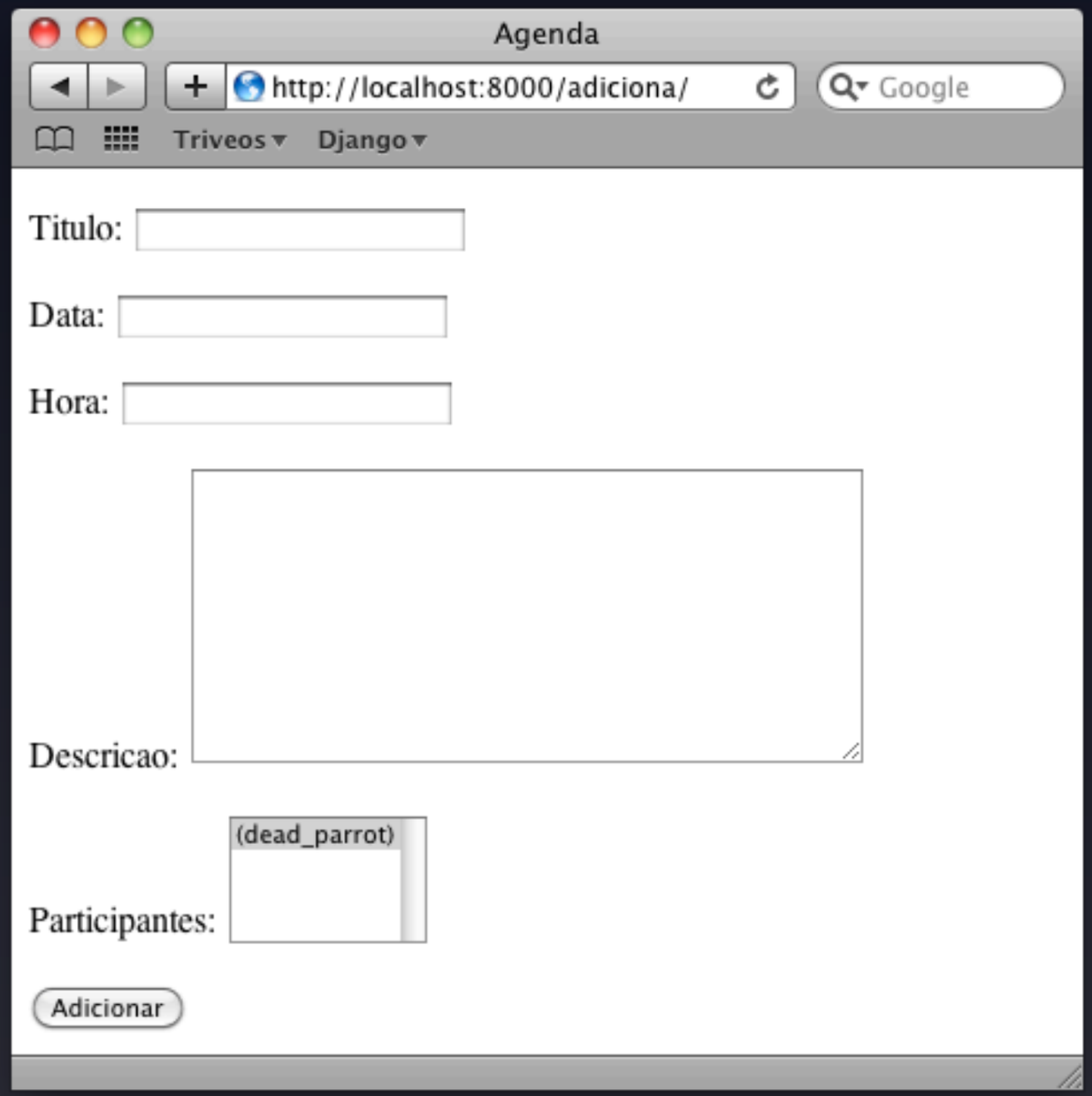
```
from django.db import models

from django.contrib.auth.models import User

class ItemAgenda(models.Model):
    data = models.DateField()
    hora = models.TimeField()
    titulo = models.CharField(max_length=100)
    descricao = models.TextField()
    usuario = models.ForeignKey(User)
    participantes = models.ManyToManyField(User,
                                related_name='item_participantes')
```

Relacionamentos

Arquivo:
agenda/
forms.py

```python
from django import forms
from django.contrib.auth.models import User
from agenda.models import ItemAgenda

class ModelMultipleUserField(forms.ModelMultipleChoiceField):
    def label_from_instance(self, user):
        return u"%s (%s)" % (user.get_full_name(), user)

class FormItemAgenda(forms.ModelForm):
    data = forms.DateField(
            widget=forms.DateInput(format='%d/%m/%Y'),
            input_formats=['%d/%m/%Y', '%d/%m/%y'])
    participantes = ModelMultipleUserField(queryset=User.objects.all())

    class Meta:
        model = ItemAgenda
        fields = ('titulo', 'data', 'hora', 'descricao', 'participantes')
```

Relacionamentos



triveos

Manipulando Queries

```
$ ./manage.py shell
Python 2.5.1 (r251:54863, Feb  6 2009, 19:02:12)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from agenda.models import ItemAgenda as ia
>>> print ia.objects.all()
[<ItemAgenda: ItemAgenda object>, <ItemAgenda: ItemAgenda object>,
<ItemAgenda: ItemAgenda object>]
>>> print ia.objects.get(pk=1)
ItemAgenda object
>>> print ia.objects.filter(data__month=8)
[<ItemAgenda: ItemAgenda object>]
>>> print ia.objects.order_by('-data')
[<ItemAgenda: ItemAgenda object>, <ItemAgenda: ItemAgenda object>,
<ItemAgenda: ItemAgenda object>]
```

Executando SQL

```
>>> from django.db import connection, transaction
>>> cursor = connection.cursor()
>>> c = cursor.execute("SELECT * FROM agenda_itemagenda")
>>> cursor.fetchall()
[(1, datetime.date(2009, 4, 25), datetime.time(12, 0), u'foo', u'bar', 1),
(2, datetime.date(2009, 8, 20), datetime.time(23, 41, 17), u'bla', u'blek',
1), (3, datetime.date(2000, 1, 10), datetime.time(20, 0), u'6', u'foo', 1)]
>>> c = cursor.execute("UPDATE agenda_itemagenda SET titulo='novo titulo'
WHERE id=1")
>>> transaction.commit_unless_managed()

>>> c = cursor.execute("SELECT * FROM agenda_itemagenda")
>>> cursor.fetchall()
[(1, datetime.date(2009, 4, 25), datetime.time(12, 0), u'novo titulo',
u'bar', 1), (2, datetime.date(2009, 8, 20), datetime.time(23, 41, 17),
u'bla', u'blek', 1), (3, datetime.date(2000, 1, 10), datetime.time(20, 0),
u'6', u'foo', 1)]
```

triveos

Signals & E-Mail

Signals

Arquivo:
agenda/
models.py

```
def envia_email(**kwargs):
    print "Implementar..."

models.signals.post_save.connect(envia_email,
        sender=ItemAgenda,
        dispatch_uid="agenda.models.ItemAgenda")
```

E-mail

Arquivo:
agenda/
models.py

```
from datetime import datetime

def envia_email(**kwargs):
  try:
    item = kwargs['instance']
  except KeyError:
    return
for participante in item.participantes.all():
  if participante.email:
    dados = (item.titulo,
             datetime.strftime(item.data, "%d/%m/%Y"),
             item.hora)
    participante.email_user(
        subject="[evento] %s dia %s as %s" % dados,
        message="Evento: %s\nDia: %s\nHora: %s" % dados,
        from_email=item.usuario.email
    )
```

E-mail

Arquivo:
settings.py

```
DEFAULT_FROM_EMAIL = "eventos@empresa.com"
EMAIL_HOST='smtp.empresa.com'
EMAIL_PORT=587
EMAIL_HOST_USER="eventos"
EMAIL_HOST_PASSWORD="senha123"
EMAIL_USE_TLS=True
```

Signals & E-Mail

« Back to Inbox | Archive | Report spam | Delete | Move to ▾ | Labels ▾ | More actions ▾

[evento] Jantar de Negócios dia 30/12/2006 as 12:00:00 Inbox | X

from [redacted] hide details 6:38 PM (0 minutes ago)
sender-time Sent at 6:38 PM (UTC). Current time there: 9:39 PM.
to [redacted]
date Mon, Apr 26, 2010 at 6:38 PM
subject [evento] Jantar de Negócios dia 30/12/2006 as 12:00:00

```
Evento: Jantar de Negócios
Dia: 30/12/2006
Hora: 12:00:00
```

Reply Forward

triveos

Outras funções

- Arquivos estáticos e de mídia
- GeoDjango (aplicações GIS)
- Internacionalização e Localização
- Funções "locais" (ex. CPF, CNPJ, CEP, …)
- Aplicações de comentários, moderação, …
- Serviços de segurança: signed cookies, CSRF Protection, Clickjacking protection
- Cache multi-backend
- Páginas planas, multisite, etc.

Referências

- Python
  - http://www.python.org/
  - http://www.python.org.br/
- Django
  - http://www.djangoproject.com/
  - http://www.djangobook.com/
  - http://www.djangobrasil.org/
- Geral
  - http://www.pythonologia.org/
  - http://triveos.ludeos.com.br/

triveos

http://db.tt/j2wwFvX3

Cursos online

http://www.ludeos.com.br