Tutorial:

Virtualização de Ambientes em Python e Gerenciamento de Bibliotecas

Neste tutorial, você aprenderá sobre a importância da virtualização de ambientes em Python e como criar, ativar, desativar e gerenciar ambientes virtuais em diferentes sistemas operacionais, como Linux, Mac e Windows. Além disso, aprenderá como instalar uma biblioteca via pip (Disponibilizada no Python Index Package - https://pypi.org/) e como listar todas as bibliotecas instaladas no ambiente usando o comando pip freeze.

Use sem moderação!

Pré-requisito:

Ter o Python 3 instalado no seu computador. Linux e Mac o Python já vem instalado por padrão. No Windows é necessário instalar manualmente. Para qualquer Sistemas Operacional, você encontrará o Download em https://www.python.org/downloads/.

Por que usar virtualização de ambientes em Python?

A virtualização de ambientes em Python permite isolar projetos e suas dependências em ambientes separados. Isso é útil quando você está trabalhando em diferentes projetos que podem exigir versões diferentes das mesmas bibliotecas. Com ambientes virtuais, você pode ter configurações específicas de bibliotecas para cada projeto, evitando conflitos e problemas de compatibilidade.

Criando e ativando um ambiente virtual (Linux, Mac e Windows):

Siga as instruções abaixo para criar e ativar um ambiente virtual em diferentes sistemas operacionais.

1. Linux e Mac:

Abra o terminal e execute os seguintes comandos:

```
python3 -m venv nome_ambiente  # Cria o ambiente virtual
source nome_ambiente/bin/activate  # Ativa o ambiente virtual
```

2. Windows:

Abra o prompt de comando e execute os seguintes comandos:

```
C:\> python -m venv nome_ambiente  # Cria o ambiente virtual
C:\> nome_ambiente\Scripts\activate  # Ativa o ambiente virtual
```

Observações:

Por convenção, chamamos os ambientes virutais de venv (uma abreviatura para virtual environment, em português Ambiente Virtual). Então é comum dentro do diretório do seu projeto, criar uma virtualenv com o comando python3 -m venv venv, onde o primeiro venv é o comando e o segundo o nome do ambiente.

Quando um ambiente está ativado, verás o nome dele entre parênteses na esquerda do seu prompt de comando.

Enquanto o ambiente estiver ativado naquele prompt (pode ser o terminal do seu sistema operacional ou no próprio terminal da sua IDE) o comando pip com os seus derivados (install/freeze/uninstall) estarão relacionados aquele ambiente virtual ativado.

Sugiro criar o ambiente virtual Python (a venv), dentro do diretório (pasta) do seu projeto.

Desativando o ambiente virtual:

Quando você terminar de trabalhar em um ambiente virtual, pode desativá-lo usando o seguinte comando:

```
(nome_ambiente) deactivate
```

Instalando uma biblioteca via pip:

Depois de ativar o ambiente virtual, você pode instalar bibliotecas usando o pip. Execute o seguinte comando para instalar uma biblioteca chamada "exemplo":

```
(nome_ambiente) $ pip install exemplo
```

Listando todas as bibliotecas instaladas no ambiente:

Para verificar todas as bibliotecas instaladas em um ambiente virtual, use o comando pip freeze. Execute o seguinte comando:

```
(nome_ambiente) $ pip freeze
```

Isso exibirá uma lista de todas as bibliotecas instaladas junto com suas versões.

Conclusão:

Neste tutorial, você aprendeu sobre a importância da virtualização de ambientes em Python e como criar, ativar, desativar e gerenciar ambientes virtuais em diferentes sistemas operacionais. Além disso, você viu como instalar uma biblioteca via pip e como listar todas as bibliotecas instaladas usando o comando pip freeze. Agora você está pronto para começar a trabalhar com ambientes virtuais e gerenciar suas dependências de forma eficiente em projetos Python.

Robertson Ebling dos Santos - Junho/2023