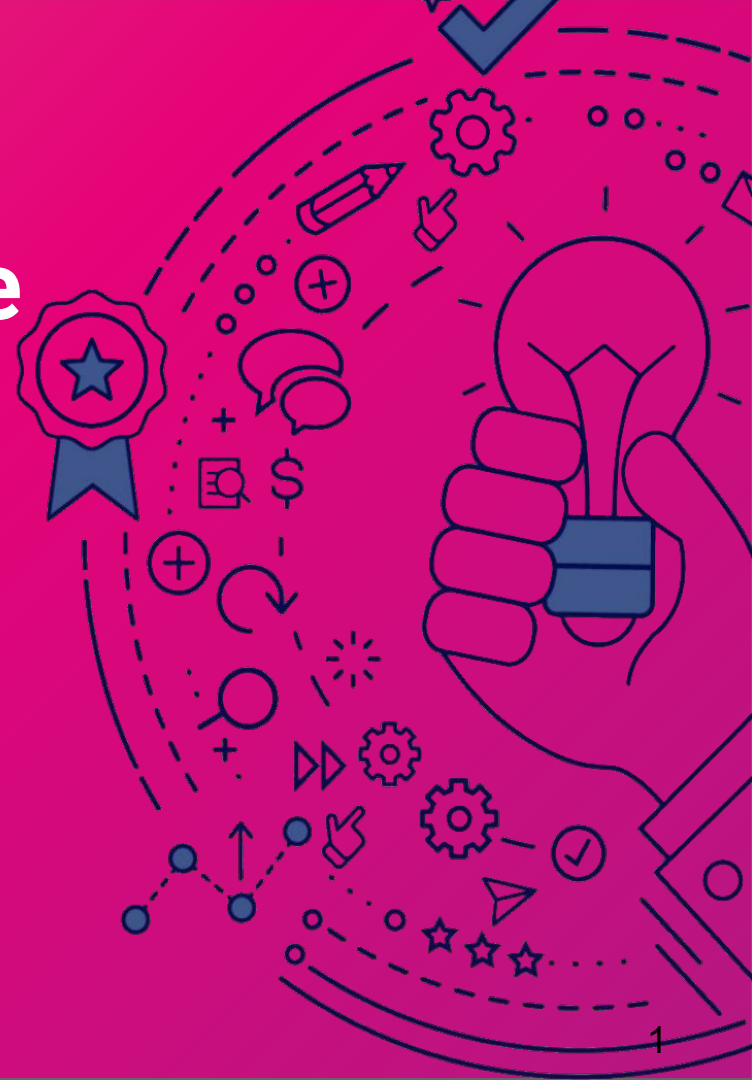


# Código Legado de Machine Learning?

## Como migramos para Python 3 sem sofrimento

Outubro - 2019 - São Paulo  
Eder Martins - Data Scientist



# Quem sou eu?



- Senior Data Scientist na SEEK
- Mestre em ciência da computação pela UFMG
- Trabalho com aplicação de machine learning em problemas de recomendação e busca

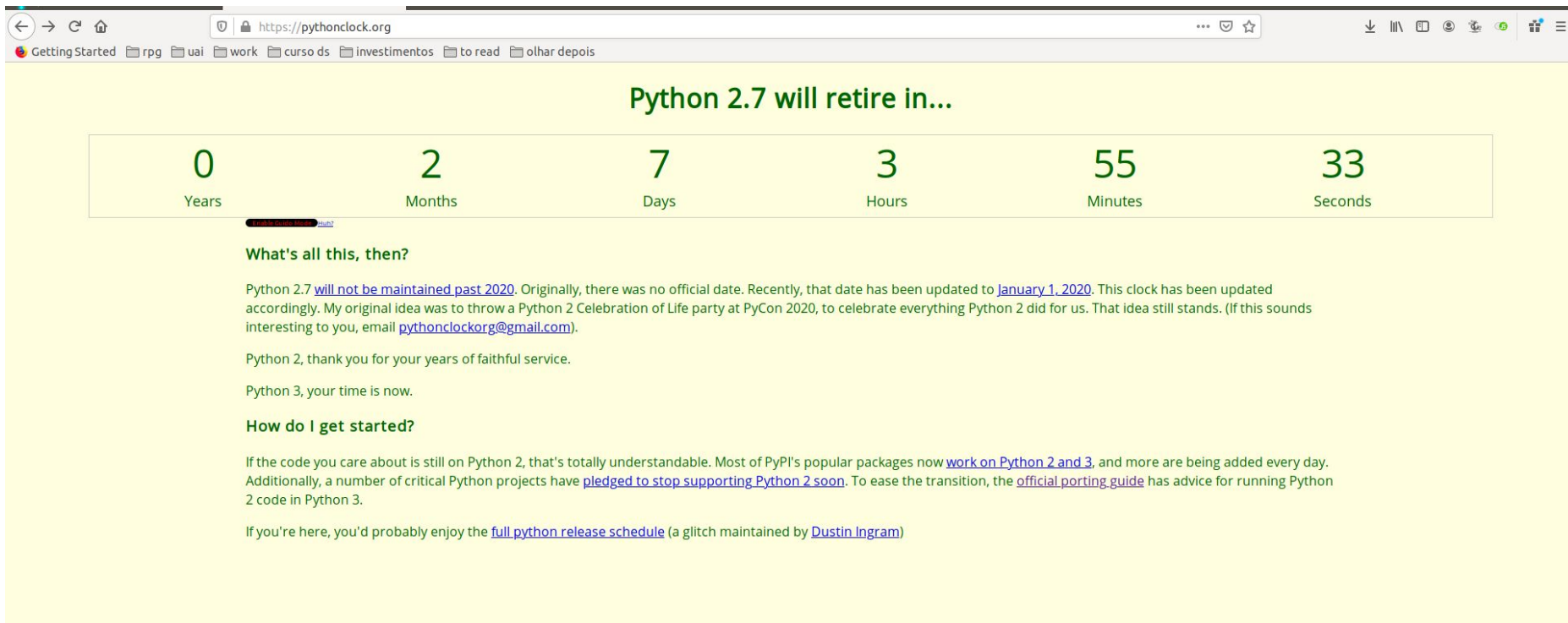


[github.com/ederfmartins](https://github.com/ederfmartins)



<https://br.linkedin.com/in/ederfmartins>

# Qual o objetivo dessa palestra



The screenshot shows a web browser window with the address bar displaying `https://pythonclock.org`. The browser's file explorer shows folders like 'Getting Started', 'rpg', 'uai', 'work', 'curso ds', 'investimentos', 'to read', and 'olhar depois'. The website content has a yellow background and features a large green title 'Python 2.7 will retire in...'. Below the title is a horizontal bar divided into six segments representing time units: Years (0), Months (2), Days (7), Hours (3), Minutes (55), and Seconds (33). A small red progress bar is visible under the 'Years' segment. Below the timer, the text 'What's all this, then?' is followed by a paragraph explaining the retirement date as January 1, 2020, and providing contact information. Further down, the text 'How do I get started?' is followed by instructions on how to transition from Python 2 to Python 3, including links to a release schedule and a porting guide.

**Python 2.7 will retire in...**

0	2	7	3	55	33
Years	Months	Days	Hours	Minutes	Seconds

**What's all this, then?**

Python 2.7 [will not be maintained past 2020](#). Originally, there was no official date. Recently, that date has been updated to [January 1, 2020](#). This clock has been updated accordingly. My original idea was to throw a Python 2 Celebration of Life party at PyCon 2020, to celebrate everything Python 2 did for us. That idea still stands. (If this sounds interesting to you, email [pythonclockorg@gmail.com](mailto:pythonclockorg@gmail.com)).

Python 2, thank you for your years of faithful service.

Python 3, your time is now.

**How do I get started?**

If the code you care about is still on Python 2, that's totally understandable. Most of PyPI's popular packages now [work on Python 2 and 3](#), and more are being added every day. Additionally, a number of critical Python projects have [pledged to stop supporting Python 2 soon](#). To ease the transition, the [official porting guide](#) has advice for running Python 2 code in Python 3.

If you're here, you'd probably enjoy the [full python release schedule](#) (a glitch maintained by [Dustin Ingram](#))





# O que significa a aposentadoria do Python 2?

- PEP 373 (novembro de 2008)
  - Última release planejada (2.7.18) será lançada em 2020
    - Isso quer dizer que não serão lançadas mais melhorias para Python 2, nem mesmo para vulnerabilidades de segurança ou bugs críticos
  - Já deveríamos estar tratando Python 2 como código legado a muito tempo

# Em termos práticos

- Seu código py 2 **não** vai deixar de funcionar
  - Novas features, da maioria das bibliotecas disponíveis para a linguagem, não serão compatíveis com o seu código
  - E se eu decidir não migrar
    - Preso ao **legado** e ao passado!

# Em uma área em que

- Depende de tecnologia de ponta
  - Evolui rapidamente
  - Um novo algoritmo revolucionário pode surgir a qualquer momento





# Migrar para Python 3!



# Mãos a obra



# Planejamento é essencial

- O segredo do sucesso é ser pragmático
  - Estudar o seu próprio código
  - Estudar as ferramentas disponíveis para fazer a migração

# Planejamento é essencial

- O segredo do sucesso é ser pragmático
  - Estudar o seu próprio código
  - Estudar as ferramentas disponíveis para fazer a migração
    - [Porting Python 2 code to Python 3](#)
    - [Migration strategies](#)

# Manter o suporte a Python 2

- Não

- Migrar o código para Python 3 é mais simples

- ~~Supportar Python 2 e 3~~

- Deixemos essa dificuldade para autores de bibliotecas

# Para quem gosta de receita de bolo

1. Se preocupe apenas com o Python 2.7
2. Tenha uma boa cobertura de testes
3. Saiba as diferenças entre Python 2 e 3
4. Use [caniusepython3](#) para ter certeza que as libs que você usa já são compatíveis com Python 3
5. Use o [2to3](#) para fazer a migração

# O estado atual do código importa





# O cenário do nosso código

- Cultura de entrega de resultados de negócio
  - Cultura fraca de engenharia
    - Sem testes
    - Vários débitos técnicos
    - Código espalhado em vários repositórios
- Código bem instrumentado para deploy
- Boa arquitetura

# Plano de migração

- Criação de um backlog
  - Tarefas divididas em: antes, durante e depois da migração
  - Novas features ficariam bloqueadas até o fim da migração
    - Comportamento de uma função não deveria mudar durante a migração

# Trade of

- Preferência por ferramentas para fazer a migração de forma automatizada
- Débito técnico da falta de testes de unidade **não** seria pago
  - Testes manuais
  - Shadow dos requests de produção

# Mãos a obra



# Suas dependencias suportam Python 3?

```
$ caniusepython3 --requirements requirements.txt
```

```
ederfmartins@dragon:~/work/recsys/recsys_api$ caniusepython3 --requirements requirements.txt
Finding and checking dependencies ...

You need 1 project to transition to Python 3.
Of that 1 project, 1 has no direct dependencies blocking its transition:

flasgger
```

```
$ pip install caniusepython3
```

# E se alguma dependência não for compatível?

- Verifique com os desenvolvedores da biblioteca o que está impedindo a transição
  - Muitas vezes será algo simples que você mesmo pode fazer
- Portar o código da biblioteca para dentro do seu código
  - Uma boa opção caso você utilize muito pouco dos recursos da biblioteca em questão
- Existe alguma outra biblioteca que faça a mesma coisa e tenha uma API semelhante a que você usa?

# Nossa experiência

- 2 bibliotecas não foram compatíveis
  - Uma apenas não tinha declarado isso como a ferramenta pedia
  - Outra não tinha suporte de fato e decidimos extrair a funcionalidade dela para dentro de nosso código
- Algumas tinham evoluído
  - Uma tinha evoluído e decidimos aceitar o novo comportamento [\[refactoring\]](#)

# Que partes do código eu deveria alterar?



```
$ pip install 2to3
```

```
$ 2to3 -w -n <my_python2_code_dir>
```

-w sobrescrever o arquivo de entrada

-n sem backup



# Nem tudo pode ser automatizado

- Divisão
- Diferenciação entre tratamento de strings binárias e de texto



# Gotcha: Divisão em Python 3

- Em Python 3 a divisão de dois **inteiros** resulta num **float**
- Engenharia de features muitas vezes requer divisão
  - Importante manter o comportamento do código inalterado

# Gotcha: Divisão em Python 3

- Antes de aplicar o 2to3 foi necessário avaliar cada uso do sinal de divisão
  - Alteramos para divisão inteira (//) onde fazia sentido

# Pitfall: Texto ou bytes?

```
arquivo = open('meu-gif.gif', 'rb')  
  
identificador = arquivo.read(6)  
print(identificador == 'GIF89a')
```

True ou False?

# Texto ou bytes?

- Se um trecho de código lida com os dois
  - Saiba com qual tipo você está lidando a cada momento
  - Converta texto para byte (ou vice-versa) nas bordas do seu código
    - Ex: Quando carregar o dado do banco de dados
  - Escolha corretamente o tipo do literal
  - Leia um arquivo corretamente (r ou rb)

# Bônus: Carregar arquivo serializado em Python 2 no Python 3

```
with open(filename, 'rb') as f:  
    object = pickle.load(f, encoding='latin1')
```

# E se eu não puder migrar agora?

```
$ python -3 app.py
```

Observe os warnings gerados pelo seu próprio código!

# Considerações finais

- Planejamento prévio
- Decisões baseadas em nosso contexto
  - Melhores práticas foram usadas como recomendações e não como regras
  - Aproveitamos pontos fortes de nosso código
- Conheça a si mesmo
  - Tamanho do problema
  - Riscos





[github.com/ederfmartins/pybr2019](https://github.com/ederfmartins/pybr2019)