

# Como Python pode ajudar no desenvolvimento ágil de software

---

Rebeca Sarai

# Quem sou eu

- Rebeca Sarai
  - Recife
  - Estudante de Engenharia da Computação - UPE
  - Torcedora do ~~melhor time de Pernambuco~~ Náutico
  - Organizadora do Django Girls Recife
  - 1 ano como desenvolvedora

@rsarai007

[github.com/rsarai](https://github.com/rsarai)



The background image shows The Flash in his red and yellow suit, running at high speed through a dark, stormy environment. He is surrounded by bright yellow and orange lightning bolts and energy trails, suggesting immense speed. The sky is filled with dark, heavy clouds, and the ground below is a mix of dark and light patches, possibly representing a battlefield or a city street. The overall tone is dramatic and action-packed.

**Ágil**

# O que é Ágil?

- Que se movimenta com excesso de facilidade; que se move de maneira **rápida; veloz**.
- Que se comporta ou trabalha de maneira **eficaz e rápida**; diligente, expedito e trabalhador.
- **Habilidade de criar e responder a mudanças** em um ambiente incerto e turbulento, com o objetivo de obter sucesso.

# Desenvolvimento Ágil de Software?



© Scott Adams, Inc./Dist. by UFS, Inc.



TRADUÇÃO LIVRE: TIRINHAS.COM

Não exatamente!



# Desenvolvimento Ágil de Software



É um termo genérico para um conjunto de métodos e práticas baseadas nos valores e princípios expressos no Manifesto Ágil.



# Manifesto Ágil

# História

- **1970** - Metodologias tradicionais
- **1990** - Métodos leves (Lightweight Methods) começaram a chamar atenção
- **2001 - 17** desenvolvedores se reuniram para discutir abordagens de desenvolvimento de software
- Vários conceitos presentes na metodologias ágeis foram refinados e incorporados pela comunidade.

# Valores



**Indivíduos e interações** mais que processos e ferramentas

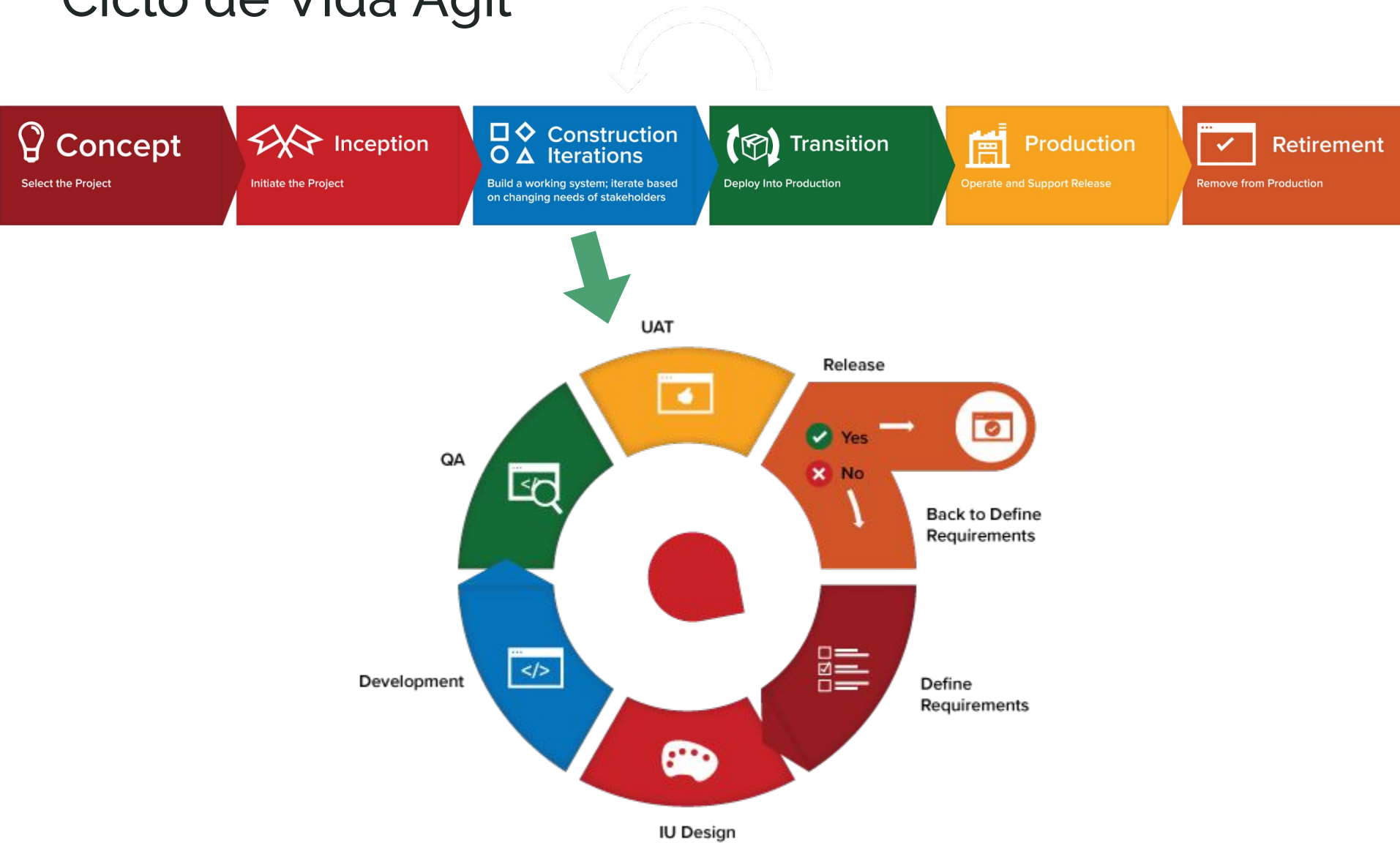
**Software em funcionamento** mais que documentação abrangente

**Colaboração com o cliente** mais que negociação de contratos

**Responder a mudanças** mais que seguir um plano

...e 12 princípios correspondentes

# Ciclo de Vida Ágil



# Metodologias Ágeis

- Scrum, XP, Kanban
- Ou uma customização para realidade da empresa. Processos customizados podem gerar os mesmos benefícios.

VINTA

HOME

BLOG

WORK

TEAM

PLAYBOOK

KNOWLEDGE

Vinta's Playbook

Plans and practices

CONTACT US

**Indivíduos e interações**



processos e ferramentas

# Indivíduos e interações

## ❏ Princípios

- **Princípio 6** - O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
- **Princípio 5** - Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.

## ❏ Práticas Ágeis

- Pair Programming
- Daily meeting
- Code Review

| Como Python entra nisso?



# Simplicidade



0 1 0 1 0  
1 0 1 0 1  
0 1 0 1 0  
1 0 1 0 1  
0 1 0 1 0  
1 0 1 0 1  
0 1 0 1 0  
1 0 1 0 1  
0 1 0 1 0  
1 0 1 0 1  
0 1 0 1 0  
1 0 1 0 1  
0 1 0 1 0

# Python promove simplicidade

- ❑ Syntax limpa e simples
- ❑ Tipos de dados de alto nível incorporados
  - Strings, Lists, Tuples, Dictionaries
- ❑ The Zen of Python
  - Simples é melhor que complexo
  - Explícito é melhor que implícito.

# List Comprehension

JAVA

PYTHON

```
numeros = [1, 2, 3, 4, 5]

# é bom
dobrar_chances = []
for n in numeros:
    if n % 2 == 1:
        dobrar_chances.append(n
* 2)

# é melhor
dobrar_chances = [n * 2 for n
in numeros if n % 2 == 1]
```

---

# Duck Typing

Se caminha como um pato, nada como um pato e grasna como um pato, provavelmente é um pato.

```
class Pato:
    def quack(self):
        print("Quack, quack!")

    def fly(self):
        print("Flap, Flap!")
```

```
class Pessoa:
    def quack(self):
        print("I'm Quackin'!")

    def fly(self):
        print("I'm Flyin'!")

def na_floresta(mallard):
    mallard.quack()
    mallard.fly()
```

```
def main():
    na_floresta(Pato())
    na_floresta(Pessoa())
```

```
>>> main()

>>> Quack, quack!
>>> Flap, Flap!
>>> Quackin'!
>>> Flyin'!
```

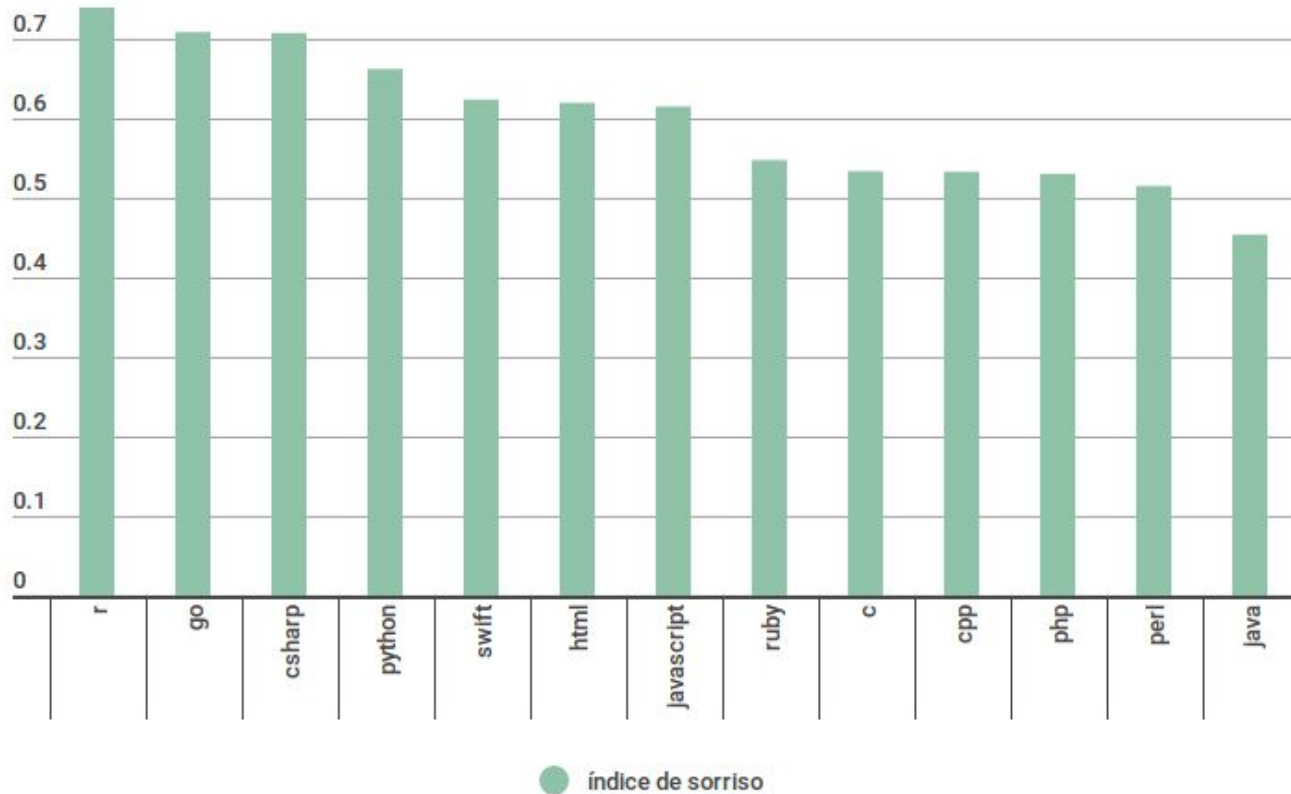


Comunicação

# Python promove **comunicação**

- ❑ Idiomas poderosos e simples que permitem aos desenvolvedores comunicarem claramente suas intenções através do código
- ❑ Estilo de codificação padrão -- [PEP8](#)
- ❑ A sessão interativa do shell fornece feedback instantâneo
- ❑ Python deixa você programar e não entra no seu caminho

Então programadores Python são mais motivados?



**Software em  
funcionamento**



documentação  
abrangente



# Software em funcionamento

## ❑ Princípios

- **Princípio 3** - Entregar software funcionando com frequência, na escala de semanas até meses, com preferência em períodos mais curtos.

## ❑ Práticas Ágeis

- Desenvolvimento iterativo
- Desenvolvimento incremental
- Sprints
- Integração contínua

# Integração contínua



A Integração Contínua (CI) é uma prática de desenvolvimento que exige que os desenvolvedores integrem código em um repositório compartilhado várias vezes ao dia. Cada check-in é verificado por uma compilação automatizada.

# django-react-boilerplate

 vintasoftware / **django-react-boilerplate**

 Unwatch ▾

18

★ S

<> Code

! Issues 34

🔗 Pull requests 1

⚙ Settings

Insights ▾

Django, React, Bootstrap 4 with Python 3 and webpack project boilerplate

django

react

redux

webpack

python

bootstrap4

Manage topics

🕒 135 commits

🌿 23 branches

📦 0 releases

👤 10 contributors

Branch: master ▾

New pull request

Create new file

Upload files

Find file







**rsarai** committed on **GitHub** Merge pull request [#102](#) from vintasoftware/updates-post-compile ...

Latest co


# django-react-boilerplate



o-react-boilerplate

 <b>FIXED</b>	feature/add-more-linters #82  Updated prospector config	 3 days ag  #104
 <b>FAILED</b>  rebuild	feature/add-more-linters #81  Switched from flake8 to prospector	 3 days ag
 <b>SKIPPED</b>	boilerplate-release #80  Replacing circle.yml	 -
 <b>SUCCESS</b>	master #79  Merge pull request #102 from vintasoftware/updates-post-...	 6 days ag
 <b>SUCCESS</b>	updates-post-compile #78  Review pull request	 6 days ag  #102





*“Integração contínua não se livra dos erros,  
mas torna-os dramaticamente mais fáceis  
de encontrar e remover.”*

*— Martin Fowler*

| Como Python entra nisso?

# Integração contínua em Python

## ❑ Buildbots

- Conjunto de máquinas dedicadas (ou build slaves) usadas para a integração contínua. Eles abrangem uma série de combinações de hardware / sistema operacional.

## ❑ [Python Developer's Guide](#)

## ❑ The Zen of Python

- Erros nunca devem passar silenciosamente.

**Colaboração com o  
cliente**



**negociação de contratos**



# Colaboração com o cliente

## ❑ Princípios

- **Princípio 1** - Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
- **Princípio 2** - Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
- **Princípio 7** - Software funcional é a medida primária de progresso.

## ❑ Práticas Ágeis

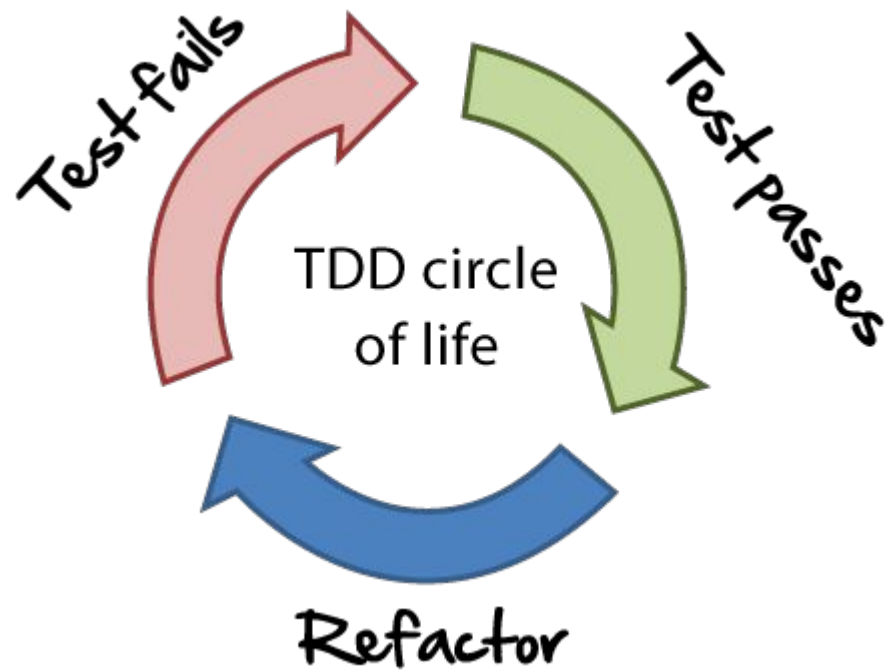
- User Stories
- TDD

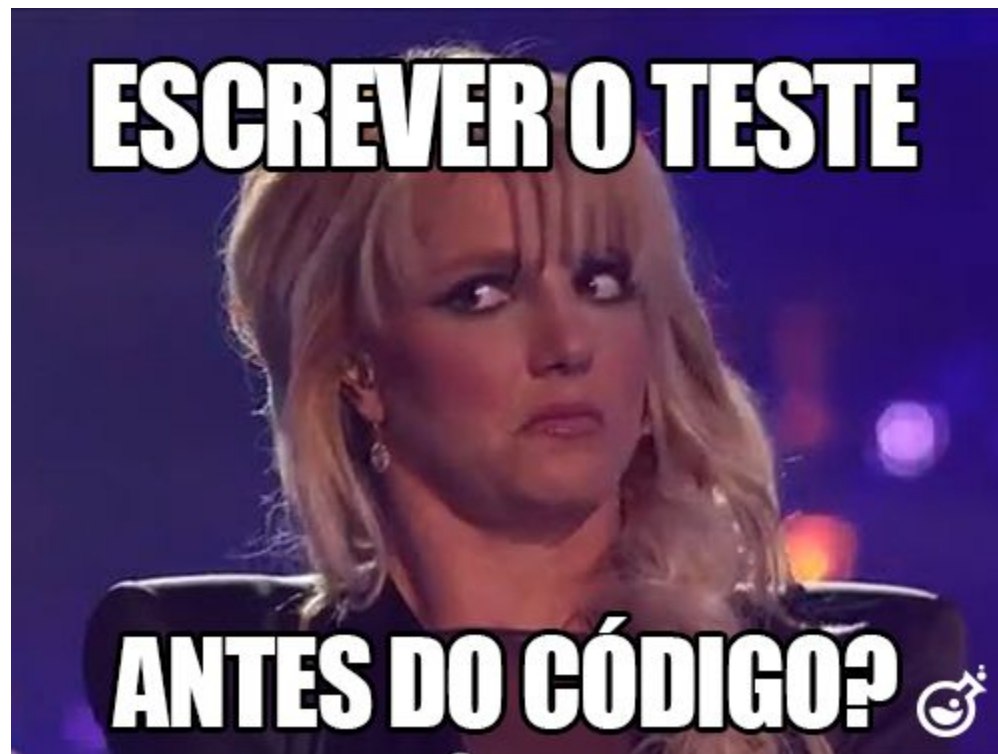
# TDD



Desenvolvimento orientado a teste é um estilo de programação em que três atividades estão estreitamente interligadas: codificação, teste (na forma de unit tests) e design (sob a forma de refatoração).

# TDD





| Como Python entra nisso?

# Python promove TDD

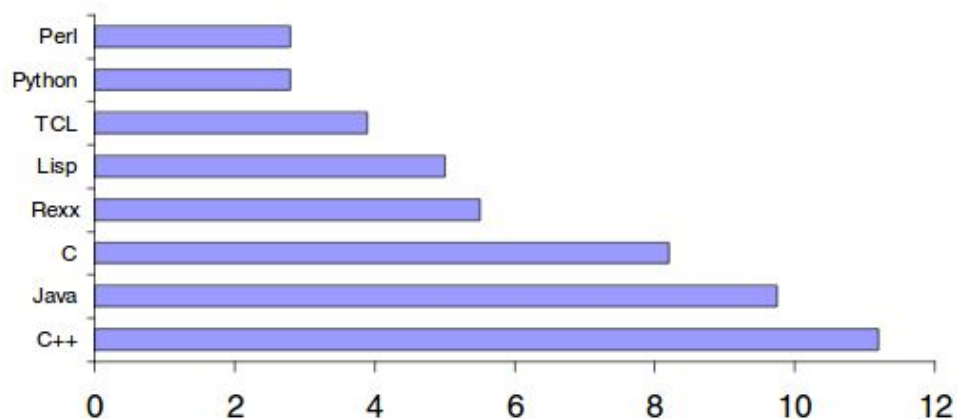
- ❑ Vários frameworks de testes (unittest, doctest, py.test) estão disponíveis para feedback
  
- ❑ The Zen of Python
  - Casos especiais não são especiais o bastante para quebrar as regras.
  - Diante da ambiguidade, recuse a tentação de adivinhar.
  - Deve haver um — e preferencialmente só um — modo óbvio para fazer algo.

# Produtividade



# Python promove produtividade

- Não significa velocidade
- O recurso mais caro de uma empresa é agora o tempo do seu empregado



Horas para escrever um aplicativo de processamento de sequência de caracteres em várias linguagens



**Responder a mudanças**



seguir um plano

# Responder a mudanças

## ❑ Princípios

- **Princípio 9** - Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
- **Princípio 10** - Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.

## ❑ Práticas Ágeis

- Refatoração



Agilidade

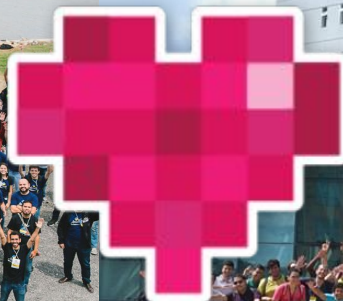
# Python promove agilidade

- ❏ Desenvolvimento rápido
- ❏ Riqueza de pacotes de terceiros
  - [matplotlib](#)
  - [django-recurrence](#)
  - [arrow](#)
  - [pandas](#)
  - [tensorflow](#)

# Comunidade







django girls

**Slides:**

**[bit.ly/vinta-pyne-17](https://bit.ly/vinta-pyne-17)**



Twitter: <https://twitter.com/rsarai007>



GitHub: <https://github.com/rsarai>



Email: [rebeca@vinta.com.br](mailto:rebeca@vinta.com.br)