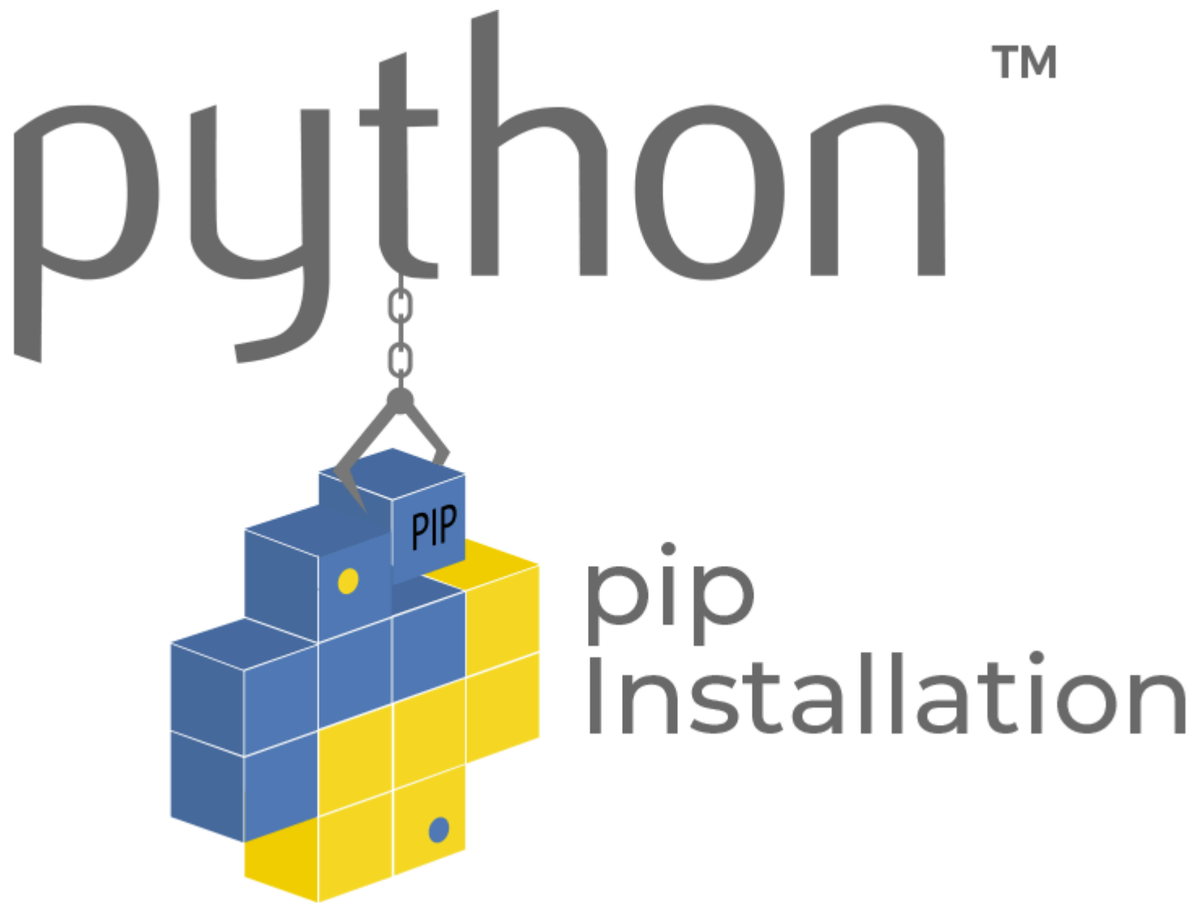
Ciência de Dados com Python/IA - SENAI-DF - Vespertino - 08/08/2024 - Marcelo Pereira Avelino

`PIP - Pip Installs Packages | Requirements.txt`

`1. PIP - PIP INSTALLS PACKAGES`

- O pip é um acrônimo para "Pip Installs Packages". Ele é o gerenciador de pacotes oficial para o Python. Com o pip, você pode instalar, atualizar e remover pacotes Python que são distribuídos via o Python Package Index (PyPI) e outras fontes.



- O pip é uma ferramenta essencial para qualquer desenvolvedor Python, pois facilita o processo de configuração e gerenciamento de bibliotecas e dependências em projetos Python.

`1.1. Principais funções do pip:`

- Instalação de pacotes: Você pode usar pip para instalar pacotes Python diretamente do PyPI, um repositório centralizado de bibliotecas Python.
- Gerenciamento de dependências: Quando você instala um pacote com pip, ele também instala automaticamente todas as dependências necessárias para que o pacote funcione corretamente.
- Atualização de pacotes: Você pode atualizar os pacotes instalados para suas versões mais recentes usando pip.
- Desinstalação de pacotes: pip também permite que você remova pacotes instalados que não são mais necessários.

`1.2. Exemplos de uso:`

- Instalar um pacote:

  ```
  pip install nome_do_pacote
  ```

- Atualizar um pacote:

  ```
  pip install --upgrade nome_do_pacote
  ```

- Desinstalar um pacote:

  ```
  pip uninstall nome_do_pacote
  ```

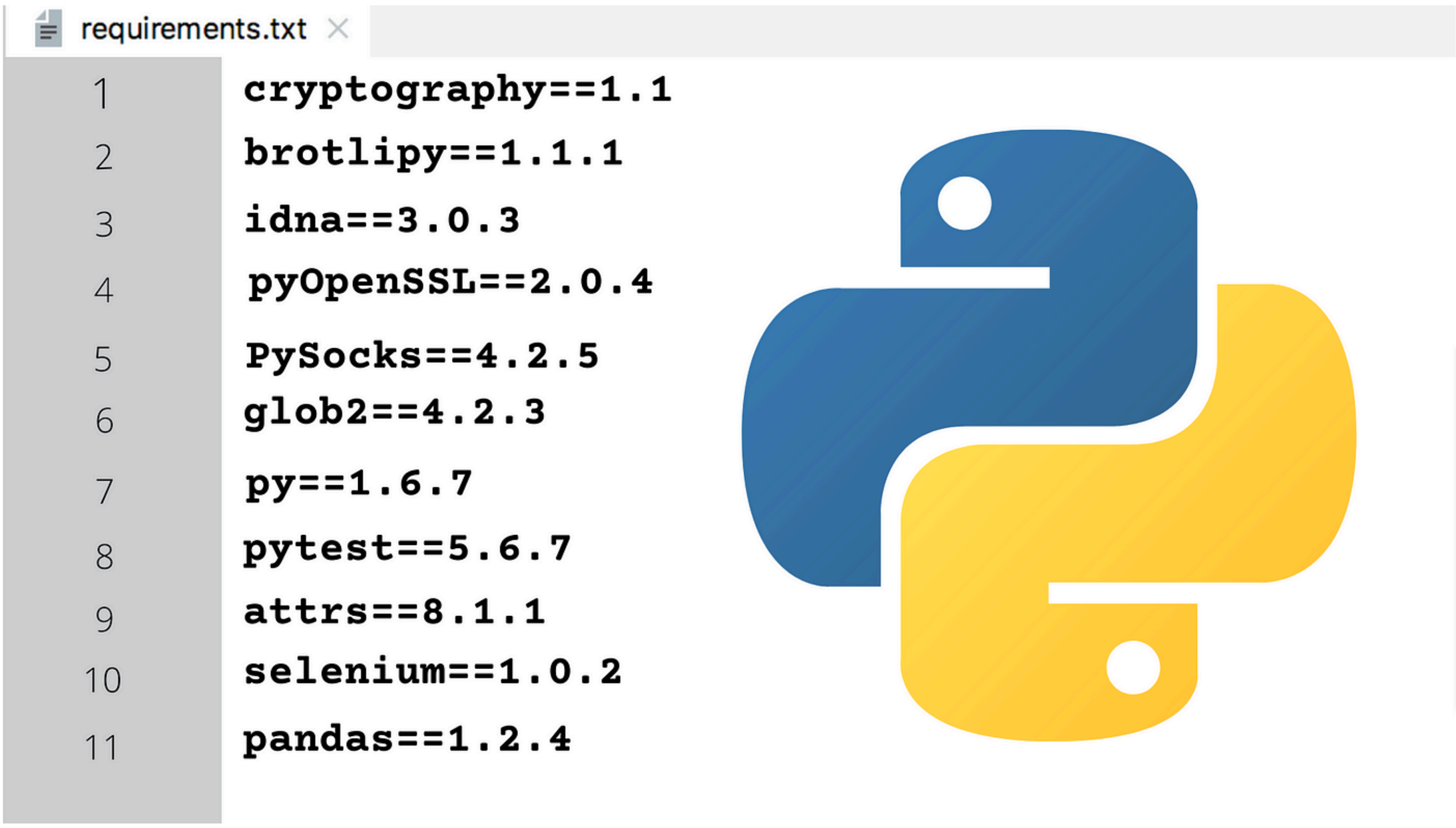
- Listar pacotes instalados:

  ```
  pip list
  ```

- Atualizar o pip para a versão mais recente

  ```
  python -m pip install --upgrade pip
  ```

`2. REQUIREMENTS.TXT:`



- O arquivo *requirements.txt* é um arquivo de texto simples que lista as dependências (bibliotecas e pacotes) que um projeto Python precisa para funcionar corretamente.
- Ele desempenha um papel crucial na gestão de dependências e na criação de ambientes de desenvolvimento e produção reprodutíveis.

- Detalhes do conteudo do arquivo requirements.txt:
  - O arquivo requirements.txt contém uma lista de pacotes Python, cada um em uma linha, seguido opcionalmente pela versão específica necessária. Cada linha segue um formato padrão.
  - Exemplo: `nome-do-pacote==versão`
    - numpy==1.21.0
    - pandas==1.3.0
    - matplotlib==3.4.2
  - Você também pode listar apenas o nome do pacote, sem especificar a versão:
    - flask
    - requests

`2.1. Propósito e Benefícios:`

- Reprodutibilidade: O principal benefício do requirements.txt é garantir que o ambiente de desenvolvimento seja reprodutível. Isso significa que qualquer pessoa que use o projeto pode instalar exatamente as mesmas versões dos pacotes necessários.
- Facilidade de Instalação: Com o requirements.txt, um novo desenvolvedor ou um ambiente de produção pode instalar todas as dependências necessárias simplesmente executando:

  ```
  pip install -r requirements.txt
  ```
- Controle de Versão: Especificar as versões dos pacotes evita que mudanças em versões futuras quebrem o código existente, pois garante que as versões usadas são aquelas testadas e confirmadas como funcionando corretamente.

`2.2. Criando um requirements.txt:`

Se você já tem um ambiente virtual (venv) configurado com todos os pacotes instalados, ara criar o *requirements.txt* com as dependências atuais do seu projeto, você geralmente executa o comando a seguir.

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

- A criação de um arquivo requirements.txt dentro de uma virtual environment (venv) é opcional, mas altamente recomendada.
- Ele serve para listar todas as dependências do seu projeto, facilitando a replicação do ambiente em outras máquinas e garantindo a instalação correta dos pacotes.
- Embora seja comum gerar o requirements.txt dentro da venv, é recomendável mantê-lo no diretório raiz do projeto, fora da venv, para facilitar o compartilhamento e controle de versão.

`2.3. Uso Típico`

- Instalação de Pacotes: Quando alguém clona o repositório do seu projeto, eles podem configurar o ambiente de desenvolvimento com todas as dependências necessárias usando:

  ```
  pip install -r requirements.txt
  ```
- Atualização de Dependências: Se novos pacotes forem adicionados ou atualizados, você pode atualizar o requirements.txt e compartilhar essa atualização com a equipe

`2.4. Boas Práticas`

- Manter o Arquivo Atualizado: Sempre que você adicionar, remover ou atualizar um pacote, lembre-se de atualizar o requirements.txt.
- Uso de Versões Específicas: É uma boa prática especificar as versões dos pacotes para evitar problemas de compatibilidade.

`2.5. Exemplo Real`

- Suponha que você está desenvolvendo uma aplicação web usando Flask e precisa de algumas bibliotecas adicionais para manipulação de dados e autenticação. Seu requirements.txt pode se parecer com isso:
  - Flask==2.0.1
  - SQLAlchemy==1.4.22
  - requests==2.26.0
  - numpy==1.2
- Qualquer desenvolvedor que queira trabalhar nesse projeto pode simplesmente clonar o repositório e rodar:

  ```
  pip install -r requirements.txt
  ```
- Isso garantirá que todas as dependências necessárias sejam instaladas na versão correta.