Machine Learning Engineer

Python para ML e IA

Ferramentas Necessárias

Leonardo Pena

/ Seja muito bem vindo

FERRAMENTAS

Apresentaremos ferramentas para gerenciar ambientes e versões (pyenv, pipx, Poetry)

PROBLEMAS

Veremos soluções para problemas comuns na instalação de pacotes e manutenção do ambiente global

VANTAGEM

Diminuir conflitos e tornar o fluxo de trabalho mais organizado

Objetivo dessa primeira parte

Passo 1

Entender por que gerenciar várias versões de Python pode ser complicado

Passo 2

Conhecer pyenv e pipx como soluções que isolam ambientes

Passo 3

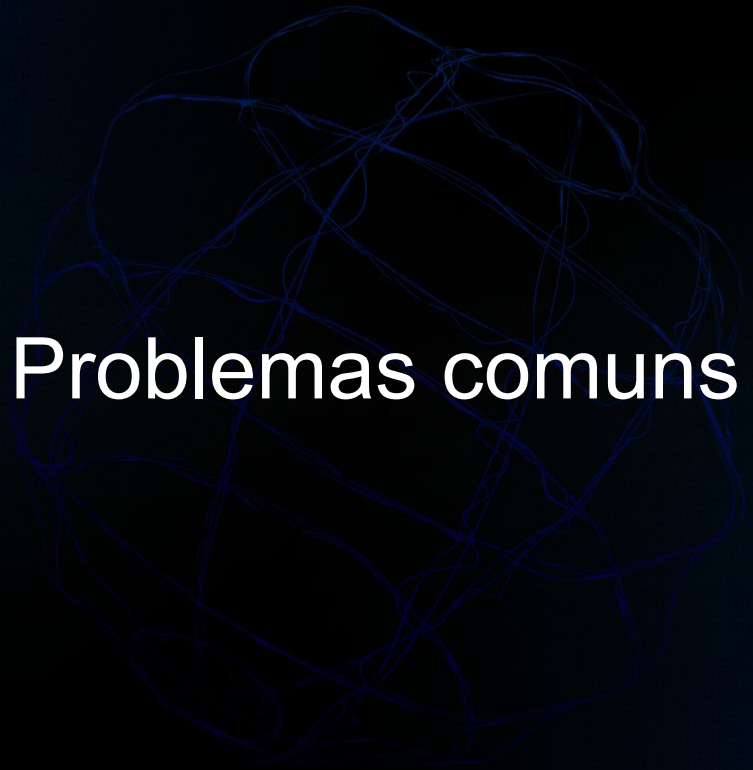
Ver como Poetry administra dependências e ambientes virtuais

Passo 4

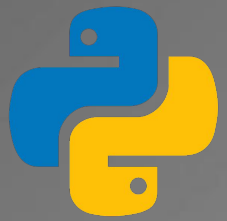
Aprender boas práticas de setup inicial para projetos Python

Passo 5

Preparar o terreno para validação e migrações (nos próximos vídeos)

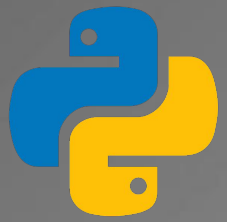


Problemas comuns



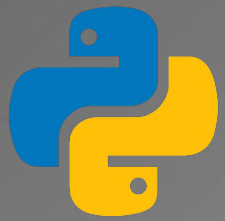
/ Problemas comuns

- Múltiplas versões do Python (3.7, 3.8, 3.9, etc.) instaladas na máquina
- Conflitos de dependências quando se instalam pacotes globalmente
- Dificuldade em reproduzir o ambiente de desenvolvimento em outra máquina
- Problemas com versionamento de pacotes: upgrade quebra projeto
- Ineficiência ao gerenciar ambientes manualmente



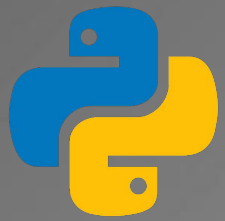
/ pyenv

- pyenv **gerencia** diferentes versões do Python lado a lado
- Permite instalar e alternar rapidamente entre versões (3.8, 3.11, etc.)
- “pyenv local X” para um projeto específico, “pyenv global Y” para todo o sistema
- Evita bagunça no PATH e conflito em sistemas operacionais
- Útil para testar compatibilidade de códigos em várias versões de Python



/ pyenv: O que é então?

O Pyenv é um gerenciador de ambientes Python. De maneira resumida, é uma ferramente que te permite escolher entre diversas versões do Python para usar.

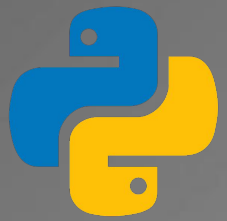


/ pyenv: Como instalar?

```
$ curl https://pyenv.run | bash
```

Com o comando acima, no terminal

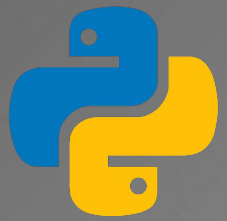
https://github.com/pyenv/pyenv-installer



/ pyenv: Demonstração

- pyenv install X.Y.Z baixa e compila a versão desejada
- pyenv local define versão para um repositório/projeto
- pyenv global define versão global do Python no sistema
- Testar com python --version para confirmar as mudanças
- Praticidade para alternar entre ambientes sem mexer em configurações do sistema

```
pyenv install 3.11.2
pyenv local 3.11.2
pyenv global 3.8.12
```

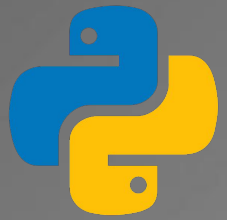


/ pipx

- pipx instala ferramentas de linha de comando Python em ambientes virtuais
- Cada ferramenta fica “encapsulada”, sem poluir o ambiente global
- Ideal para ferramentas como black, flake8, httpie, etc.
- Atualizações independentes, evitando conflitos de versão
- pipx também facilita upgrade ou remoção dessas ferramentas

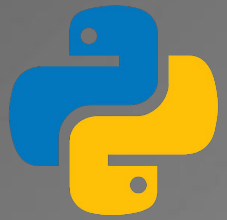
https://github.com/pypa/pipx

```
pipx install black
pipx upgrade black
pipx uninstall black
```



/ pipx: Demonstração

- pipx install black cria ambiente virtual e instala o black
- pipx upgrade black atualiza a ferramenta sem afetar outros pacotes
- pipx uninstall black remove o ambiente virtual do black
- Funciona para qualquer CLI Python, não só formatadores
- Protege contra incompatibilidades de bibliotecas internas



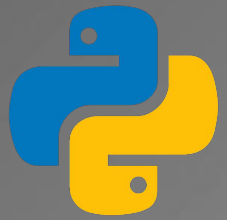
/ poetry

- Poetry integra gerenciamento de pacotes + ambiente virtual
- Usa arquivo pyproject.toml para listar dependências
- Cria automaticamente um venv para cada projeto
- Simplifica a distribuição do projeto com poetry build
- Ajuda a manter versões de pacotes travadas (via poetry.lock)

https://python-poetry.org/

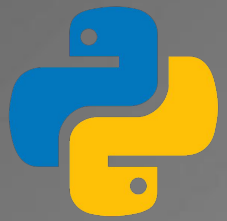
/ poetry: Demonstração

```
poetry new meu_projeto
cd meu_projeto
poetry add requests
poetry install
poetry shell
```



- poetry new gera estrutura básica (inclui pyproject.toml)
- poetry add requests adiciona requests como dependência
- poetry install baixa e instala dependências no venv do projeto
- poetry shell ativa o ambiente virtual gerenciado pelo Poetry
- poetry.lock registra versões exatas instaladas

https://python-poetry.org/

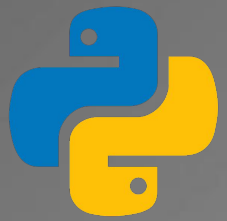


/ Comparação

- **pyenv**: Foca em múltiplas versões de Python
- **pipx**: Instala ferramentas de CLI em ambientes isolados
- **Poetry**: Gerencia dependências e ambientes em projetos específicos
- Podem ser usados juntos: pyenv para versão de Python, pipx para CLIs, Poetry para projetos
- **Ganho de organização e escalabilidade em qualquer tamanho de equipe**

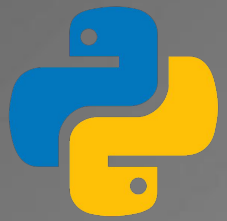
/ Comparação

Ferramenta	Função Principal	Exemplo de Uso
pyenv	Gerenciar versões de Python	Alterar entre Python 3.7 e 3.11
pipx	Instalar ferramentas em ambientes isolados	`pipx install black` e `pipx install flake8`
Poetry	Gerenciar dependências e ambientes	`poetry new`, `poetry add requests`, `poetry install`



/ Boas práticas com Poetry

- Versionar o arquivo *pyproject.toml* no repositório
- Não instalar pacotes globalmente quando estiver usando Poetry
- Lembrar de *poetry lock* para congelar as versões de dependências
- Usar *poetry export -f requirements.txt* se precisar de compatibilidade com pip
- Mantenha o repositório limpo, sem arquivos de dependências obsoletos



/ Vantagens de um setup limpo

- Facilita colaboração: qualquer pessoa pode clonar o projeto e dar poetry install
- Evita conflitos de versão de Python (pyenv cuida disso)
- Ferramentas de CLI ficam isoladas via pipx, sem “poluir” o ambiente
- Código reproduzível em produção (versões idênticas ao ambiente local)
- Menos tempo gasto debugging instalações e paths

FIAP + alura