

Máquinas virtuais - Venv(Python)

Para que serve :

Uma máquina virtual é uma ferramenta usada para isolar as dependências de um projeto instaladas sem interferir no sistema operacional ou em outros projetos em desenvolvimento na mesma máquina.

Ou seja, tenho na minha máquina uma versão do Pandas x, porém o meu projeto tem em suas dependências a versão y, logo é necessário a instalação de uma venv (máquina virtual no python) para que eu isole aquela dependência do meu projeto da tecnologia já instalada em minha máquina.

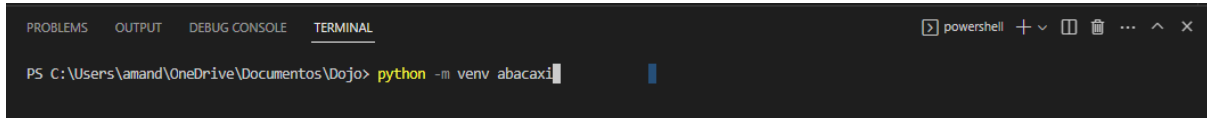
Imagine o cenário onde várias pessoas estão mexendo no mesmo projeto e cada uma tem uma versão das dependências do projeto em sua máquina, certamente uma hora ou outra o projeto não iria funcionar por conta das funcionalidades disponíveis em cada versão das tecnologias, as máquinas virtuais servem para inibir esse cenário.

para isso usamos um arquivo chamado “requirements.txt”, ele serve para armazenar todas as dependências do projeto, quando uma pessoa quiser rodar o projeto apenas irá instalar o pacote requirements.txt em sua máquina virtual que todas as dependências serão baixadas na sua máquina virtual sem que vá para o seu computador.

Criando e rodando Ven / requirements.txt

criando uma venv -

para criar uma venv utilizamos o seguinte código(Windows) :

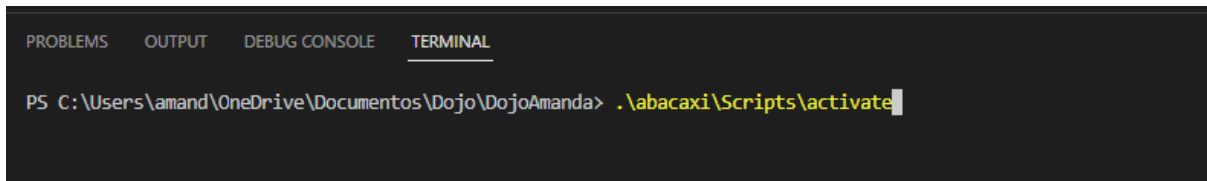


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\amand\OneDrive\Documentos\Dojo> python -m venv abacaxi
```

onde chamamos o python e iniciamos uma Virtual env (venv) e definimos o nome dela como abacaxi.

Porém, depois da criação devemos ativar venv em nosso terminal para baixarmos nossas dependências.

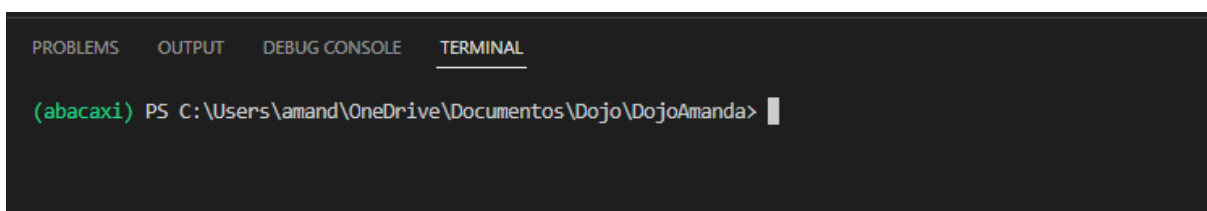
Terminal sem venv ativada:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> .\abacaxi\Scripts\activate
```

Terminal com Venv ativada:

para ativarmos nossa venv rodamos o seguinte código(Windows):




```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda>
```

onde chamamos nossa venv que demos o nome de abacaxi e navegamos até o diretório activate dela, assim fica nosso terminal com nossa venv ativada:

```
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> |
```

okay, agora podemos baixar nossas dependências. Para isso iremos criar um código que cria uma matriz aleatória usando a biblioteca Numpy
baixando a biblioteca em nossa venv:

```
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> pip install numpy
Collecting numpy
  Downloading numpy-1.24.3-cp310-cp310-win_amd64.whl (14.8 MB)
     14.8/14.8 MB 9.5 MB/s eta 0:00:00
Installing collected packages: numpy
|
```

código usado :

```
import numpy as np

# Cria uma matriz 2D aleatória com 3 linhas e 4 colunas
matriz_aleatoria = np.random.rand(3, 4)

print(matriz_aleatoria)
```

ok, agora rodando o código “pip freeze” veremos todas as dependências baixadas em nossa venv:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> pip freeze
numpy==1.24.3
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> 
```

Criando nosso arquivo requirements.txt

```
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> pip freeze > requirements.txt
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> 
```

rodando o código `pip freeze > requirements.txt` ele criará um arquivo chamado `requirements.txt` com todas as dependências relacionadas ao nosso código

```
≡ requirements.txt
```

```
ensinandoPandas.py  ≡ requirements.txt  X
DojoAmanda > ≡ requirements.txt
1  numpy==1.24.3
2  
```

ok, agora iremos adicionar outra biblioteca para entendermos o conceito da VM

```
ensinandoPandas.py requirements.txt X
DojoAmanda > requirements.txt
1 numpy==1.24.3
2 pandas==2.0.1
```

adicionamos o pandas com versão de release 2.0.1 nas dependências do nosso projeto, e agora para que essa dependência não interfira meu computador iremos baixar na nossa venv(abacaxi) com o código `pip install -r requirements.txt`

```
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> pip install -r requirements.txt
Requirement already satisfied: numpy==1.24.3 in c:\users\amand\onedrive\documentos\dojo\dojoamanda\abacaxi\lib\site-packages (from -r requirements.txt (line 1)) (1.24.3)
Collecting pandas==2.0.1
  Downloading pandas-2.0.1-cp310-cp310-win_amd64.whl (10.7 MB)
    10.7/10.7 MB 5.6 MB/s eta 0:00:00
```

rodando um `pip freeze` agora podemos ver que a nova dependência pandas foi instalada:

```
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> pip freeze
numpy==1.24.3
pandas==2.0.1
python-dateutil==2.8.2
pytz==2023.3
six==1.16.0
tzdata==2023.3
```

rodando nosso projeto em nossa venv:

```
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> python ensinandoPandas.py
[[0.67565556 0.56664117 0.66596255 0.0376361 ]
 [0.72041829 0.3340592 0.07015726 0.15356886]
 [0.9580127 0.83234838 0.40464439 0.38393641]]
(abacaxi) PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda>
```

rodando nosso projeto em nossa máquina local onde não temos as dependências do projeto baixadas:

```
PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> python ensinandoPandas.py
Traceback (most recent call last):
  File "C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda\ensinandoPandas.py", line 1, in <module>
    import numpy as np
ModuleNotFoundError: No module named 'numpy'
PS C:\Users\amand\OneDrive\Documentos\Dojo\DojoAmanda> █
```