Ambiente Virtual e pip

Ambiente Virtual e pip

▼ O que é e para que serve

> ? Um ambiente virtual é uma ferramenta que cria um espaço isolado para instalar pacotes e dependências de um projeto, sem interferir no sistema ou em outros projetos. Isso evita conflitos entre versões de bibliotecas e facilita a organização do desenvolvimento

- Sem um ambiente virtual, todas as bibliotecas são instaladas no sistema global do Python, o que pode gerar conflitos quando diferentes projetos precisam de versões distintas de pacotes
- Exemplos de uso do ambiente virtual:
  - Desenvolvimento de aplicações web: Um projeto Flask pode precisar da versão 2.x do `Flask`, enquanto outro usa a versão 1.x. Com ambientes virtuais, cada projeto mantém sua própria versão sem interferências
  - Projetos científicos e de machine learning: Um projeto pode precisar do `NumPy` 1.21, enquanto outro exige o `NumPy` 1.19 para compatibilidade com um modelo antigo
  - Ambientes de testes: Criar um ambiente virtual permite testar pacotes sem afetar a instalação principal do Python

▼ Criar, ativar e desativar um ambiente virtual (**venv**, **activate** e **deactivate**)

- Para criar um ambiente virtual, pode-se usar os seguintes comandos no terminal/cmd dentro da pasta do projeto:

```
python -m venv venv
```

  - Esse comando segue a lógica `python -m venv nome_do_ambiente`

> ! É comumente utilizado o nome `venv` para criar o ambiente virtual. Outros nomes comuns podem ser `env` ou `.venv`

- Ativando um ambiente virtual:

```
venv\Scripts\activate
```

  - Dentro da pasta do ambiente virtual(`venv`) e dentro da pasta Scripts, utiliza-se o comando `activate` para ativar o ambiente virtual
  - Após ativar, o nome do ambiente aparecerá no início do terminal, indicando que ele está em uso

- Desativando um ambiente virtual:

```
deactivate
```

  - Já estando com o ambiente virtual ativado, utilizar o comando `deactivate` permite desativá-lo
  - Isso retornará o terminal ao ambiente Python global

▼ O que é pip (Package Installer for Python)

> ?
>
> O `pip` (Package Installer for Python) é o gerenciador de pacotes do Python. Ele permite instalar, atualizar e remover bibliotecas e módulos, facilitando o desenvolvimento de projetos

- No contexto do ambiente virtual, os pacotes instalados com `pip` ficam isolados do Python global. Isso evita conflitos entre projetos e garante que cada um tenha as dependências corretas.

▼ Instalar e remover pacotes (**pip install** e **pip uninstall**)

- Para instalar pacotes com `pip install` :

```
pip install nome_do_pacote
```

  - Exemplo: `pip install numpy` instalará a biblioteca `NumPy`, utilizada para computação numérica e manipulação de arrays em Python

- Para remover pacotes com `pip uninstall` :

```
pip uninstall nome_do_pacote
```

  - Exemplo: `pip uninstall numpy`

- Para instalar versões específicas de pacotes:

```
pip install nome_do_pacote==versão
```

  - Exemplo: `pip install numpy ==1.21.0`

- Para verificar pacotes instalados:

```
pip list
```

▼ Listar as bibliotecas instaladas no ambiente (**pip freeze**)

- O `pip freeze` é um comando utilizado para listar todas as bibliotecas instaladas no ambiente atual, junto com suas versões. O comando gera uma saída no formato padrão que pode ser registrada em um arquivo, como o `requirements.txt`, facilitando a replicação do ambiente em outras máquinas
- Esse comando lista as bibliotecas instaladas em seu respectivo ambiente, apresentando diferentes resultados quando utilizado no ambiente Python global e quando é utilizado em um venv, por exemplo
- Exemplo de uso:

```
pip freeze
```

  - Saída:

```
Flask==2.1.1
requests==2.26.0
numpy==1.21.2
```

▼ Listar requisitos de um projeto (**requirements.txt**)

> ❔ O arquivo `requirements.txt` é utilizado para listar as dependências de um projeto Python, ou seja, os pacotes de bibliotecas que o projeto precisa para funcionar corretamente. Ele ajuda a garantir que, ao compartilhar o código com outros desenvolvedores ou ao implantar o projeto, todos usem as mesmas versões das bibliotecas, facilitando a instalação e a configuração do ambiente.

- O nome `requirements.txt` é uma padronização para esse arquivo que lista as dependências de um projeto
- Para criar um arquivo `requirements.txt`, utiliza-se o comando `pip freeze`, que lista todas as bibliotecas instaladas no seu ambiente virtual, juntamente com suas versões. Exemplo:

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

    - Isso criará o arquivo com todas as dependências e suas versões, como por exemplo:

      ```
      Flask==2.1.1
      requests==2.26.0
      numpy==1.21.2
      ```

- Para instalar as dependências listadas, como em casos de instalar um projeto de outro desenvolvedor ou instalar as dependências para um outro projeto, deve-se executar:

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

    - Isso instalará as bibliotecas especificadas no arquivo, nas versões exatas mencionadas.