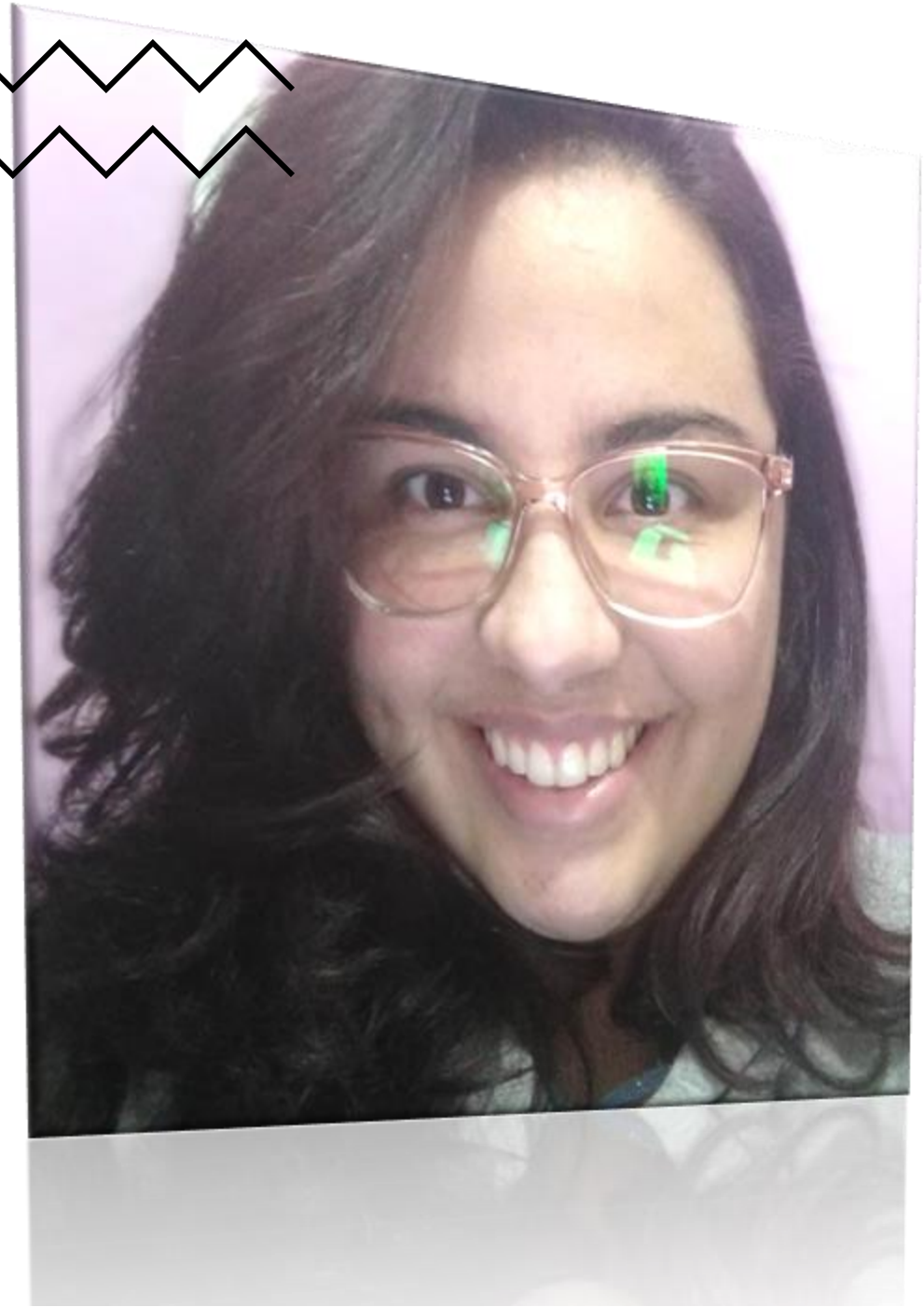

PYTHON BR4S1L 2020
CARREGAN

COMO SUBMETER UMA PEP PARA A COMUNIDADE ;-PYTHON

DANIELE NAZARÉ TAVARES



Sobre Mim

Sou Mineirinha e Bacharel em Engenharia de Computação e contribuo para a comunidade de desenvolvedores ministrando palestra em eventos. Já tive a oportunidade de palestrar na TDC, Brazil JS On the Road, Python Brazil 2019.

https://br.linkedin.com/in/danielenazare
https://github.com/danynt14

Python Enhancement Proposals
PEPs

As PEP's são arquivos que são liberados pela comunidade para apresentar alguma funcionalidade nova, uma boa prática, um propósito ou um ambiente

(PEP 1)

PEPs

"As PEPs são Documentos Equivalente as RFCs, Porém com Propósitos Diferentes "

RFC

python
SOFTWARE FOUNDATION

Quem é responsável pela Curadoria das PEPs na comunidade?

Não sou só eu, é o ***PEP-Advogate*** também ;-D

Quem é responsável pela Curadoria das PEPs na comunidade?

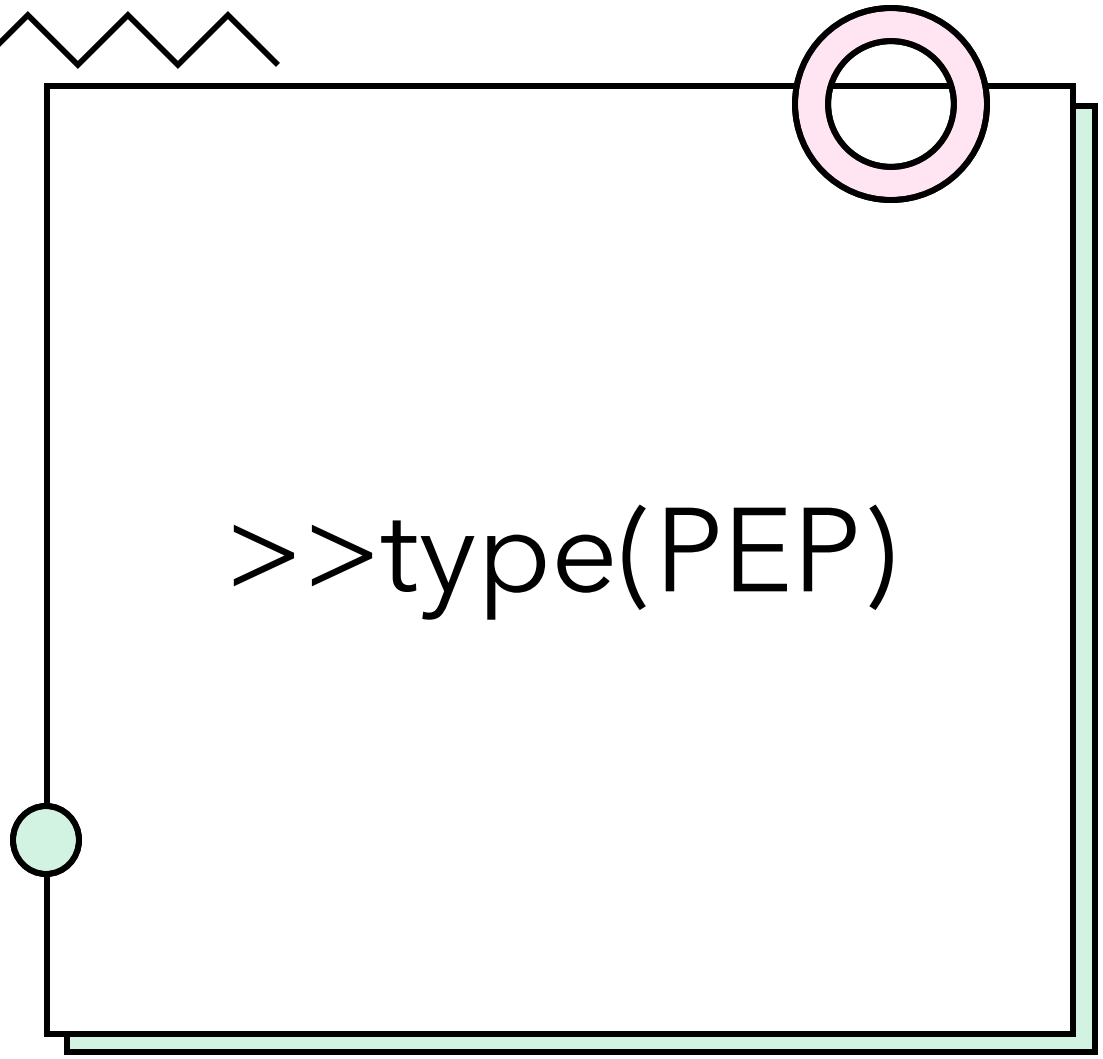
E AGORA?
POR ONDE COMEÇAR?

1
- Você tem uma idéia?

2
- Recebeu uma aceitação geral da Comunidade?

Se sim
- Construa a Sua Ideia

Se não
- Abra uma discussão em um fórum para entender as necessidades da comunidade



- Normas ou Padrões
  - Novas características ou implementação
- Informacional
  - Problema de projeto, diretrizes gerais ou uma informação
- Processo
  - Processo relacionado ao Python

>>def start():

- Os detalhes de como escrever uma PEP estão registrados na PEP 1
- Repósitório onde são mantidas as PEPs:
  - https://github.com/python/peps

ARQUITETURA DE UMA PEP

- Codificado com arquivo de texto UTF-8
- Usa o arquivo de texto no formato reSructuredText
  - Detalhes na PEP12
  - Pip install docutils
- Composta por:
  - Cabeçalho
  - Corpo
    - O conteúdo da PEP

CABEÇALHO

PREÂMBULO NO ESTILO RFC822

```
  PEP: <pep number>
  Title: <pep title>
  Author: <list of authors' real names and optionally, email addrs>
* Sponsor: <real name of sponsor>
* PEP-Delegate: <PEP delegate's real name>
* Discussions-To: <email address>
  Status: <Draft | Active | Accepted | Provisional | Deferred | Rejected |
           Withdrawn | Final | Superseded>
  Type: <Standards Track | Informational | Process>
* Content-Type: <text/x-rst | text/plain>
* Requires: <pep numbers>
  Created: <date created on, in dd-mmm-yyyy format>
* Python-Version: <version number>
  Post-History: <dates of postings to python-ideas and/or python-dev>
* Replaces: <pep number>
* Superseded-By: <pep number>
* Resolution: <url>
```

ESTRUTURA

PEP:	20
Title:	The Zen of Python
Author:	tim.peters at gmail.com (Tim Peters)
Status:	Active
Type:	Informational
Created:	19-Aug-2004
Post-History:	22-Aug-2004

DETALHES DO CABEÇALHO

- **Índice**
  - Número no formado XXXX
- **Título**
- **Nome dos Autores**
  - Nome do Autor < email@email.com >
- **Data de Criação**
- **Histórico de atualização**
- **Tipo**
  - **I** – Informacional
  - **P** – Processo
  - **S** – Normas
- ***Status***
  - A – aceito
  - D - deferido
  - F – proposta final
  - P - propósta provisória
  - R – proposta rejeitada
  - S – Proposta Suspendida
  - W – Propostas retiradas

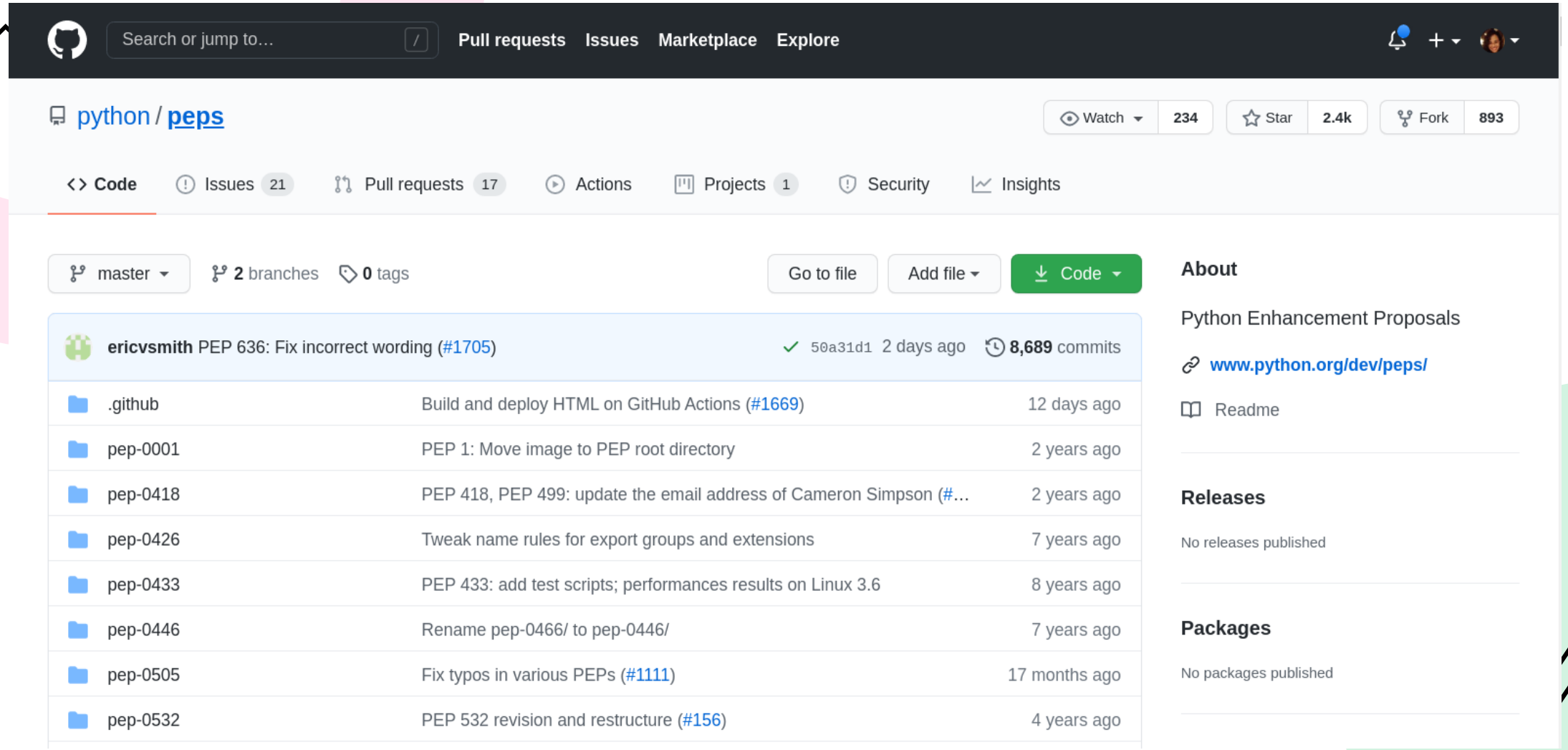
THE ZEN OF PYTHON

PEP 20

```
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

PULL REQUEST NO REPOSITÓRIO



FORMATO DA PEP

pep-9999.rst

reStructuredText

Markup Syntax and Parser Component of Docutils

Date: 2016-05-24

Note

"reStructuredText" is **ONE** word, *not two!*

CURADORIA

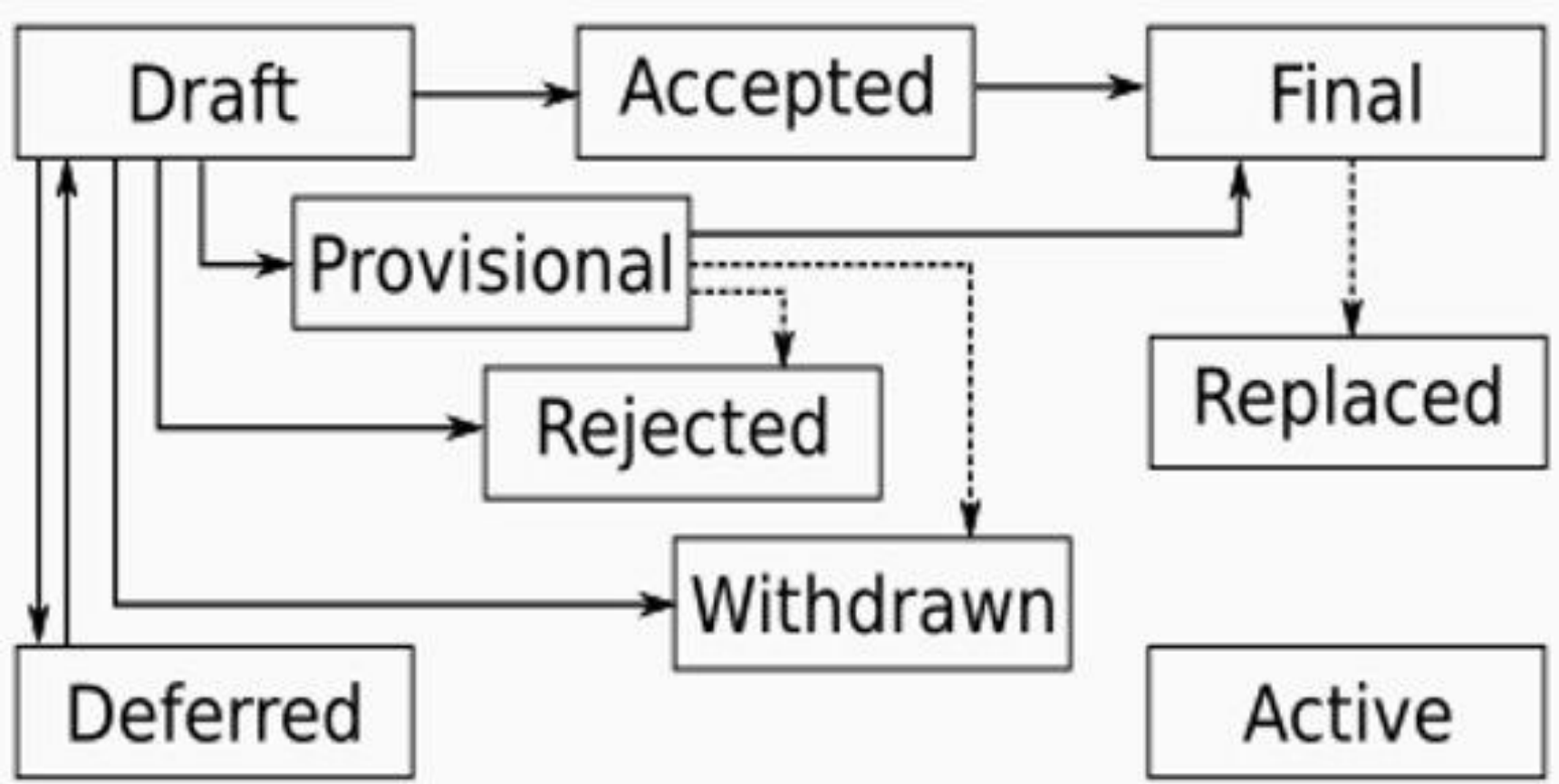
- A curadoria vai verificar o formato da PEP através do TRAVIS CI, antes de ser aprovada
- Se caso for aprovada, a PSF irá enviar um as revisões para o co-autor da pep

APROVAÇÃO

Depois de aprovada

será atribuida um número

FLUXO DA PEP


Draft
Accepted
Final
Provisional
Rejected
Replaced
Withdrawn
Deferred
Active

COMO TER SUCESSO EM UMA PEP?

- Foco a um assunto específico
- Pequenos melhoramentos ou gambiarras não precisam de PEP.
- Examinar a ideia da PEP com a comunidade(Pesquisa na Internet nem sempre resolve)
- Primeiro Apresentar o esboço da PEP para a python-ideas@python.org
- Lista de discussão: python-list@python.org

PEPS OS PYTHONEIROS DEVEM LER

- PEP8 – Guia de Estilo de Python
- PEP 7 – Guia de estilo para códigos em C
- PEP257 - Convenção de *Docstrings*
- PEP 3099 – O que não deve ser modificado em Python
- PEP20 – The Zen of Python

VAMOS KKKKK?

pip install that

THE ANTI-ZEN OF PYTHON
>.<

```
lubuntu@lubuntu-20aws28201:~$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import that
The Anti-Zen of Python, by Daniel Greenfeld

Ugly is better than beautiful.
Implicit is better than explicit.
Complicated is better than complex.
Complex is better than simple.
Nested is better than flat.
Dense is better than sparse.
Line code counts.
Special cases are special enough to break the rules.
Although purity beats practicality.
Errors should always pass silently.
Spelchek iz fur loosers.
In the face of ambiguity, one guess is as good as another.
There should be many ways to do it.
Because only a tiny minority of us are Dutch.
Get things running, then fix them later.
If the implementation is hard to explain, it's enterprisey.
If the implementation is easy to explain, it won't take enough time to do.
Namespaces are too hard, just use "from module import *"!
>>>
```

PYTHON ❤️

> . <

JavaScript

Eva uses **JavaScript** to create funny animations

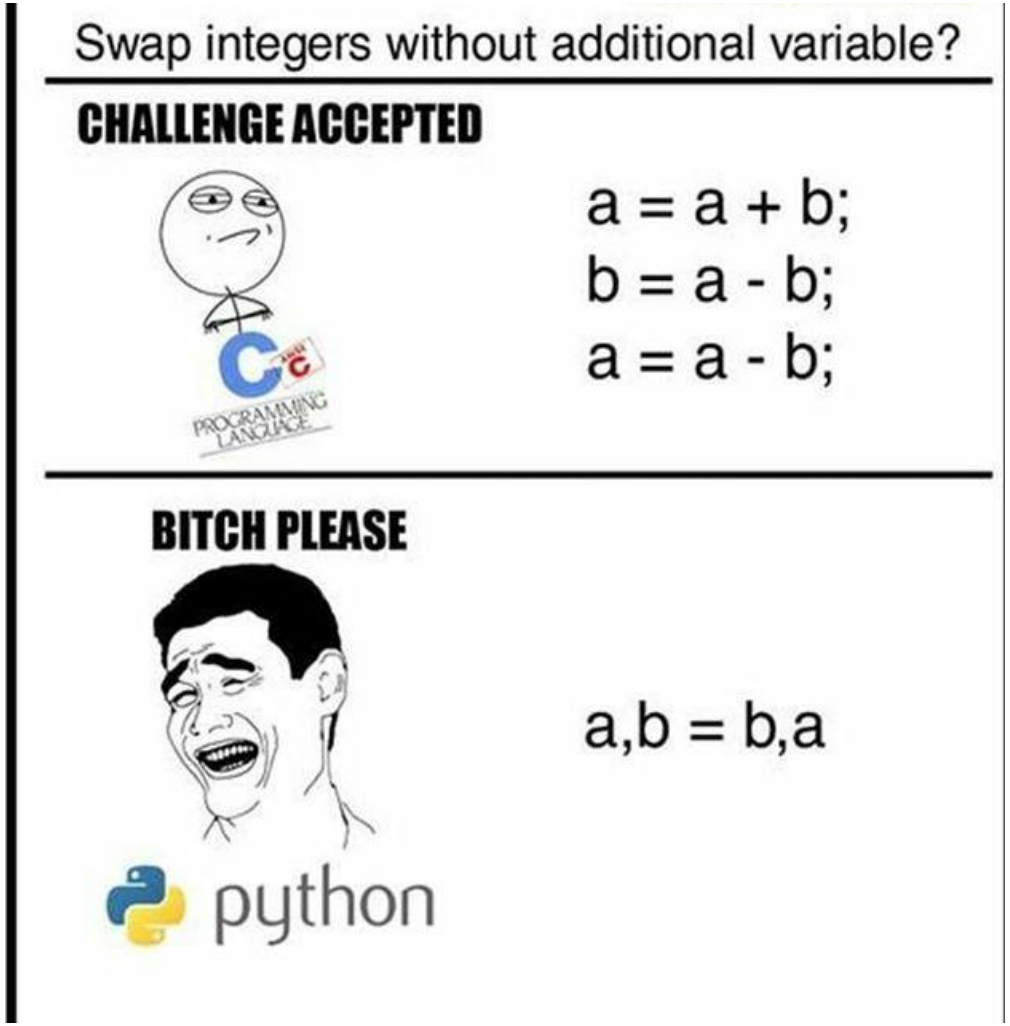
Python

Pixar uses **Python** to create amazing movies.

Por: codeconde.com

P Y T H O N

> . <



Por:Henrique Faísca

REFERÊNCIAS

- [Python Software Fundation]. Disponível em: <https://www.python.org/dev/peps/pep-0001/>. Acessado em: 31 nov de 2020.
- [Python Software Fundation]. Disponível em: <https://www.python.org/dev/peps/pep-0000/>. Acessado em: 31 nov de 2020.
- [Euro Python]. Disponível em: <https://ep2019.europython.eu/talks/LEhyYaK-pep-yourself-10-peps-you-should-pay-attention-to/>. Acessado em: 31 nov de 2020.

MUITO OBRIGADO COMUNIDADE