### **#3 Ambientes virtuais: Como se estrutura um ambiente virtual?**

#### **Como estão dispostos os diretórios e como funciona, na prática, um ambiente virtual?**

[Imagem por Masaru Suzuki](https://unsplash.com/pt-br/fotografias/u5h0pSeyCYU)



Agora que nos familiarizamos com o tema dos ambientes virtuais a vimos como criar um a partir do zero, podemos nos aprofundar um pouco mais neste tópico, explorando a estrutura de diretórios e o funcionamento interno de um ambiente virtual. Um ponto importante para ser ressaltado antes de iniciar a leitura é que estou, mais uma vez, usando uma distribuição do Linux e que, portanto, a estrutura de diretórios terá suas particularidades. Mas isto não impede a compreensão da lógica por trás da organização do *pyenv*. Além disso esta postagem terá alguns *print screens* novamente, para melhor ilustrar os diretórios e caminhos. Caso você ainda não tenha lido os textos anteriores da série sobre ambientes virtuais, aqui estão os links para o [primeiro](https://medium.com/thevirtualenvironmentalist/ambientes-virtuais-nas-ci%C3%AAncias-de-dados-dfdd9ac05a19) e para o [segundo](https://medium.com/thevirtualenvironmentalist/2-ambientes-virtuais-como-iniciar-um-ambiente-virtual-para-ci%C3%AAncia-de-dados-f53977cfddb8?source=your_stories_page) textos.

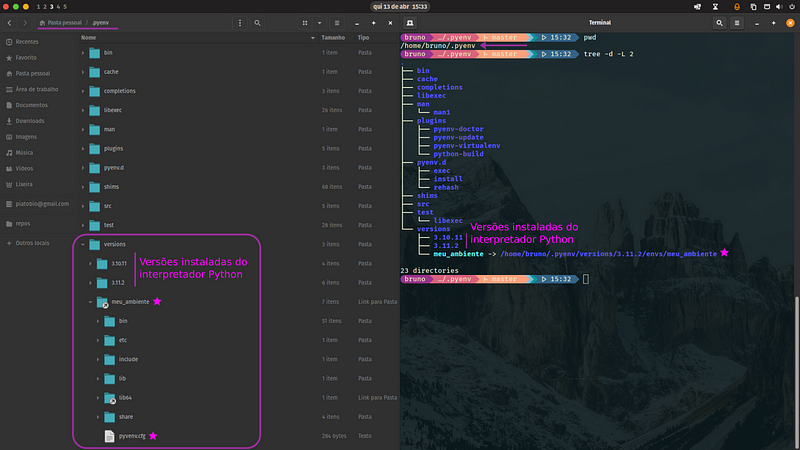
Um ambiente virtual é, em sua essência, uma pasta com uma estrutura específica de diretórios e arquivos em seu interior. Diferentes gerenciadores de ambientes virtuais em diferentes sistemas operacionais apresentam algumas especificidades e diferenças. Aqui eu focarei apenas no *pyenv* que, apesar de não ser um gerenciador de ambientes virtuais em sua essência, conta atualmente com o *plugin* nativo *virtualenv* que o torna um potente gerenciador de versões do interpretador Python ao mesmo tempo que consegue gerenciar muito bem ambientes virtuais. O pacote com seu plugin é comumente chamado de *pyenv-virtualenv*, mas por comodidade de escrita, aqui usarei apenas *pyenv* para me referir a eles.

Antes de mais nada devemos ter em mente que o *pyenv* é, originalmente, um gerenciador de versões de interpretadores Python e que, portanto, se destina a instalar e manusear estas diferentes versões. Por isso, ele apresenta uma estrutura básica relativamente simples dentro do qual está inserido um diretório específico para as versões do Python instaladas. O próprio *pyenv* é instalado em uma pasta oculta dentro de */home/usuario*, que no meu caso, o usuário se chama *bruno*.

cd /home/bruno/.pyenv

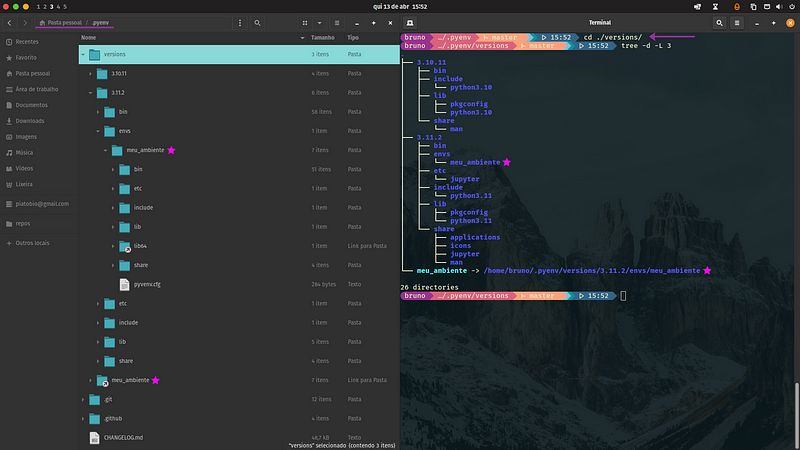
As diferentes versões do Python instaladas usando o *pyenv* são adicionadas à pasta ~/*.pyenv/versions*. Elas apresentam estrutura interna idêntica à do Python quando instalado diretamente no sistema, contendo um diretório para os binários do interpretador (*/bin*), um diretório contendo as bibliotecas necessárias para que o interpretador funcione (*/lib*). Na imagem abaixo podemos ver o interior do diretório *~/.pyenv/versions*. Nele há três itens: um diretório chamado .*/3.10.11*, um segundo chamado *./3.11.2* e, por fim, um link simbólico (*symlink*) chamado *./meu\_ambiente*. O link simbólico *./meu\_ambiente* e um arquivo dentro dele chamado *pyvenv.cfg* estão marcados na imagem anterior com estrelas pois serão importates pra gente no decorrer deste texto.

Fig 1. Estrutura básica de diret7órios do *pyenv*. Os diretórios **./3.10.11** e **./3.11.2** são os locais em que as versões instaladas do interpretador Python estão. O symlink **./meu\_ambiente** se conecta ao interior do diretório da versão do interpretador usada para criá-lo (neste caso, a 3.11.2) e está marcada com uma estrela. Além deste há um arquivo marcado com uma estrela chamado **pyvenv.cfg** sobre o qual falaremos mais a frente. Imagem do autor.



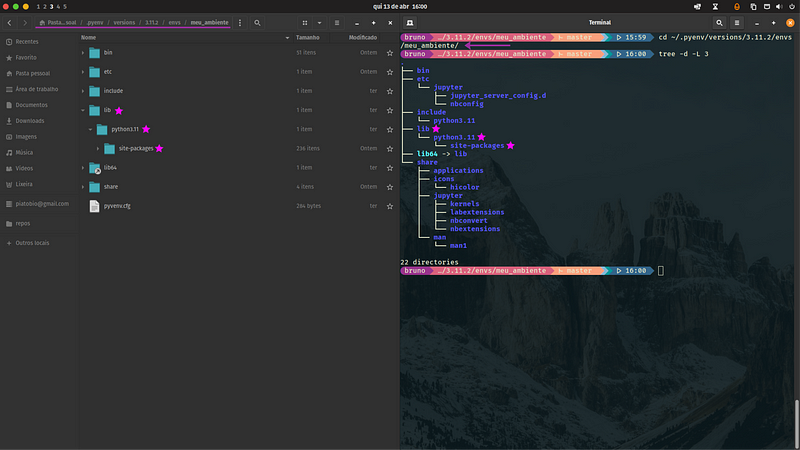
Os diretórios *./3.10.11* e *./3.11.2* são os locais onde estão instaladas as versões do Python que instalamos durante o tutorial do texto anterior. O link simbólico se conecta ao interior do diretório da versão usada para criá-lo. Se verificarmos o interior do diretório dedicado à versão *3.11.2*, veremos uma pasta chamada *./envs*, onde estão os ambientes virtuais criados com esta versão e, dentro dela, a pasta do nosso ambiente meu\_ambiente. Desta forma temos duas maneiras de acessar o diretório do ambiente virtual. Via *symlink* e via diretório da versão.

Fig 2. O symlink se conecta ao interior da pasta da versão do interpretador usado para criá-la. Imagem do autor.



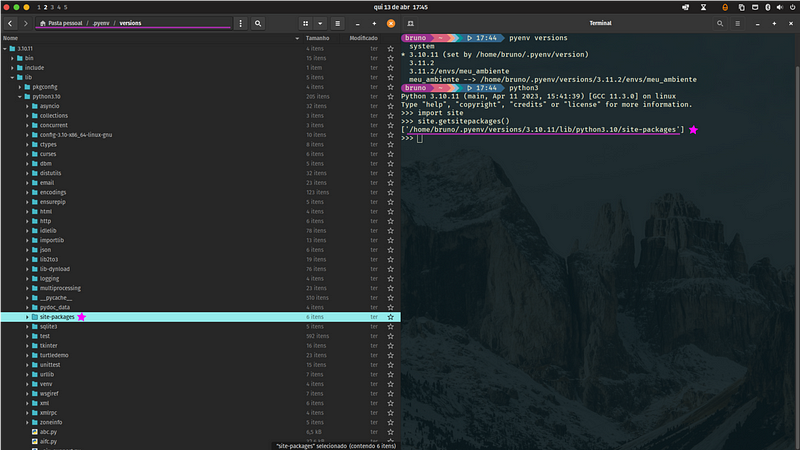
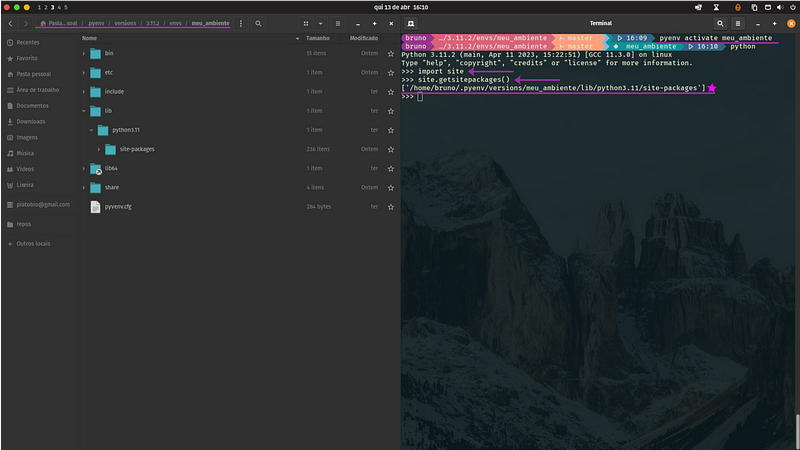
Entramos agora no diretório do ambiente virtual *meu\_ambiente*, que criamos anteriormente. Nele podemos ver uma pasta de binários onde estão alocados todos arquivos compilados que usaremos para rodar o interpretador, bem como das demais bibliotecas que instalamos, como o Jupyter Notebook. Além disso temos uma pasta importante chama *./lib* em que estão alocadas as bibliotecas que instalamos (*pandas*, *numpy*, *seaborn*, *matplotlib*, etc). Como podemos usar mais de uma versão para criar um ambiente virtual no *pyenv*, temos a pasta Python3.11 para indicar a versão na qual instalamos as bibliotecas e, por fim, dentro desta, a pasta *site-packages*. É dentro desta pasta que estão alocados os pacotes que estalamos.

Fig 3. Dentro da pasta do ambiente virtual encontramos a pasta **/lib**, onde estão sitiadas as bibliotecas que instalamos de cada versão do interpretador usada no ambiente. Imagem do autor.



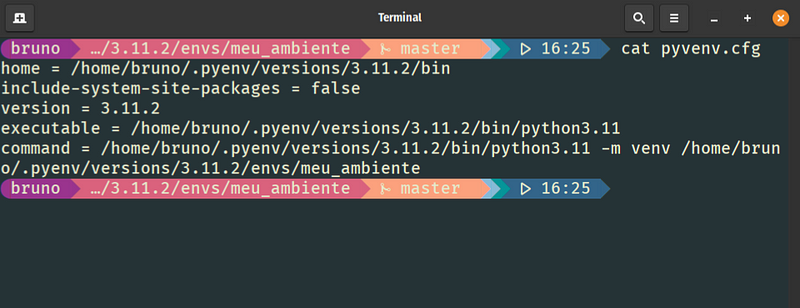
Podemos verificar o local em que estão sendo instaladas nossas bibliotecas do Python usando a função *getsitepackages()* do pacote *site*. Lembrando que, para tanto, precisamos ativar o ambiente virtual *meu\_ambiente*.

Fig 4. Usando o comando **site.getsitepackages()** para confirmar o local em que estão sendo instaladas nossas bibliotecas. Imagem do autor.Fig 5. Quando verificamos o local em que os pacotes do ambiente global estão sendo instalados, encontramos o caminho para um diretório diferente daquele do ambiente virtual. Imagem do autor.



Voltemos agora para o arquivo *pyvenv.cfg*, situado na pasta do ambiente virtual. Se abrirmos este arquivo, notaremos que ele contém os seguintes atributos:

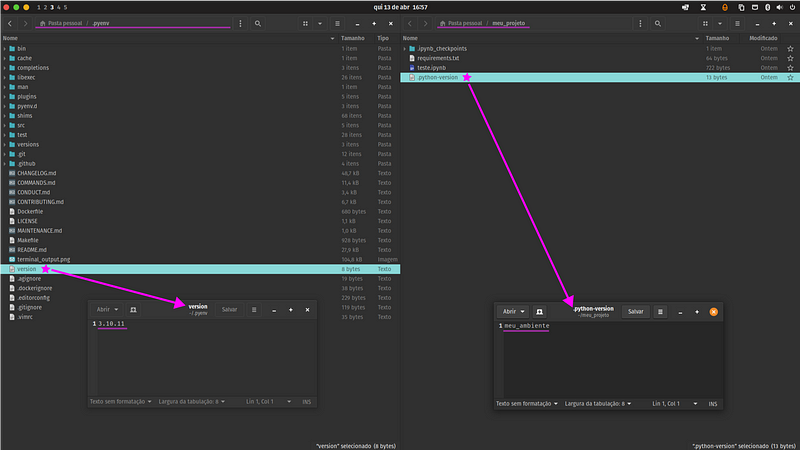
**home**: o diretório em que estão situados os binários referentes ao ambiente virtual  
**include-system-site-packages**: um atributo booleano que indica se o ambiente virtual deve ultrapassar suas barreiras e usar os pacotes do interpretador do sistema  
**version**: a versão do interpretador que estamos usando nestes ambiente  
**command:** instruções de como se comportar para acessar este ambiente virtualFig 6. Arquivo **pyvenv.cfg**, um arquivo contendo algumas chaves e valores para ajustar o comportamento do pyenv. Imagem do autor.



Estes atributos são instruções para o funcionamento adequado do *pyenv*. Como dito anteriormente, o *pyenv* trabalha com um gradiente de prioridade de definições das versões e ambientes a serem usados em cada situação. Esta prioridade é dada por três arquivos principais:

* **~/.pyenv/versions/3.11.2/envs/meu\_ambiente/pyvenv.cfg**: configura o ambiente virtual e é usado quando o ativamos, seja manual ou automaticamente.
* **~/.pyenv/version**: define a versão global do Python a ser utilizada. Aquela que definimos com o comando *pyenv global 3.10.11* no texto anterior.
* **~/meu\_projeto/.python\_version**: define a versão ou ambiente local a ser usado dentro de um diretório específico. No texto anterior definimos como versão local para o projeto *meu\_projeto* o ambiente virtual *meu\_ambiente*.

Fig 7. Os arquivos **version** no diretório **./.pyenv/** e **.python-version** dentro do diretório do projeto definem, respectivamente a versão



Por fim podemos, ainda, definir uma versão específica para o Python para todas as vezes que estivermos usando o shell. Ela será sobreposta sobre todas as outras configurações, sendo assim prioritária sobre as demais. Para isso usamos o comando *pyenv shell <versão\_desejada*>. Quando definimos uma versão específica para o shell, esta informação é adicionado aos comando do shell.

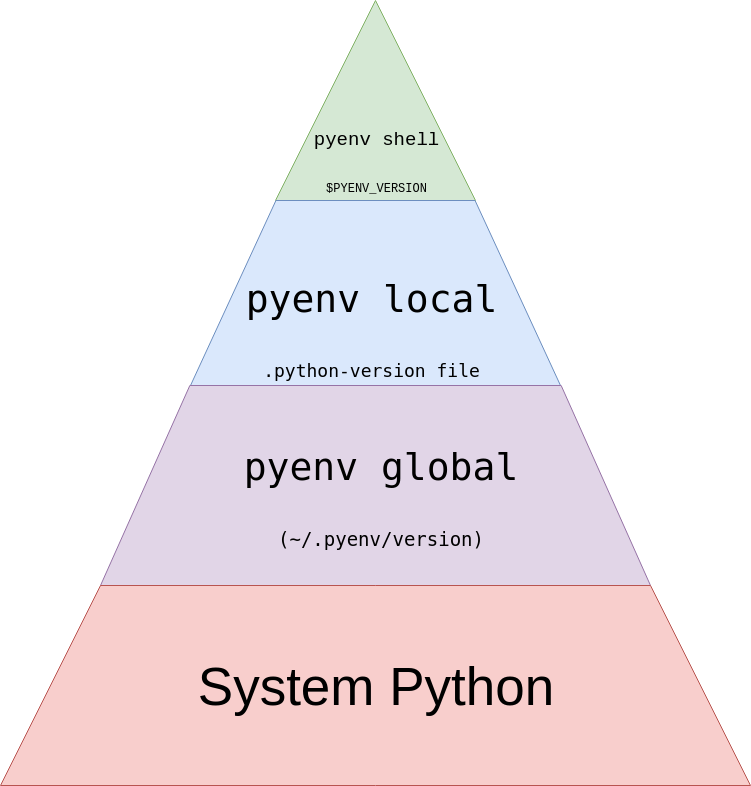
Desta maneira, note que há uma ordem de prioridade entre as formas de definição das versões através do *pyenv*. Esta ordem é a seguinte:

Shell > local > global > sistema.Fig 8. A ordem de prioridade de ambientes e versões do pyenv segue do local em vermelho para fora, ou seja, o ambiente virtual ativada tem maior prioridade, seguido pelo ambiente local, o ambiente global e, por fim o ambiente do sistema. Imagem do autor.



A ideia é similar à da pirâmide desenvolvida por Logan Jones no blogue RealPython.

Fig 9. Pirâmide que representa a ordem de resolução prioritária do pyenv. Ela deve ser lida de cima baixo, de modo que o pyenv procura primeiro uma especificação de versão dada no shell. Caso não encontre, ele busca uma especificação de versão local, seguida pela especificação da versão global e,por fim, usa a versão do próprio sistema. Esta ordem de especificações de versões está alocada nos arquivos e na estrutura de diretórios do pyenv, como vimos. [Imagem original de RealPython](https://realpython.com/intro-to-pyenv/#installing-pyenv)



#### **Conclusões**

* O pyenv tem uma estrutura de diretórios própria que ele usa para estabelcer e decidir qual versão e ambiente usar em cada situação
* Ele faz isso através de um sistema de prioridade de caminhos, escolhendo sempre o ambiente ativado no *shell*, o ambiente ativado localmente, o ativado globalmente e o do sistema, respectivamente.
* As bibliotecas instaladas em ambientes virtuais ficam alocadas separadamente das bibliotecas do ambiente global, que ficam separadas das do sistema, garantindo a organização e ausência de conflitos.

#### **Referências**

[Real Python — Python Virtual Environments: A Primer](https://realpython.com/python-virtual-environments-a-primer/#use-third-party-tools) (<https://realpython.com/python-virtual-environments-a-primer/#use-third-party-tools>)

Real Python — Managing Multiple Python Versions With pyenv (<https://realpython.com/intro-to-pyenv/>)

pyenv — pyenv Documentation (<https://github.com/pyenv/pyenv>)

TowardsDataScience — Managing virtual environments with pyenv (<https://towardsdatascience.com/managing-virtual-environment-with-pyenv-ae6f3fb835f8>)