# Python gil

## IV Encontro GITEC - 2011

Ramiro Batista da Luz

26 Outubro 2011

Python Ágil - IV Encontro GITEC - 2011.© Ramiro Batista da Luz, 2011.

- Python Ágil - IV Encontro do GITEC - 2011

Python Ágil - IV Encontro GITEC - 2011.© Ramiro Batista da Luz, 2011.

- Python Ágil - IV Encontro do GITEC - 2011
- Ramiro Batista da Luz - ramiroluz@gmail.com

Python Ágil - IV Encontro GITEC - 2011.© Ramiro Batista da Luz, 2011.

- Python Ágil - IV Encontro do GITEC - 2011
- Ramiro Batista da Luz - ramiroluz@gmail.com
- Twitter: @ramiroluz

Python Ágil - IV Encontro GITEC - 2011.© Ramiro Batista da Luz, 2011.

- Python Ágil - IV Encontro do GITEC - 2011
- Ramiro Batista da Luz - ramiroluz@gmail.com
- Twitter: @ramiroluz
- http://www.slideshare.net/ramiroluz/pythonagilivengitec

Python Ágil - IV Encontro GITEC - 2011.© Ramiro Batista da Luz, 2011.

- Python Ágil - IV Encontro do GITEC - 2011
- Ramiro Batista da Luz - ramiroluz@gmail.com
- Twitter: @ramiroluz
- `http://www.slideshare.net/ramiroluz/pythonagilivengitec`
- Códigos exemplo:

Python Ágil - IV Encontro GITEC - 2011.© Ramiro Batista da Luz, 2011.

- Python Ágil - IV Encontro do GITEC - 2011
- Ramiro Batista da Luz - ramiroluz@gmail.com
- Twitter: @ramiroluz
- `http://www.slideshare.net/ramiroluz/pythonagilivengitec`
- Códigos exemplo:
  - `http://www.ramiroluz.eti.br/python-agil/`

Python Ágil - IV Encontro GITEC - 2011.© Ramiro Batista da Luz, 2011.

- Python Ágil - IV Encontro do GITEC - 2011
- Ramiro Batista da Luz - ramiroluz@gmail.com
- Twitter: @ramiroluz
- `http://www.slideshare.net/ramiroluz/pythonagilivengitec`
- Códigos exemplo:
  - `http://www.ramiroluz.eti.br/python-agil/`
  - `http://www.python.org.br/wiki/PythonAgil`

# A avestruz e o gato!

Veloz X Ágil:

Fonte: `http://va.mu/JK27`


Fonte: `http://va.mu/JK3C`

# O que é agilidade?

- Capacidade de mudar de direção rapidamente.

# O que é agilidade?

- Capacidade de mudar de direção rapidamente.
- Habilidade de adaptar-se a mudanças com eficiência.

# O que é linguagem de programação?

- Sequencia de instruções “transformadas” em programas, software aplicativo, sistemas operacionais.

# O que é linguagem de programação?

- Sequencia de instruções “transformadas” em programas, software aplicativo, sistemas operacionais.
- Algumas linguagens: C, C++, Visual Basic, Smalltalk, Haskell, Lua e Python.

# O que é Python?

- Linguagem de programação dinâmica, orientada a objetos, simples e prática.

# O que é Python?

- Linguagem de programação dinâmica, orientada a objetos, simples e prática.
- Usada para criar os sistemas do Interlegis.

# O que é Python?

- Linguagem de programação dinâmica, orientada a objetos, simples e prática.
- Usada para criar os sistemas do Interlegis.
- Usada desde a infraestrutura de servidores até a web, passando por pesquisas científicas e desenvolvimento de jogos bem como linguagem de extenso.

# O que são metodologias ágeis?

- Metodologia de desenvolvimento.

# O que são metodologias ágeis?

- Metodologia de desenvolvimento.
- Alta participação do(s) cliente(s).

# O que são metodologias ágeis?

- Metodologia de desenvolvimento.
- Alta participação do(s) cliente(s).
- Adaptação rápida à mudanças.

# O que são metodologias ágeis?

- Metodologia de desenvolvimento.
- Alta participação do(s) cliente(s).
- Adaptação rápida à mudanças.
- Software funcionando entregue com grande frequência.

# O que são metodologias ágeis?

- Metodologia de desenvolvimento.
- Alta participação do(s) cliente(s).
- Adaptação rápida à mudanças.
- Software funcionando entregue com grande frequência.
- Exemplos: Extreme Programming (XP), Scrum, Crystal, Lean, Feature Driven Development (FDD), Agile Unified Process (Agile UP or AUP), Dynamic Systems Development Method (DSDM).

# Por que Python Ágil?

- Python é fácil para integrar(Bindings C).

# Por que Python Ágil?

- Python é fácil para integrar(Bindings C).
- Possui muitas bibliotecas fáceis de aprender.

# Por que Python Ágil?

- Python é fácil para integrar(Bindings C).
- Possui muitas bibliotecas fáceis de aprender.
- Flexível, se adapta ao ambiente(IronPython, Jython, PyPy).

# Segurança

- Para mudar, para evoluir.

# Segurança

- Para mudar, para evoluir.
- Confiabilidade(Testes de integração).

# Segurança

- Para mudar, para evoluir.
- Confiabilidade(Testes de integração).
- Aceitação de funcionalidades(Testes funcionais).

# A metodologia XGH - eXtreme Go Horse

- http://va.mu/BcW

# A metodologia XGH - eXtreme Go Horse

- `http://va.mu/BcW`
  1. Pensou, não é XGH.

# A metodologia XGH - eXtreme Go Horse

- `http://va.mu/BcW`
  1. Pensou, não é XGH.
  2. Existem 3 formas de se resolver um problema, a correta, a errada e a XGH, que é igual à errada, só que mais rápida.

# A metodologia XGH - eXtreme Go Horse

- `http://va.mu/BcW`
  1. Pensou, não é XGH.
  2. Existem 3 formas de se resolver um problema, a correta, a errada e a XGH, que é igual à errada, só que mais rápida.
  3. Quanto mais XGH você faz, mais vai precisar fazer.

# A metodologia XGH - eXtreme Go Horse

- `http://va.mu/BcW`
  1. Pensou, não é XGH.
  2. Existem 3 formas de se resolver um problema, a correta, a errada e a XGH, que é igual à errada, só que mais rápida.
  3. Quanto mais XGH você faz, mais vai precisar fazer.
  4. XGH é totalmente reativo.

# TDD - Test Driven Development

- http://va.mu/JjxN

# TDD - Test Driven Development

- `http://va.mu/JjxN`
  - Adicione um teste

# TDD - Test Driven Development

- `http://va.mu/JjxN`
  - Adicione um teste
  - Execute todos os testes e veja se algum deles falha

# TDD - Test Driven Development

- `http://va.mu/JjxN`
  - Adicione um teste
  - Execute todos os testes e veja se algum deles falha
  - Escrever código

# TDD - Test Driven Development

- `http://va.mu/JjxN`
  - Adicione um teste
  - Execute todos os testes e veja se algum deles falha
  - Escrever código
  - Execute os testes automatizados e veja-os executarem com sucesso

# TDD - Test Driven Development

- `http://va.mu/JjxN`
  - Adicione um teste
  - Execute todos os testes e veja se algum deles falha
  - Escrever código
  - Execute os testes automatizados e veja-os executarem com sucesso
  - Refatorar código

# TDD - Test Driven Development

- `http://va.mu/JjxN`
  - Adicione um teste
  - Execute todos os testes e veja se algum deles falha
  - Escrever código
  - Execute os testes automatizados e veja-os executarem com sucesso
  - Refatorar código
  - Repita tudo

# BDD - Behavior Driven Development

- http://va.mu/Jjxa

# BDD - Behavior Driven Development

- `http://va.mu/Jjxa`
  - Envolver as partes interessadas no processo através de Outside-in Development (Desenvolvimento de Fora pra Dentro)

# BDD - Behavior Driven Development

- `http://va.mu/Jjxa`
  - Envolver as partes interessadas no processo através de Outside-in Development (Desenvolvimento de Fora pra Dentro)
  - Usar exemplos para descrever o comportamento de uma aplicação ou unidades de código

# BDD - Behavior Driven Development

- `http://va.mu/Jjxa`
  - Envolver as partes interessadas no processo através de Outside-in Development (Desenvolvimento de Fora pra Dentro)
  - Usar exemplos para descrever o comportamento de uma aplicação ou unidades de código
  - Automatizar os exemplos para prover um feedback rápido e testes de regresso

# BDD - Behavior Driven Development

- `http://va.mu/Jjxa`
  - Envolver as partes interessadas no processo através de Outside-in Development (Desenvolvimento de Fora pra Dentro)
  - Usar exemplos para descrever o comportamento de uma aplicação ou unidades de código
  - Automatizar os exemplos para prover um feedback rápido e testes de regresso
  - Usar deve (should em inglês) na hora de descrever o comportamento de software para ajudar esclarecer responsabilidades e permitir que funcionalidades do software sejam questionadas

# BDD - Behavior Driven Development

- `http://va.mu/Jjxa`
  - Envolver as partes interessadas no processo através de Outside-in Development (Desenvolvimento de Fora pra Dentro)
  - Usar exemplos para descrever o comportamento de uma aplicação ou unidades de código
  - Automatizar os exemplos para prover um feedback rápido e testes de regresso
  - Usar deve (should em inglês) na hora de descrever o comportamento de software para ajudar esclarecer responsabilidades e permitir que funcionalidades do software sejam questionadas
  - Usar dublês de teste (mocks, stubs, fakes, dummies, spies) para auxiliar na colaboração entre módulos e códigos que ainda no foram escritos

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - http://va.mu/Jjxq

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`
- Generators: generators.py

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`
- Generators: generators.py
  - `http://va.mu/JjyJ`

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`
- Generators: generators.py
  - `http://va.mu/JjyJ`
- Compreensão de listas: compreensao_listas.py

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`
- Generators: generators.py
  - `http://va.mu/JjyJ`
- Compreensão de listas: compreensao_listas.py
- Singleton:

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`
- Generators: generators.py
  - `http://va.mu/JjyJ`
- Compreensão de listas: compreensao_listas.py
- Singleton:
  - python-3-patterns-idioms/code/Singleton/SingletonPattern.py

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`
- Generators: generators.py
  - `http://va.mu/JjyJ`
- Compreensão de listas: compreensao_listas.py
- Singleton:
  - python-3-patterns-idioms/code/Singleton/SingletonPattern.py
- Factory:

# Padrões de projeto(Design Pattern)

- Decorators: decoradores.py
  - python-3-patterns-idioms/code/PythonDecorators/entry_exit_class.py
  - `http://va.mu/Jjxq`
- Iterators: iteradores.py
  - `http://va.mu/Jjx6`
- Generators: generators.py
  - `http://va.mu/JjyJ`
- Compreensão de listas: compreensao_listas.py
- Singleton:
  - python-3-patterns-idioms/code/Singleton/SingletonPattern.py
- Factory:
  - python-3-patterns-idioms/code/Factory/shapefact1/ShapeFactory1.py

# Ferramentas

- PyUnit: `http://va.mu/oMf`

# Ferramentas

- PyUnit: `http://va.mu/oMf`
- DocTest: `http://va.mu/Jjyb`

# Ferramentas

- PyUnit: `http://va.mu/oMf`
- DocTest: `http://va.mu/Jjyb`
- Nose+TDaemon:

# Ferramentas

- PyUnit: `http://va.mu/oMf`
- DocTest: `http://va.mu/Jjyb`
- Nose+TDaemon:
  - `http://va.mu/Jjyh`

# Integração Continua(Continuos Integration)

- GetWindmill: `http://va.mu/Jjyy`

# Integração Continua(Continuos Integration)

- GetWindmill: `http://va.mu/Jjyy`
- Martin Fowler, continuos integration: `http://va.mu/Jjy2`

# Integração Continua(Continuos Integration)

- GetWindmill: `http://va.mu/Jjyy`
- Martin Fowler, continuos integration: `http://va.mu/Jjy2`
- TeamCity: `http://va.mu/JjzD`

# Integração Continua(Continuos Integration)

- GetWindmill: `http://va.mu/Jjyy`
- Martin Fowler, continuos integration: `http://va.mu/Jjy2`
- TeamCity: `http://va.mu/JjzD`
- Skink: `http://va.mu/Jjzx`

# Integração Continua(Continuos Integration)

- GetWindmill: `http://va.mu/Jjyy`
- Martin Fowler, continuos integration: `http://va.mu/Jjy2`
- TeamCity: `http://va.mu/JjzD`
- Skink: `http://va.mu/Jjzx`
- Jenkins(Hudson): `http://jenkins-ci.org/`

# Demonstração

- Testes Unitários: testes_unitarios.py

# Demonstração

- Testes Unitários: testes_unitarios.py
- Testes e Documentação(DocTests): doctests.py

# Demonstração

- Testes Unitários: testes_unitarios.py
- Testes e Documentação(DocTests): doctests.py
- Testes com Django:

# Demonstração

- Testes Unitários: testes_unitarios.py
- Testes e Documentação(DocTests): doctests.py
- Testes com Django:
  - Django sem desculpas: `http://va.mu/JjOL`

# Conclusão

- Mais referências:

# Conclusão

- Mais referências:
  - Osvaldo Santana, TDD com Python: `http://va.mu/Jj0f`

# Conclusão

- Mais referências:
  - Osvaldo Santana, TDD com Python: `http://va.mu/Jj0f`
  - Rodrigo Alves Vieira, `http://va.mu/Jj0s`

# Conclusão

- Mais referências:
  - Osvaldo Santana, TDD com Python: `http://va.mu/Jj0f`
  - Rodrigo Alves Vieira, `http://va.mu/Jj0s`
  - Plone app testing: `http://va.mu/Jj04`

# Conclusão

- Mais referências:
  - Osvaldo Santana, TDD com Python: `http://va.mu/Jj0f`
  - Rodrigo Alves Vieira, `http://va.mu/Jj0s`
  - Plone app testing: `http://va.mu/Jj04`
  - Exemplos de testes com web2py: `http://va.mu/Jj1F`

# Conclusão

- Mais referências:
  - Osvaldo Santana, TDD com Python: `http://va.mu/Jj0f`
  - Rodrigo Alves Vieira, `http://va.mu/Jj0s`
  - Plone app testing: `http://va.mu/Jj04`
  - Exemplos de testes com web2py: `http://va.mu/Jj1F`
  - Test unitário com web2py: `http://va.mu/Jj1T`

# Conclusão

- Mais referências:
  - Osvaldo Santana, TDD com Python: `http://va.mu/Jj0f`
  - Rodrigo Alves Vieira, `http://va.mu/Jj0s`
  - Plone app testing: `http://va.mu/Jj04`
  - Exemplos de testes com web2py: `http://va.mu/Jj1F`
  - Test unitário com web2py: `http://va.mu/Jj1T`
  - Test unitário com pylons: `http://va.mu/Jj1p`

# Agradecimentos

- Bruce Eckel

# Agradecimentos

- Bruce Eckel
- Adriano Petrich

# Agradecimentos


- Bruce Eckel
- Adriano Petrich
- Rodrigo Bernardo Pimentel

# Agradecimentos


- Bruce Eckel
- Adriano Petrich
- Rodrigo Bernardo Pimentel
- Davi Lima

# Agradecimentos


- Bruce Eckel
- Adriano Petrich
- Rodrigo Bernardo Pimentel
- Davi Lima
- Angelo Marcondes, Júnior(Ishida), Jean Ferri e comunidade Interlegis

# Contato e perguntas

- e-mail e gtalk: ramiroluz@gmail.com

# Contato e perguntas

- e-mail e gtalk: ramiroluz@gmail.com
- twitter: @ramiroluz

# Encerramento

- http://www.python.org.br

# Encerramento

- http://www.python.org.br
- http://associacao.python.org.br

# Encerramento

- http://www.python.org.br
- http://associacao.python.org.br
- http://groups.google.com/group/grupy-pr