Treinamento Python

Terra 2014

Pré-requisitos

- sudo pip install argparser
- sudo pip install ipython iPy

Linguagem interpretada

- Source code > Bytecode > Output
- Arquivo .pyc

```
#!/usr/bin/env pytho
# encoding: utf-8

my_dict = { 'a':1 }
```

```
0 BUILD_MAP                1
3 LOAD_CONST               0 (1)
6 LOAD_CONST               1 ('a')
9 STORE_MAP
10 STORE_NAME              0 (my_dict)
13 LOAD_CONST              2 (None)
16 RETURN_VALUE
```

Fortemente tipada

```
In [1]: inteiro = 1

In [2]: type(inteiro)
Out[2]: int

In [3]: string = '1'

In [4]: type(string)
Out[4]: str

In [5]: resultado = string + inteiro
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-72a90d30cacc> in <module>()
----> 1 resultado = string + inteiro

TypeError: cannot concatenate 'str' and 'int' objects

In [6]:
```

Fortemente tipada

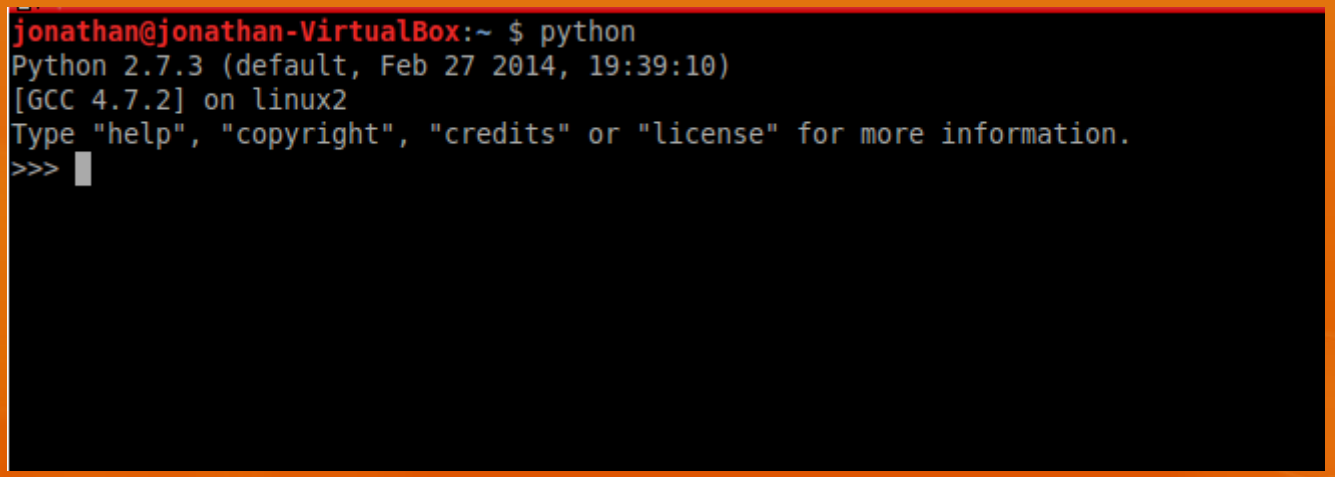
```
Starting php
type 'h' or 'help' to see instructions & features
php> $inteiro = 1

php> echo gettype($inteiro)
integer
php> $string = '1'

php> echo gettype($string)
string
php> $resultado = $string + $inteiro

php> echo gettype($resultado)
integer
php> echo $resultado
2
php>
```

Delimitação

```
1  # PYTHON
2
3  if a == 0:
4      print 'zero'
5  else:
6      print a
7
8  #PHP
9  
10 <?php
11 if ($a === 0) {
12     echo "zero";
13 } else {
14     echo $a;
15 }
16 ?>
```

Python & IPython

- Abrir o shell
- Digitar python



Python & IPython

- Operações básicas
  - + adição
  - - subtração
  - * multiplicação
  - / divisão

Python & IPython

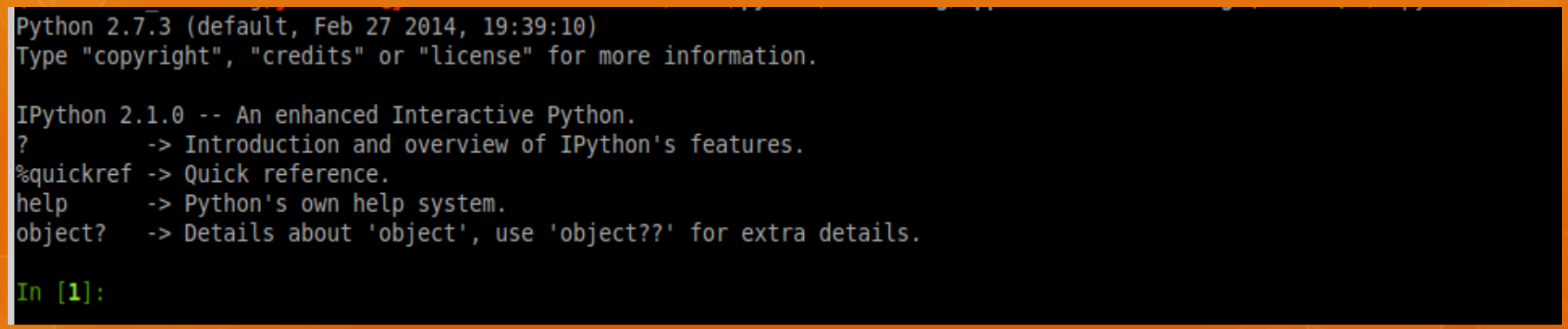
- Desafio 1:
  - 20 + 5 * 4 + 2 deve dar 150. O que falta para isso?

Python & IPython

- >>> exit
  - Use exit() or Ctrl-D (i.e. EOF) to exit

Python & IPython

- $ sudo apt-get install ipython



Python & IPython

- Desafio 2:
  - 1 Euro = 1.35 dólares.
  - Quanto é 65.54 dólares em Euros?
  - Utilizar o ipython

Primeiro arquivo em python

- Salvar como 1.py
- $ python 1.py
- >>> Ola!

```
1  #!/path/to/python
2
3  print "Ola!"
```

hashbang e PEP0263

```
1  #!/path/to/python
2
3  print "Ola!"
```

$ chmod 755 1.py

$ ./1.py

```
#!/usr/bin/python

print "Ola!"
```

hashbang e PEP0263

```
#!/usr/bin/python

print "Olá!"
```

```
jonathan@jonathan-VirtualBox:~ $ ./1.py
  File "./1.py", line 3
SyntaxError: Non-ASCII character '\xc3' in file ./1.py on line 3,
```

PEP - *Python Enhancement Proposals*

- Guidelines
- Propostas de novos recursos
- Especificação
- Justificativas

PEP - *Python Enhancement Proposals*

- PEP8
  - Coding conventions for the Python code
- PEP0263
  - Defining Python Source Code Encodings

PEP0263

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

print "Olá!"
```

- *Python will default to ASCII as standard encoding if no other encoding hints are given.*

Input

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

fulano = raw_input("Digite seu nome: ")
print "Olá " + fulano + "!"
```

If, elif, else

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  fulano = raw_input("Digite seu nome: ")
5  if fulano == 'João':
6      print "Seja bem vindo " + fulano + "!"
7  else:
8      print "Você não é bem vindo aqui!"
9
```

If, elif, else

- == Igual
- != Diferente
- > Maior que
- < Menor que
- And
- Or

If, elif, else

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  fulano = raw_input("Digite seu nome completo: ")
5  if 'Silva' in fulno:
6      print "Você faz parte da família!"
```

- Operador ‘in’ em strings.

If, elif, else

- Desafio:
  - Fazer um script que pergunte o nome do servidor.
  - Validar de o servidor pertence ao terra checando se estão nos domínios terra.* ou trrsf.*
  - Escrever na tela se o servidor faz parte de MIA, POA ou CIS. Senão, devolver “Datacenter desconhecido”.

list, dict, tuple

```
lista = []
dicionario = {}
tupla = ()

lista.append('nome')
dicionario['nome'] = 'x'
tupla = ('string', 1)
```

For loop

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  minha_lista = []
5  meu_dicionario = {}
6
7  minha_lista.append('nome')
8  meu_dicionario['nome'] = 'x'
9
10 for item in minha_lista:
11     print item
12
13 for item in meu_dicionario:
14     print meu_dicionario[item]
15
```

While Loop

- Fazer um programa que escreva na tela o que o usuário digitou.
- Se ele digitar 'exit', sair do programa.
- Utilizar o raw_input + while

```
12
13  while <cond>:
14      # fazer alguma coisa
```

list, dict, tuple

- Desafio:
  - Escrever um dicionário onde a chave é a nome da equipe no Terra ou produto principal em que trabalha e o valor é uma lista de tuplas onde cada item é o nome de cada colega de equipe e o RE (fake).
  - Listar esses dados

import

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  print "Bem vindo. Escrve 'sair' para sair."
5
6  while True:
7      entrada = raw_input("Nome: ")
8      if 'Silva' in entrada:
9          print "Você faz parte da família!"
10     elif entrada == 'sair':
11         sys.exit()
12
13
```

import this

- Bonito é melhor que feio.
- Explícito é melhor que implícito.
- Simples é melhor que complexo.
- Complexo é melhor que complicado.
- Linear é melhor do que aninhado.
- Esparso é melhor que denso.
- Legibilidade conta.
- Casos especiais não são especiais o bastante para quebrar as regras.
- Ainda que praticidade vença a pureza.
- Erros nunca devem passar silenciosamente.
- A menos que sejam explicitamente silenciados.
- Diante da ambigüidade, recuse a tentação de adivinhar.
- Deveria haver um — e preferencialmente só um — modo óbvio para fazer algo.
- Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês.
- Agora é melhor que nunca.
- Embora nunca freqüentemente seja melhor que *já*.
- Se a implementação é difícil de explicar, é uma má idéia.
- Se a implementação é fácil de explicar, pode ser uma boa idéia.
- Namespaces são uma grande idéia — vamos ter mais dessas!

import

- from lib import Classe
  - classe= Classe()
- import lib
  - classe = lib.Classe()

import

- $> pip install Ipy

```
from IPy import IP
network = IP('192.168.0.0/30')
for ip in network:
    print ip
```

import

- Desafio:
  - Continuando o programa anterior, adicionar uma verificação que, se o usuário digitar uma rede, ele deve escrever os IPS dessa rede.
  - Se o usuário digitar um hostname, deve manter a validação anterior.

Conversão de dados

- valor = str(valor)
- valor = int(valor)
- Usando as funções soma e subtração, fazer um script que receba o num1 e num2.

Funções

- Como ficam as funções soma, divisão, subtração e multiplicação?

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  def soma(num1, num2):
5      print num1 + num2
6
```

- soma(1, 3)
- >>> 4

Funções

- Lambda são funções anonimas.
- Alterar a calculadora para usara lambda

```
17
18  soma = lambda numero1, numero2: numero1 + numero2
19
```

Funções

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  def escreve_ips(rede):
5
6      print 'Continuar aqui!'
```

```
6      
7  def escreve_ips(rede=None):
8      print 'continuar aqui'
```

Funções

```
1  def func(*args, **kwargs):
2      print kwargs # dict dos argumentos nomeados
3      print args # tuplas dos argumentos posicionais
4
```

Funções

- Desafio:
  - Alterar a calculadora para que ela possua uma função apenas chamada calcular(1, 2, op=‘soma’)
  - Utilizando o * e **

Funções - Decorators

```
def bolder(func):

    def bolder_func(valor):
        return '<b>' + func(valor) + '</b>'

    return bolder_func

@bolder
def ola(string):
    return 'Ola! ' + string

print ola('Jonathan')
```

Funções - Decorators

- Desafio:
  - Criar uma função que retorne um dicionário de chave e valor como String.
  - Criar um decorator que transforme esse dicionário em json como String.
  - **Sem libs extenas!**

argparse

```
import argparse
parser = argparse.ArgumentParser()

parser.add_argument(
    '--numero1', '-N1', type=int, help="Primeiro numero", required=True)

parser.add_argument(
    '--numero2', '-N2', type=int, help="Segundo numero", required=True)

parser.add_argument(
    '--operador', '-OP', type=str, help="Operador", required=True)

args = parser.parse_args()
print args.numero1
print args.numero2
```

Calculadora com params

- Desafio:

  - Usando o argparse, fazer um programa que receba n1,op e n2 como parâmetros.
  - Exemplo:
  - *python calc.py --numero1 1 --numero2 2 --operador soma*
  - >> 3

Qualidade do código

- Desafio:
  - Trocar os scripts:
    - 1 deve funcionar sem nenhuma alteração.
    - 2 deve ser legível.
    - 3 não podem ocorrer erros de charset, indentação, etc.
    - 4 sugerir melhorias.

Requisitos

- $ sudo apt-get install facter

str

```
$ipython

>> 'terra'.capitalize()
Terra

>> 'terra'.find('rra') # indexOf
2

>> 'terra'.rfind('r') # lastIndexOf
3

>> 'terra'.count('r')
2

>> 'TERRA'.lower()
terra

>> 'terra'.upper()
TERRA
```

str

```
>> '   terra   '.strip() #trim
terra

>> '   terra   '.replace('r', 'X')
teXXa

>> linhas = 'linha1\nlinha2'.split('\n')
print linhas[0]

lista = ['Terra', 'Empresas ']
', '.join(lista)
```

str – Format e %s

```
'Trabalho no {0} em {1}'.format('Terra', 'Porto Alegre')
```

```
'Trabalho no %s em %' % ('Terra', 'Porto Alegre')
```

Strings

- Desafio:
  - Criar um gerador de senhas, onde o usuário digita ‘terra’ e a saída seja ‘T3Rr4’

popen

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  from subprocess import Popen, PIPE
5
6  process = Popen('facter', stdout=PIPE, stderr=PIPE)
7  stdout, stderr = process.communicate()
8
9  print stdout
10
11
```

Verificando o servidor

- Desafio:
  - Usando o Popen com facter e o atributo “is_virtual => true/false”, descobrir de a maquina é virtual ou não.
  - $ python check_vm.py
  - >>> É uma maquina virtual

Verificando o servidor

- Desafio:

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  from subprocess import Popen, PIPE
5  process = Popen('facter', stdout=PIPE, stderr=PIPE)
6  stdout, stderr = process.communicate()
7  linhas = stdout.split('\n')
8
9  ...
10
```

  - Usando o Popen com facter, a função split, colocar os dados do facter em um dicionario.

  - $ python facter.py
  - >>>{ ‘interfaces’: ‘eth1,lo’} (...)

Verificando o servidor parte 2

- Desafio:
  - Utilizando tudo que foi visto anteriormente, criar um script que receba o valor que o facter precisa buscar. Se o valor for 'all', então retorna todos.
  - $ python facter.py –v swapfree
  - >>> *1.89 GB*
  - $ python facter.py –v all
  - >>> *architecture = i386*
  - >>> *domain = corp.terra*
  - *(...)*

Lendo um arquivo de texto

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  arquivo = open('/tmp/teste_de_leitura.txt', 'r')
5  lido = arquivo.read()
6  arquivo.close()
7
8  print lido
9
```

```
with open('workfile', 'r') as fh:
    read_data = f.read()

print read_data
```

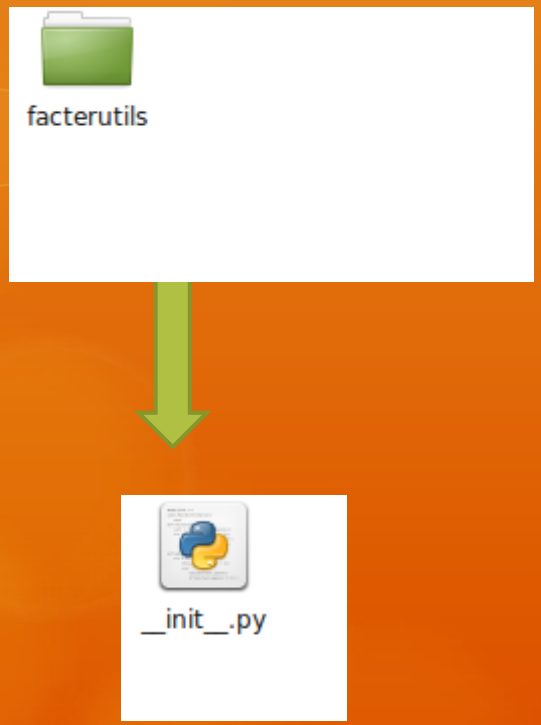
Escrevendo um arquivo de texto

```
1  arquivo = open('/tmp/teste_de_escrita.txt','a')
2  arquivo.write('Ola Python')
3  arquivo.close()
```

- *datetime.datetime.now().strftime("%A, %d. %B %Y %I:%M%p")*

from facterutils import log

- Criando um módulo




```
# -*- coding: utf-8 -*-

def log(msg):
    pass
```

Facter + Log

- Desafio:
  - Adicionar logs ao script facter.py.
  - Deve ser feita uma função def log(msg) no módulo facterutils
  - Deve usar *with*

set()

```
conjunto1 = {1, 2, 3, 4, 5}
conjunto2 = {1, 2}

# testa se os elementos do
# conjunto2 estão no conjunto 1

conjunto2 <= conjunto1

# cria um novo conjunto
# com os elementos que estão
# em qualquer um dos conjuntos

conjunto2 | conjunto1

# cria um novo conjunto
# com os elementos que
# estão nos dois conjuntos

conjunto2 & conjunto1

# cria um conjunto com a diferença
# dos conjuntos

conjunto1 - conjunto2
```

set()

- Desafio:
  - $ wget http://goo.gl/dsc4Jv
  - Abrir o arquivo de texto usando open().
  - Cada linha representa uma lista separada por ponto e virgula. Colocar os nomes em conjuntos.
  - Verificar quais nomes do conjunto 1 aparecem no conjunto 2, porém que não estejam no conjunto 3.

virtualenv

"A Virtual Environment, put simply, is an isolated working copy of Python which allows you to work on a specific project without worry of affecting other projects."

virtualenv

```
$ cd /tmp
$ virtualenv venv
$ source venv/bin/activate

$ pip install simplejson
$ pip freeze

$ deactivate

$ pip freeze
```

OO em python

```
1  class MinhaClasse:
2
3      def get_X(self, X):
4          return 'hello world'
5
6
7  meu_objeto = MinhaClasse()
8  print meu_objeto.get_X
9
10
```

```python
# coding: utf-8

class NumeroRomano():

    def __init__(self, numero):
        self.mapa = {'V': 5, 'X': 10}
        self.numero = self.mapa[numero]
        self.romano = numero

    def get(self):
        return self.numero


num1 = NumeroRomano('X')  # 10
num2 = NumeroRomano('V')  # 5

print num2 > num1  # 5 é maior q 10?
```

OO em python

```
    def __lt__(self, other):   # comportamento do operador <
        return self.get() < other.get()

    def __gt__(self, other):   # comportamento do operador >
        return self.get() > other.get()
```

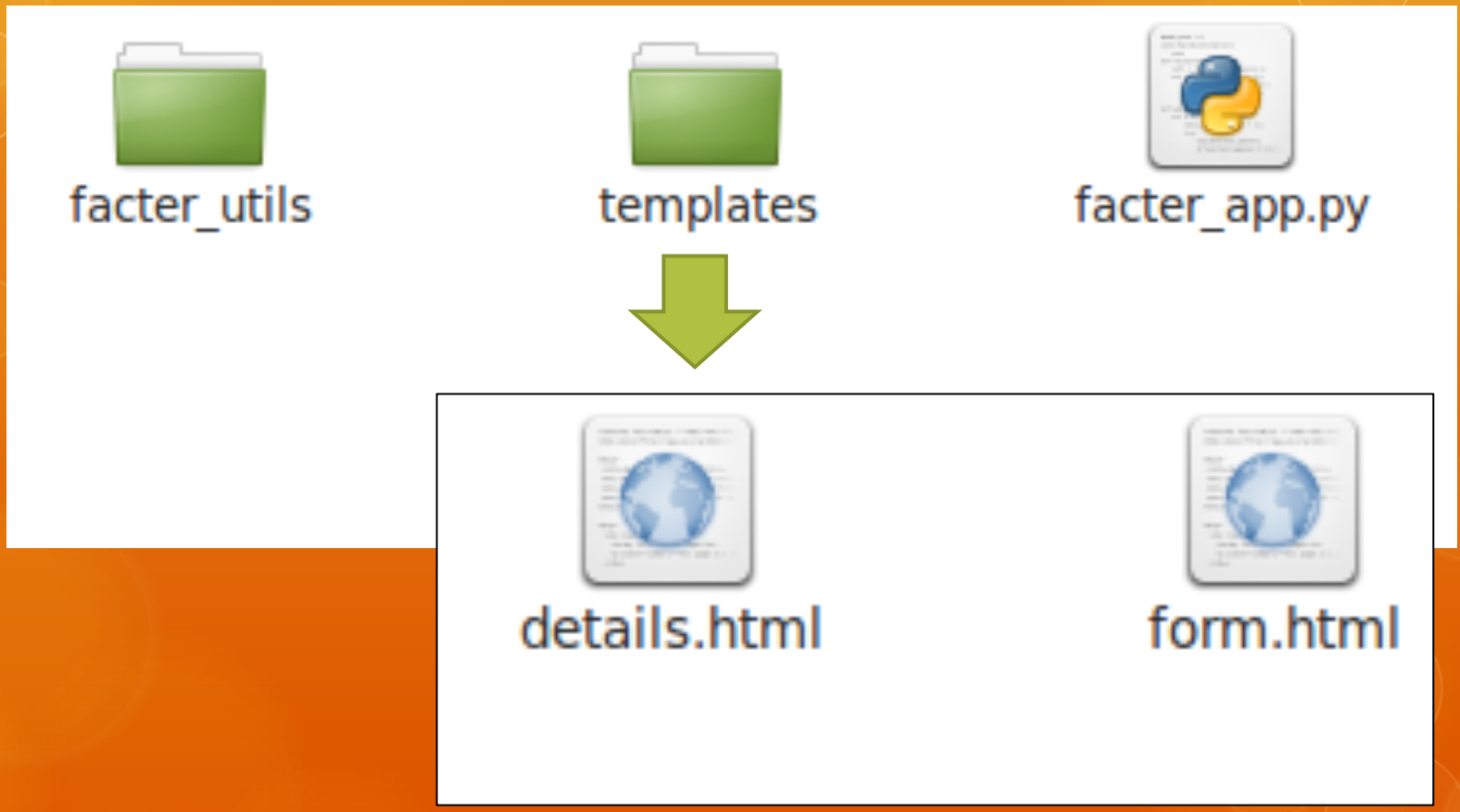
OO

- Desafio:
  - Permitir que a classe seja utilizada em:
    - print int(num1)
    - >> 10
  - Permitir que a classe seja utilizada:
    - print str(num1)
    - >> X

Pré-requisitos

- $ pip install flask

Estrutura de diretório exemplo



Hello World!

```
# -*- coding: utf-8 -*-

from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

app.run()
```

Levantando servidor

```
~/repo/python-course $ python flask_app.py

 * Running on http://127.0.0.1:5000/
 * Restarting with reloader
127.0.0.1 - - [11/Jul/2014 14:26:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Jul/2014 14:26:18] "POST /hello HTTP/1.1" 200 -
```

Utilizando templates

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def hello():
    return render_template('hello.html')

app.run(debug=True)
```

Templates->hello.html

```
{% block body %}
    <h2>Hello World</h2>
{% endblock %}
```

Levantando servidor

```
~/repo/python-course $ python flask_app.py

 * Running on http://127.0.0.1:5000/
 * Restarting with reloader
127.0.0.1 - - [11/Jul/2014 14:26:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Jul/2014 14:26:18] "POST /hello HTTP/1.1" 200 -
```

Adicionando um formulário

```
from flask import Flask, render_template, request
app = Flask(__name__)


@app.route('/')
def index():
    return render_template('form.html')

@app.route('/hello', methods=['POST'])
def hello():
    name = request.form.get('name')
    return render_template('hello.html', name=name)

app.run(debug=True)
```

Templates->hello.html

```
{% block body %}
   <h2>Hello {{ name }}</h2>
{% endblock %}
```

Templates->form.html

```
{% block body %}
  <h2>Hello World</h2>
  <form action="/hello" method=post>
    <dl>
      <dt>Hello who?
      <dd><input type=text name=name>
      <dd><input type=submit value=Hello...>
    </dl>
  </form>
{% endblock %}
```

Levantando servidor

```
~/repo/python-course $ python flask_app.py

 * Running on http://127.0.0.1:5000/
 * Restarting with reloader
127.0.0.1 - - [11/Jul/2014 14:26:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Jul/2014 14:26:18] "POST /hello HTTP/1.1" 200 -
```

Levantando servidor

- Desafio
  - Criar uma aplicação flask que receba uma campo do facter como parâmetro e retorne seu respectivo valor
  - É necessário utilizar a estrutura de templates do flask

Fim!

Obrigado!