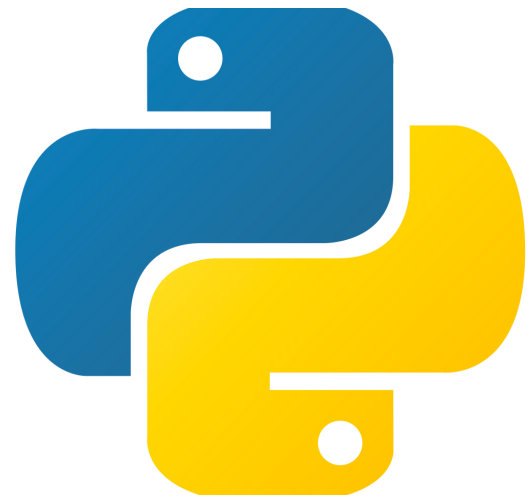
MAS NA MINHA MÁQUINA FUNCIONA!!!

A frase que tira o sono de todo time dev e do QA, mas pq isso acontece?

Aprenda de uma vez por todas a gerenciar pacotes com ambientes virtuais e acabe com o caos!



Lá vem história

O famoso dilema **"na minha máquina está funcionando"** atormenta desenvolvedores e todo o time de QA.

Para exemplificar vou explorar nos próximos slides a história de **João, o engenheiro de dados Jr.**

O começo

João, buscando formas de trabalhar com grandes volumes de dados encontrou uma biblioteca chamada **pandas**.

```
pip install pandas
```

Ele instalou, abriu seu **Jupyter Notebook**, e começou a explorar.

O encanto

A simplicidade de ler arquivos com apenas uma linha de código o deixou maravilhado. A capacidade de filtrar, agrupar e transformar dados com apenas algumas linhas parecia quase mágica.

```python
import pandas as pd

# Ler um arquivo CSV
df = pd.read_csv('data_vendas.csv')

# Filtrar e transformar dados
filtered_df = df[df['vendas'] > 10]
transformed_df =
filtered_df.groupby('categoria').sum()
```

A ilusão

João compartilhou sua **descoberta com sua equipe**, e eles também ficaram fascinados pelo poder de pandas.

Toda a equipe começou a adotar essa prática em diferentes projetos. **Vários projetos se iniciaram** e todos viveram felizes para sempre... kkk

Os primeiros problemas

João começa a notar problemas sutis no **Projeto 1** e no **Projeto 2**, mas consegue contorná-los localmente.

Confidente, ele envia o código para o time de Qualidade para testarem as novas funcionalidades.

O Caos se Instala

O time de QA começa a testar os projetos e **nada funciona!**

Os erros se acumulam, e a frustração cresce.

Eles jogam no ChatGPT para diagnosticar os problemas, mas as coisas parecem sem solução.

A Resposta Comum

Desesperados, o time de QA contata João, que testa o código em sua máquina e diz a frase infame:

"Na Minha Máquina Funciona"

O erro clássico é revelado!
Como resolver isso? Por que isso ocorre?
Vou ter que formatar o PC? (eu já fiz isso kkk)

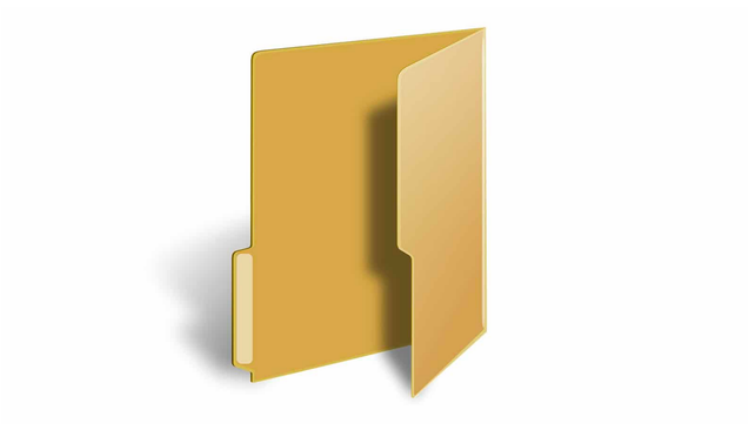
Tudo Funcionava...

...Até que nada funciona!

Mistério do 'pip install'

O comando 'pip install' é como sua pasta de downloads. Você acha que ta tudo bem. Mas no final fica uma zona se não cuidar.

```
pip install pandas
```



Pasta downloads

Pandas
Numpy
Scikit-Learn
Matplotlib
PySpark
....

Para onde tudo vai?

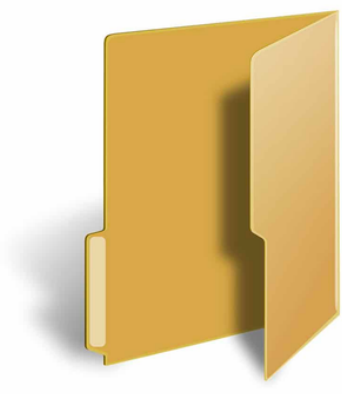
Por padrão vai para o ambiente global. É o espaço onde os pacotes são instalados e acessíveis por qualquer projeto Python em sua máquina.

Quando você executa **pip install pandas**, sem estar em um ambiente virtual, o pacote é instalado neste ambiente global.

Instalar pacotes no **ambiente global** é uma coisa que você precisa para imediatamente!

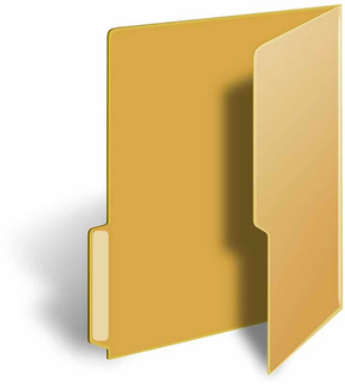
O ambiente virtual

Um ambiente virtual é um ambiente isolado e autocontido que permite que você instale pacotes e dependências **específicos para um projeto particular**, separados do sistema global Python.



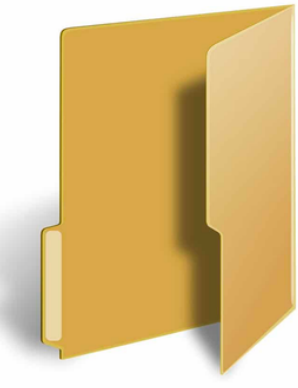
Projeto 1

Pandas
Numpy



Projeto 2

Scikit-Learn
Matplotlib



Projeto 3

PySpark

Benefício

Finalidade: Oferecer uma maneira de isolar dependências de projeto.

Como Funciona: Um ambiente virtual cria uma cópia das bibliotecas padrão em um diretório separado.

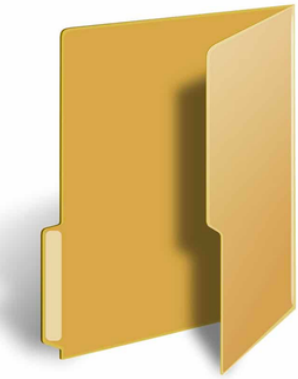
Reprodutibilidade: Facilita a recriação do ambiente de desenvolvimento em outras máquinas ou por outros desenvolvedores, garantindo que **todos estejam usando as mesmas versões dos pacotes!**

Criação

Comece com um ambiente virtual e evite surpresas desagradáveis.

Para isso é muito fácil.

```
python3 -m venv meu_ambiente
```



meu_ambiente

Esse comando cria uma pasta dentro do seu projeto para você baixar e instalar tudo o que precisa.

Ativação

Ativar o ambiente é o passo para a consistência.

Isso fala ao Python "olhe e salva só no meu ambiente local , fique longe daquela bagunça da pasta downloads"

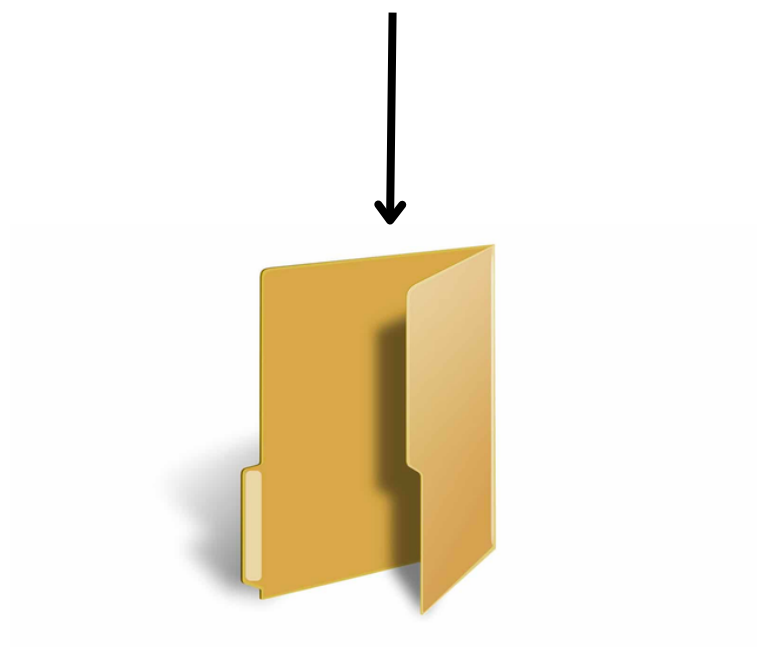
```
source meu_ambiente/bin/activate #linux e mac
meu_ambiente/Scripts/activate #windows
```

Depois de ativado

Você instala tudo normal

O PIP já vai direcionar o download para a pasta do seu projeto

```
pip install pandas
```



meu_ambiente

E o QA?

E como eu compartilho meu ambiente isolado com o time de QA?

Mando por e-mail essa pasta? Não!

```
pip freeze > requirements.txt
```

Você cria uma lista com suas lib

requirements.txt

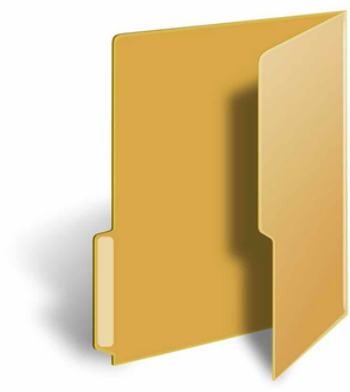
```
numpy==1.21.0
pandas==1.3.0
scikit-learn==0.24.2
matplotlib==3.4.2
seaborn==0.11.1
```

Para instalar

Você só precisa baixar o código via git e o seu time irá realizar o procedimento semelhante, mas usando o pip em cima do seu arquivo.

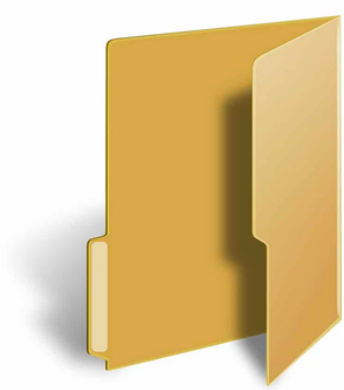
```
python -m venv ambiente_do_qa
source ambiente_do_qa/bin/activate
pip install -r requirements.txt
```

requirements.txt

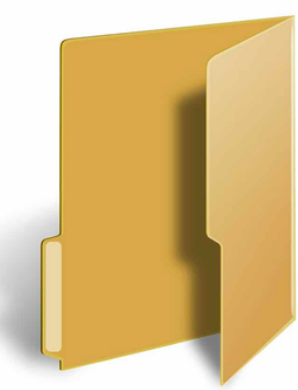


ambiente_do_qa

Ou seja



=



ambiente_do_dev **ambiente_do_qa**

Na Máquina dos dois agora Funciona

TL;DR

1. Abrir terminal e ir até o diretório
2. Criar ambiente virtual
3. Ativar
4. Instalar dependências
5. Gerar requirements.txt
6. Ser feliz

Conclusão

A gestão de ambientes virtuais é fundamental para manter a consistência e evitar conflitos entre projetos.

Utilizar um arquivo requirements.txt garante que todos na equipe estejam trabalhando com as mesmas versões de bibliotecas.

Previne o "Na Minha Máquina Funciona"

A adoção de práticas robustas de gerenciamento de ambiente evita erros comuns e frustrações no desenvolvimento e QA.

Valeu!

Se essa postagem te ajudou a compreender melhor as nuances e melhores práticas do Python, curta e comente abaixo!

Sua interação pode ajudar outras pessoas a se beneficiarem desse conhecimento.