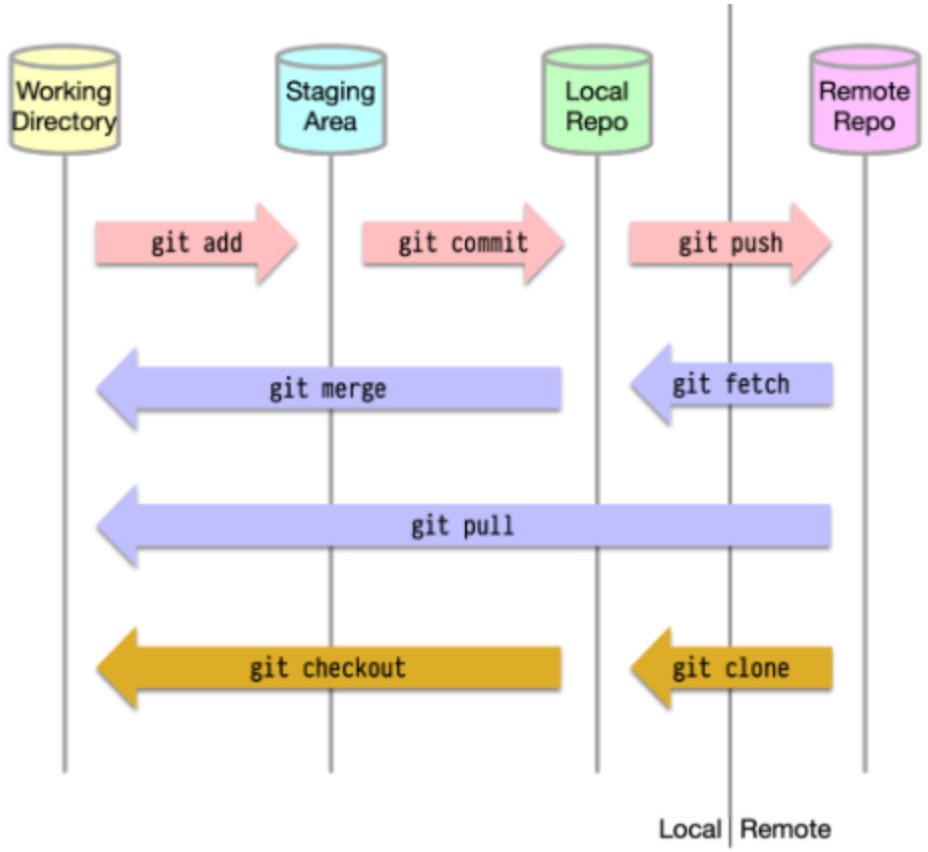
# Programação em Python (Back-end)


Instrutor: Pablo Araujo

(21) 97172-1697

# Aula 3

# Revisão - GIT


Working Directory
Staging Area
Local Repo
Remote Repo
git add
git commit
git push
git merge
git fetch
git pull
git checkout
git clone
Local
Remote

# Revisão - Classes

```python
class Carro:
    # Atributo de classe
    # Acesso - Não precisa da criação de uma instância
    rodas = 4

    # Construtor
    def __init__(self, marca, ano):
        # Atributo de instância
        # Acesso - Precisa da criação de uma instância
        self.marca = marca
        self.ano = ano

    # Método de instância (usa self)
    # Acesso - Precisa da criação de uma instância
    def ligar(self):
        print(f"O carro {self.marca} ({self.ano}) está ligado.")

    # Método de classe (usa cls)
    # Acesso - Não precisa da criação de uma instância
    @classmethod
    def mudar_numero_rodas(cls, novas_rodas):
        cls.rodas = novas_rodas
        print(f"Agora todos os carros têm {cls.rodas} rodas.")

    # Método estático (não usa self nem cls)
    # Acesso - Não precisa da criação de uma instância
    # Escopo - Não recebe o self e nem cls então não tem acesso.
    # Caso queira acessar os da Classe poderá, porém terá que passar explicitamente
    @staticmethod
    def calcular_idade(ano_fabricacao, ano_atual):
        return ano_atual - ano_fabricacao
```

Revisão (Instância de uma Classe):

- Criação.
- O construtor é responsável por receber a própria instância e os valores a serem introduzidos nos atributos da instância.
- Os métodos sem decoradores serão por padrão métodos de instância.

Entender (Métodos e atributos da Classe):

- Os atributos e métodos da classe podem ser acessados sem depender de uma instância.
- Os métodos estáticos não recebem nem valores de classe e nem de instância.

# Atividade - Classes

```python
class Carro:
    # Atributo de classe
    # Acesso - Não precisa da criação de uma instância
    rodas = 4

    # Construtor
    def __init__(self, marca, ano):
        # Atributo de instância
        # Acesso - Precisa da criação de uma instância
        self.marca = marca
        self.ano = ano

    # Método de instância (usa self)
    # Acesso - Precisa da criação de uma instância
    def ligar(self):
        print(f"O carro {self.marca} ({self.ano}) está ligado.")

    # Método de classe (usa cls)
    # Acesso - Não precisa da criação de uma instância
    @classmethod
    def mudar_numero_rodas(cls, novas_rodas):
        cls.rodas = novas_rodas
        print(f"Agora todos os carros têm {cls.rodas} rodas.")

    # Método estático (não usa self nem cls)
    # Acesso - Não precisa da criação de uma instância
    # Escopo - Não recebe o self e nem cls então não tem acesso.
    # Caso queira acessar os da Classe poderá, porém terá que passar explicitamente
    @staticmethod
    def calcular_idade(ano_fabricacao, ano_atual):
        return ano_atual - ano_fabricacao
```

- Crie um repositório
- Clone para sua máquina.
- Crie a classe da imagem.
- Suba para seu repositório.

# PIP – O que é ?

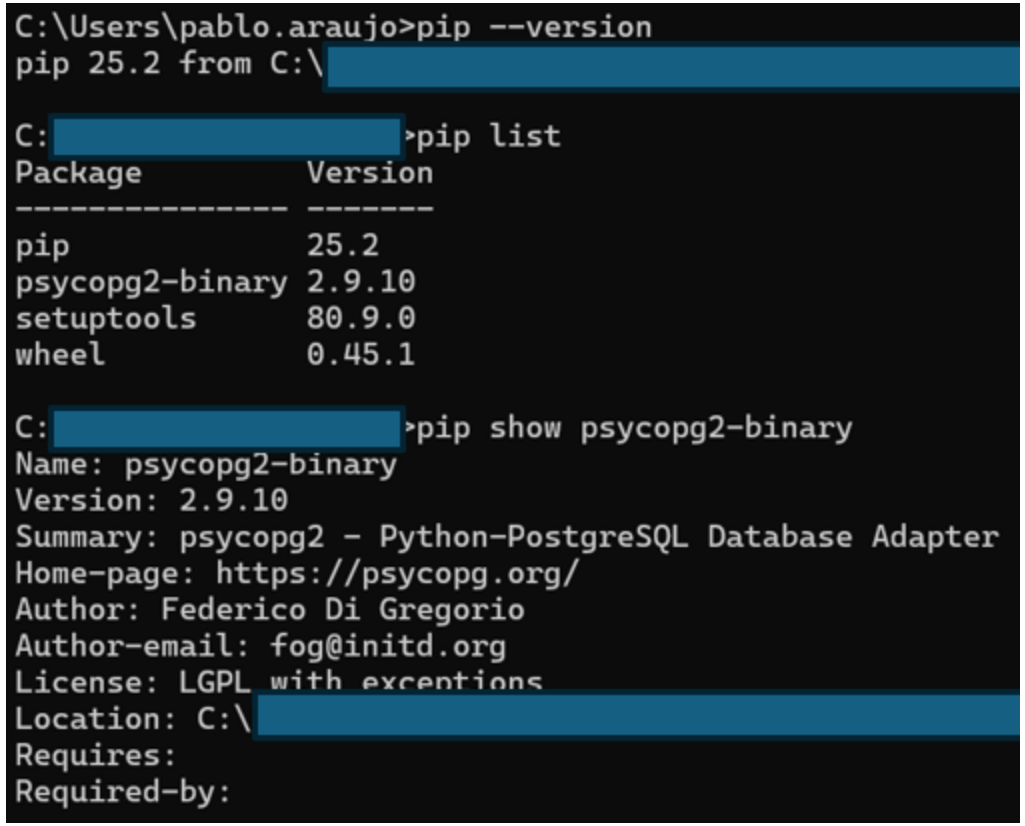
• pip é o gerenciador de pacotes oficial do Python.
• Permite instalar, atualizar e remover bibliotecas externas.
• Usado para trazer ferramentas que não fazem parte da instalação padrão do Python.

# PIP – Comandos Simples

Verificar versão pip:
pip --version

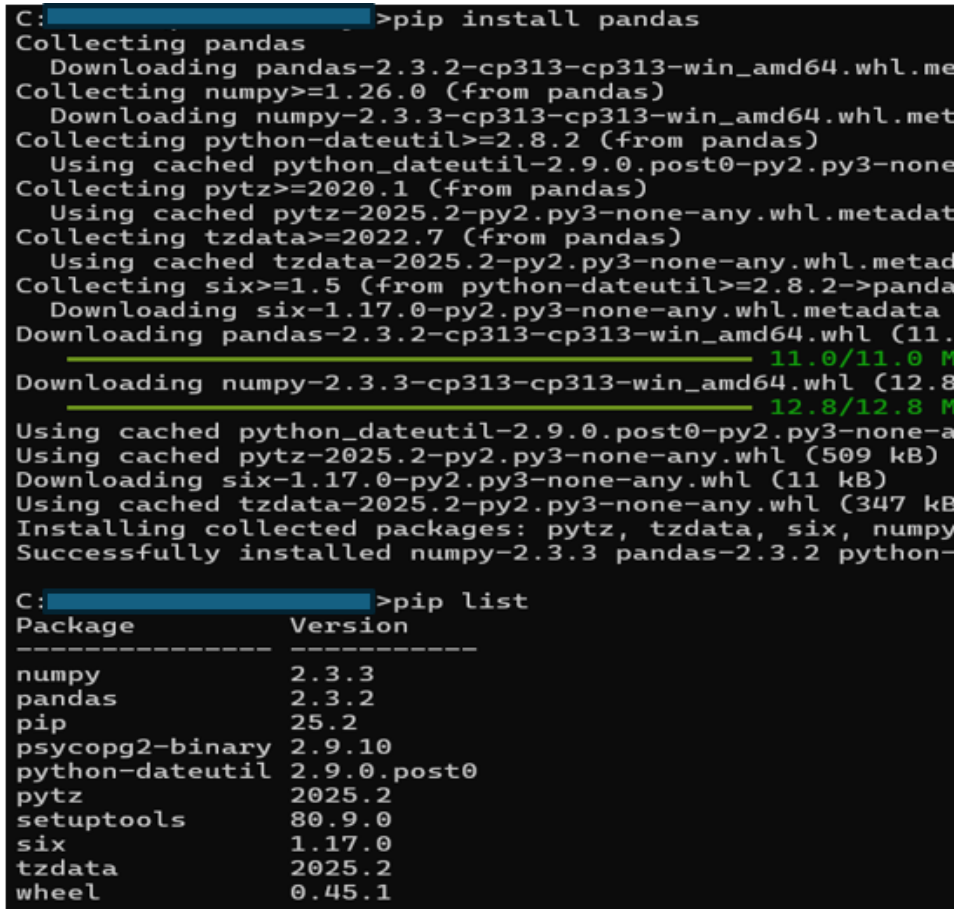
Verificar pacotes instalados:
pip list

Mostrar pacote especifico:
pip show nome-do-pacote



# PIP – Instalando pacotes

•Para instalar um pacote:
pip install nome-do-pacote
•Para instalar uma versão específica:
pip install nome-do-pacote==1.2.3
•Para atualizar:
pip install --upgrade nome-do-pacote
•Para remover:
pip uninstall nome-do-pacote



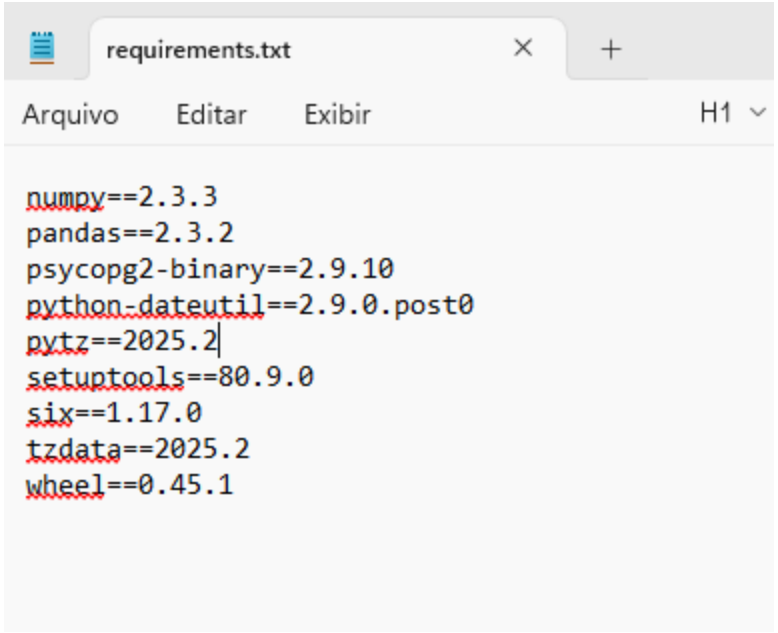
# PIP – Atividade

1 – Instalar o pacote pandas versão 2.3.0
2 – Verificar, pelo pip list, se o pandas aparece instalado.
3 – Atualizar o pacote pandas.
4 – Verificar se aconteceu a atualização
5 – Remover pandas e suas dependências.

# PIP – Usando requirements.txt

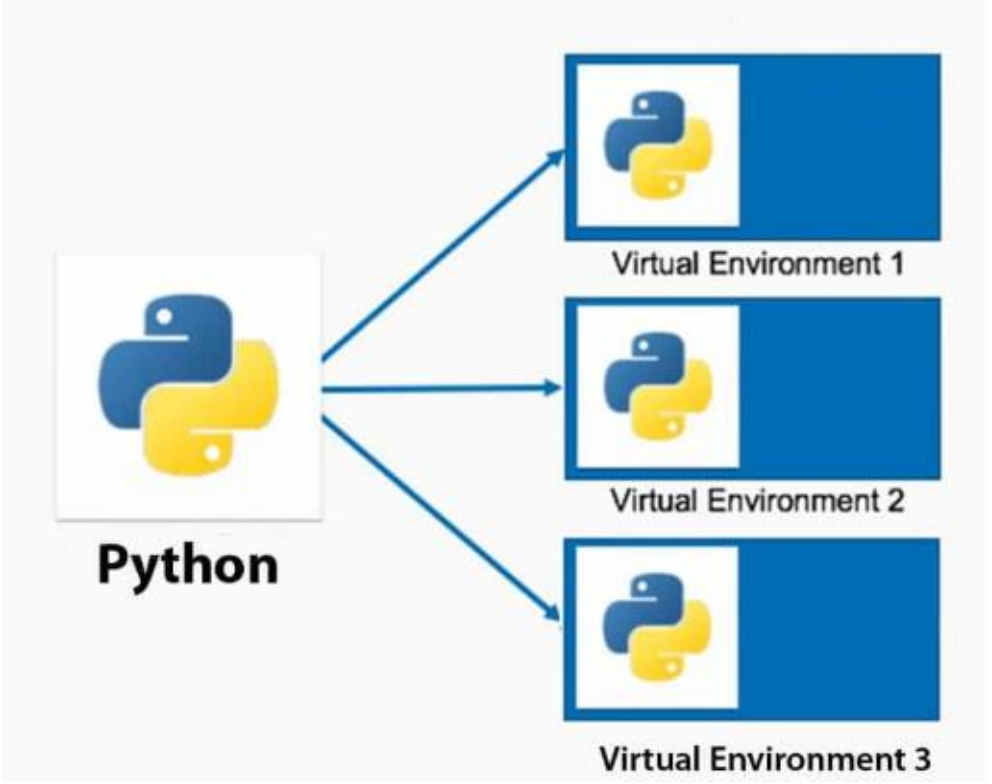
• O arquivo requirements.txt é usado para registrar todos os pacotes que o projeto precisa.
pip freeze > requirements.txt

• Permite que qualquer pessoa recrie o ambiente.
pip install -r requirements.txt



# Virtual env – O que é ?

- O venv é um ambiente virtual do Python.
- Permite criar uma instalação isolada do Python dentro de uma pasta.
- Evita conflitos de versões de pacotes entre diferentes projetos.
- Cada projeto pode ter seus próprios pacotes, sem afetar os outros.



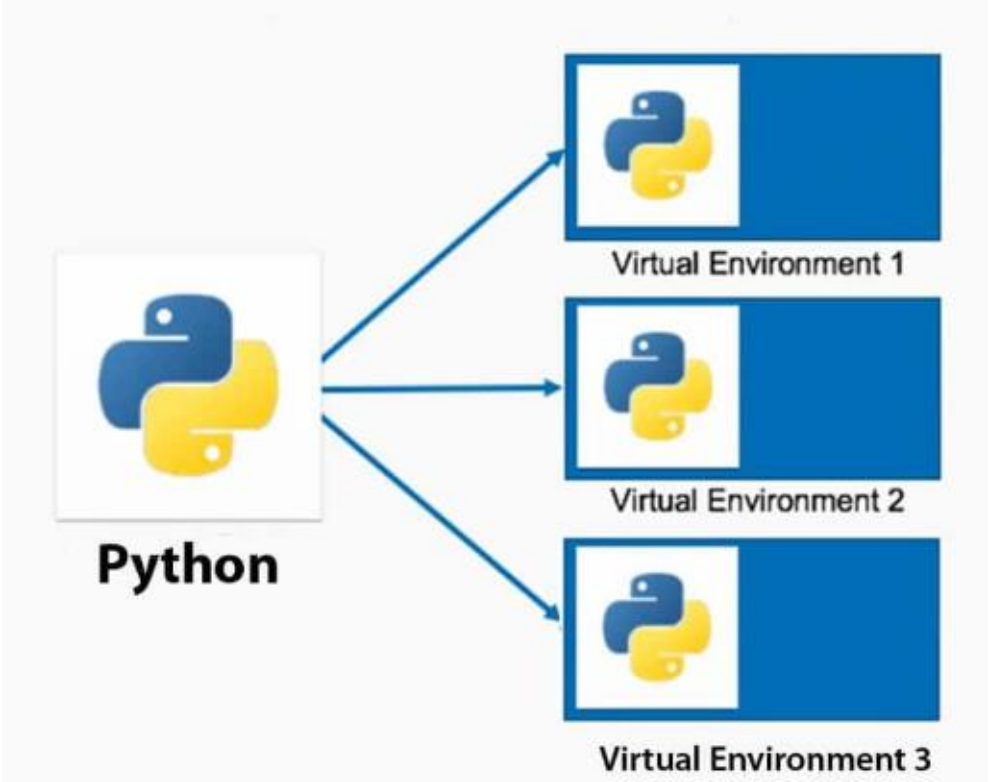
# Virtual env – Criando o ambiente

No terminal, dentro da pasta do projeto:

**Criando um venv:**
python -m venv venv

**Ativar o venv**
venv\Scripts\Activate

**Desativar:**
deactivate



# PIP – Atividade

1 – Na atividade do PIP crie um ambiente virtual venv.
2 – Ative o ambiente.
3 – Desative o ambiente.

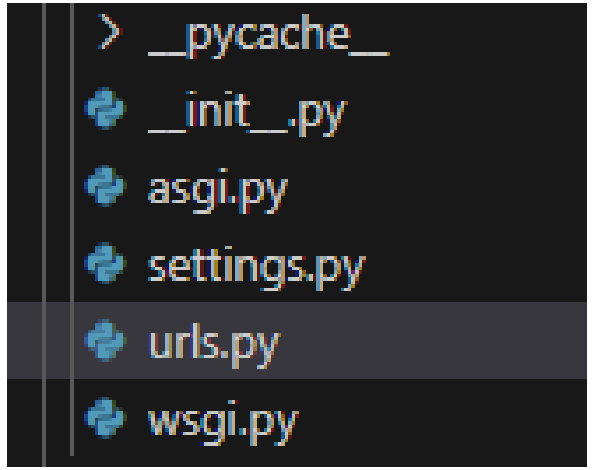
# Django – O que é ?

O que é o Django e que problema resolve:

- Django é um framework web em Python.
- Ele traz soluções prontas para:
    - Autenticação de usuários
    - Banco de dados (ORM)
    - Estrutura MVC/MVT
    - Rotas e views
- O problema que resolve: evitar reinventar a roda em projetos web, padronizando o desenvolvimento.

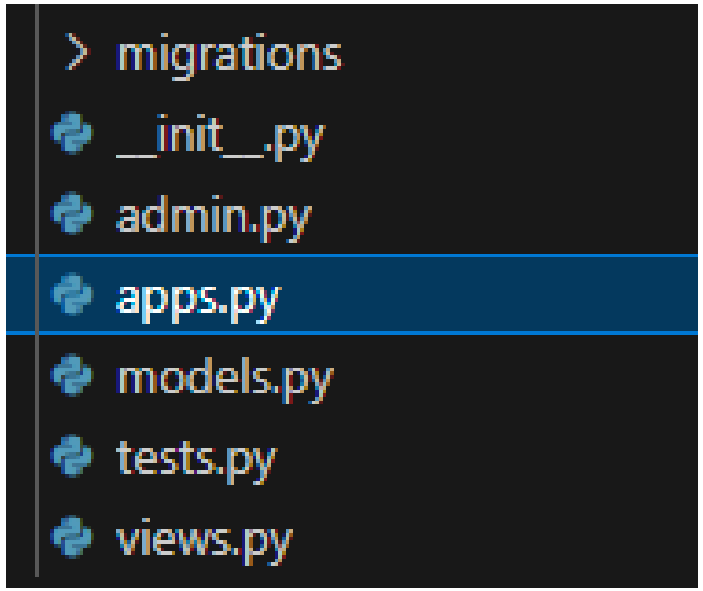
# Django – O que é um projeto no Django ?

• Um projeto é a base principal de uma aplicação Django.
• Contém as configurações globais do sistema (banco, segurança, apps instalados).
• Estrutura inicial:
• manage.py → gerencia comandos do projeto
• Pasta do projeto (ex.: meuprojeto/) com settings.py, urls.py, etc.



# Django – O que é um app Django ?

- Um app é um módulo dentro do projeto.
- Cada app tem uma responsabilidade específica.
- Estrutura inicial: models.py, views.py, admin.py, apps.py.



# Django – Como instalar o Django, criar projeto e app ?

**Instalação:**
pip install django
django-admin –version

**Criar um projeto:**
django-admin startproject nomeprojeto

**Criar um app:**
cd nomeprojeto
python manage.py startapp nomeapp

**Configurar app no projeto:**
No settings.py, adicionar nomeapp em INSTALLED_APPS

# Django – Rodando a aplicação

**Dentro da pasta do projeto:**
python manage.py runserver

**Acesse no navegador:**
http://127.0.0.1:8000

# Django – Atividade

Criando e rodando sua aplicação em Django.

1 - Criar repositório no GitHub
2 - Clonar repositório no computador
3 - Criar e ativar ambiente virtual (venv)
4 - Instalar o Django dentro da venv
5 - Criar projeto Django
6 – Criar um app.
7 - Rodar o Projeto.
8 – Se o projeto rodar normalmente, suba sua modificações para o github.

# Dúvidas ?

# Obrigado!


Instrutor: Pablo Araujo