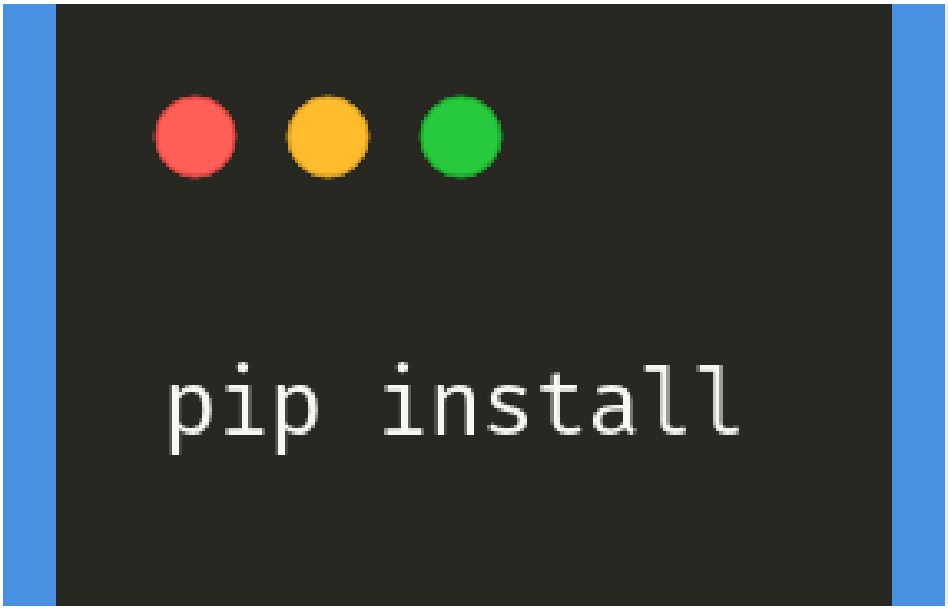
Ambientes virtuais em Python



Nádia Oliveira
10/09/2020

COMPARTILHE

Se você é um(a) desenvolvedor(a) Python ou um(a) entusiasta desta linguagem, provavelmente já se deparou com uma pasta denominada **venv** nos projetos.

Afinal, o que é essa pasta? Qual sua importância para o desenvolvimento de um projeto em Python? É isso que vamos descobrir neste artigo.

O **Python** é muito conhecido por suas facilidades, seja para codificar ou até mesmo na instalação de bibliotecas, pois apenas com o comando:

```
pip install
```

O ambiente é configurado conforme sua necessidade, porém isso pode se tornar um problema.

Imagine que você está trabalhando em um projeto que utiliza a versão 0.12 do **Flask**, porém passado algum tempo surge uma nova versão e uma nova proposta de projeto a ser feita. Você, todo animado(a) com isto, e já sabendo que o Flask está instalado em seu computador, simplesmente o atualiza para a versão mais recente. E agora? Será que a aplicação que foi feita com o Flask na versão 0.12 continuará funcionando?

Atualizei e agora?

Infelizmente a aplicação irá quebrar. Isso porque de uma versão para outra muita das vezes algumas funcionalidades são descontinuadas ou então mudam algum comportamento. Agora você terá que refatorar todo o código escrito na versão 0.12 para que ele rode na versão mais recente instalada em seu computador. É um pouco frustrante, pois isso levará um tempo.

E é este o problema que o ambiente virtual resolve, o módulo **venv** irá **isolar as dependências do projeto** de modo onde **cada projeto** terá suas bibliotecas de **maneira separada** do sistema principal. Utilizando esta abordagem, você conseguirá facilmente montar um ambiente que rode a versão 0.12 do Flask e outro que rode a mais recente sem que haja conflitos, pois agora, cada projeto possui sua própria “caixa de ferramentas”. Além disso, conseguirá replicar o mesmo ambiente em outros computadores com enorme facilidade.

Criando um ambiente virtual

Se você utiliza a IDE **PyCharm**, automaticamente quando criar um projeto será criado uma pasta **venv**. Mas, e se quisermos fazer o procedimento manualmente?

Vamos primeiro criar uma pasta para simular esse novo projeto, você pode fazer manualmente ou pelo terminal:

```
mkdir novo_projeto
```

Agora, vamos entrar na pasta do projeto:

```
cd novo_projeto/
```

Para criar o ambiente virtual, abra o terminal dentro da pasta criada e faça:

```
python3 -m venv nome_do_ambiente_virtual
```

O `nome_do_ambiente_virtual` por convenção utiliza-se `venv`, mas poderiamos utilizar outro nome sem problema algum. A partir do comando acima será criada a pasta que conterá a versão do Python instalada em sua máquina e todas as bibliotecas que serão instaladas nesse projeto, ou seja, as bibliotecas do nosso novo_projeto ficarão apenas nesse diretório e não diretamente no sistema principal como era feito antes.

Mas, não basta somente criar o ambiente, é necessário ativá-lo:

```
source nome_do_ambiente_virtual/bin/activate
```

Caso utilize o windows, para ativar o ambiente o comando se difere:

```
nome_do_ambiente_virtual\Scripts\Activate
```

Agora podemos **instalar as dependências sem que haja futuros conflitos**, basta digitar pip install e o nome da biblioteca que deseja instalar.

Replicando ambientes

No início do artigo falamos sobre os ambientes virtuais trazerem a simplicidade de replicar o ambiente em outras máquinas e isso se deve a possibilidade de exportarmos um arquivo com todas bibliotecas que o nosso projeto contém. Veja como é simples:

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

Com o comando acima, será criado um arquivo com todas as bibliotecas presentes em nosso ambiente virtual. Por exemplo:

```
flake8==3.7.9
Flask==1.1.2
Flask-API==2.0
```

Agora, se quisermos rodar o nosso projeto em outra máquina, não será necessário baixar as dependências uma a uma, basta fazer:

```
pip install -r requirements.txt
```

Com o comando acima, será instalado de forma automática todas as bibliotecas presentes no arquivo requirements.txt

E por fim, para **desativar o ambiente virtual**:

```
deactivate
```

Simples e funcional, não é? É sempre recomendado isolar as dependências do seu projeto, criando um ambiente virtual para cada um deles. Isso é apenas o básico dos ambientes virtuais, para saber mais recomendo ler a documentação e caso queria conhecer mais sobre **Python**, aqui

na **Alura** temos a **Formação Python** onde você vai aprender a linguagem na prática, aplicando orientação a objetos em seu código e boas práticas de programação.

Espero que tenha gostado do artigo e nos vemos no próximo, até lá!

Confira neste artigo:

Nádia Oliveira

Nádia é uma entusiasta do Python, graduada em Sistemas de Informação pela Universidade Federal de Viçosa e faz parte do time do Apoio Educacional aqui na Alura. No fórum, você a encontrará nas categorias de Python, Python Web e Data Science.