### **Ambientes virtuais nas Ciências de Dados**

#### **O que são ambientes virtuais? De onde vieram? Para quê servem? Como usá-los?**



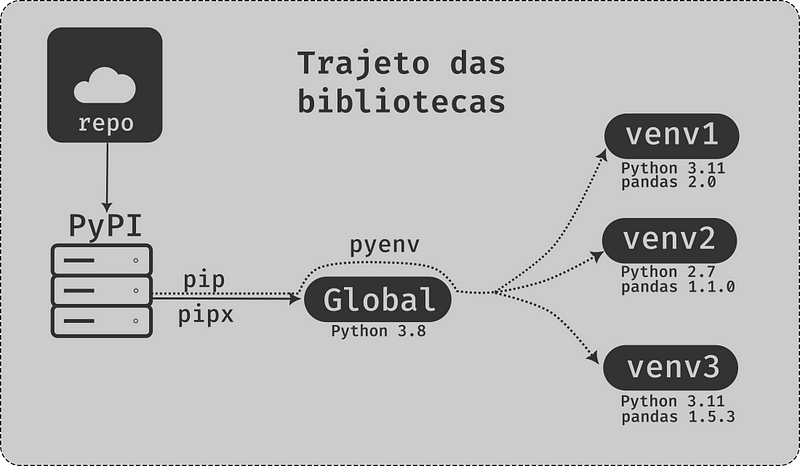
*Disclaimer: Este é a primeira postagem deste blogue. A ideia dele é que eu, no decorrer da minha trajetória como aprendiz em Ciência de Dados vindo de um arcabouço educacional, poss acompartilhar os conhecimentos e estudos que venho adquirindo e realizando, de forma que outros estudantes e iniciantes possam se beneficiar deste esforço à mesma medida que eu tenha a oportunidade de praticar e consolidar ainda mais estes estudos.*

Ambientes virtuais são ferramentas extremamente valiosas para desenvolvedores e cientistas de dados. Entretanto muitas vezes são ignoradas ou mal compreendidos pelos iniciantes na área.

Para deixar claro a importância dos ambientes virtuais é preciso que entendamos o motivo pelo qual foram criados e o trajeto que a instalação de diferentes bibliotecas, módulos e pacotes realizam até chegar nos nossos ambientes de trabalho.

A maior parte das bibliotecas, pacotes e módulos que utilzamos estão alocados em repositórios remotos (na maioria das vezes disponíveis no GitHub, GitLab e etc). Esses repositórios são centralizados e distribuídos pelo **PyPI** (*Python Package Index*), o repositório padrão da linguagem Python., da mesma forma que o **CRAN** (*Comprehensive R Archive Network*) está para a linguagem R.

Fluxo geral de aquisição de pacotes, módulos e bibliotecas.



No Python nós podemos acessar as bibliotecas do PyPI através, pirncipalmente, do gerenciador de pacotes **pip** (*package installer for Python*), que faz o serviço de buscar o pacote desejado no repositório remoto e instalá-lo no diretório local de destino, disponibilizando-o para que usemos no Python. Entretanto diferentes sistemas operacionais já apresentam versões instaladas nativamente, tanto do Python quanto das bibliotecas, o que pode acabar gerando problemas de conflito entre versões, caso o interpretador Python esteja instalado em uma versão e dependa de pacotes de versões específicas que passem a ser modificados pelo usuário.

Alternativamente o gerenciador de pacotes **pipx** faz a instalação de novas bibliotecas e suas dependências de forma isolada do restante do sistema, mas é capaz de disponibilizar estes pacotes globalmente, garantindo a facilidade de acessar as bibliotecas de qualquer diretório ao mesmo tempo que não “sujamos” o diretório de pacotes do Python global. Contudo o problema do conflito entre versões das bibliotecas, suas dependências e até mesmo do interpretador Python persiste. É ai que entram os ambientes virtuais!

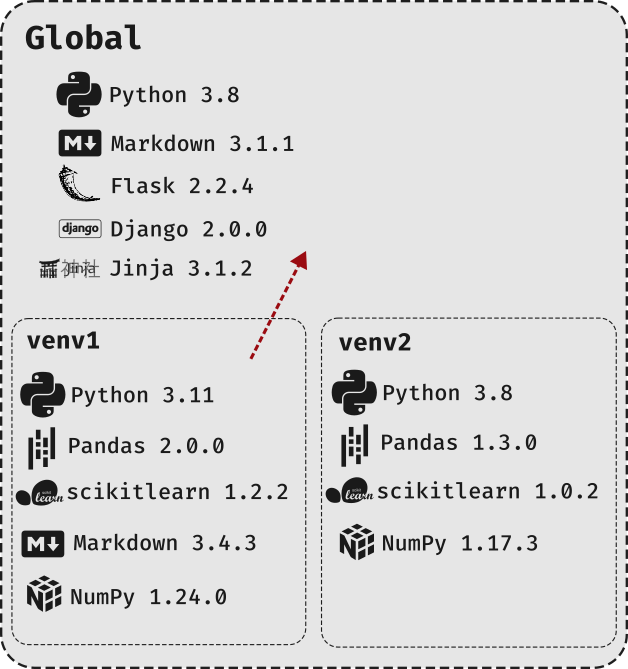
As instâncias de ambientes Python na máquina.



Os ambientes virtuais são diretórios separados e dedicados a receber versões específicas do Python, bem como de bibliotecas e dependências. Eles isolam o ambiente global do sistema do ambiente de desenvolvimento em que estamos trabalhando, nos dando a liberdade de instalar diferentes bibliotecas e versões sem que uma interfira na outra e tampouco com o restante do sistema, evitando clonflitos e permitindo maior liberdade de trabalho.

Os interpretadores, bibliotecas e suas dependências ficam isolados dentro do diretório do ambiente virtual de modo que não temos acesso a usar seu conteúdo a não ser que ativemos o ambiente virtual específico. Entretanto, uma vez que ativamos um ambiente virtual continuamos com acesso às bibliotecas que estão fora dele, no ambiente global, ainda que as versões instaladas dentro do ambiente virtual sejam prioritárias em relação ao ambiente global. É como se o Python procurasse primeiro nas pastas dentro do ambiente virtual e, caso não encontre dentro dele, ai sim procurará a biblioteca ou funçõ requisitada fora dele.

As solicitações feitas são buscadas primeiro no ambiente virtual e, caso a biblioteca ou função solicitada não seja encontrada dentro do ambiente virtual, ai sim o ambiente global será acessado para satisfazer a solicitação.



Exitem diversas ferramentas usadas para gerenciar ambientes virtuais em Python, que foram sendo desenvolvidas conforme o próprio Python foi ganhando novas versões. Entre eles podemos apontar alguns, como: **venv**, **virtualenv**, **pyenv**, **virtualenvwrapper**, **Autoenv**, **Buildout**, **conda**, **poetry**, etc. Cada um com suas vantagens e desvantagens, facilidades e espeficidades. Na próxima postagem abordarei como configurar e administrar um ambiente virtual usando o **pyenv**.

#### **Referências**

* Eduardo Mendes — Ambientes virtuais e intalação de bibliotecas-Live de Python #191: <https://www.youtube.com/live/naGF7EIUFp0?feature=share>
* Kenneth Reitz & Tanya Schlusser: The Hitchhiker’s Guide to Python
* teclado — The complete guido to Python virtual environments: <https://youtu.be/KxvKCSwlUv8>