Programação em Python (Back-end)

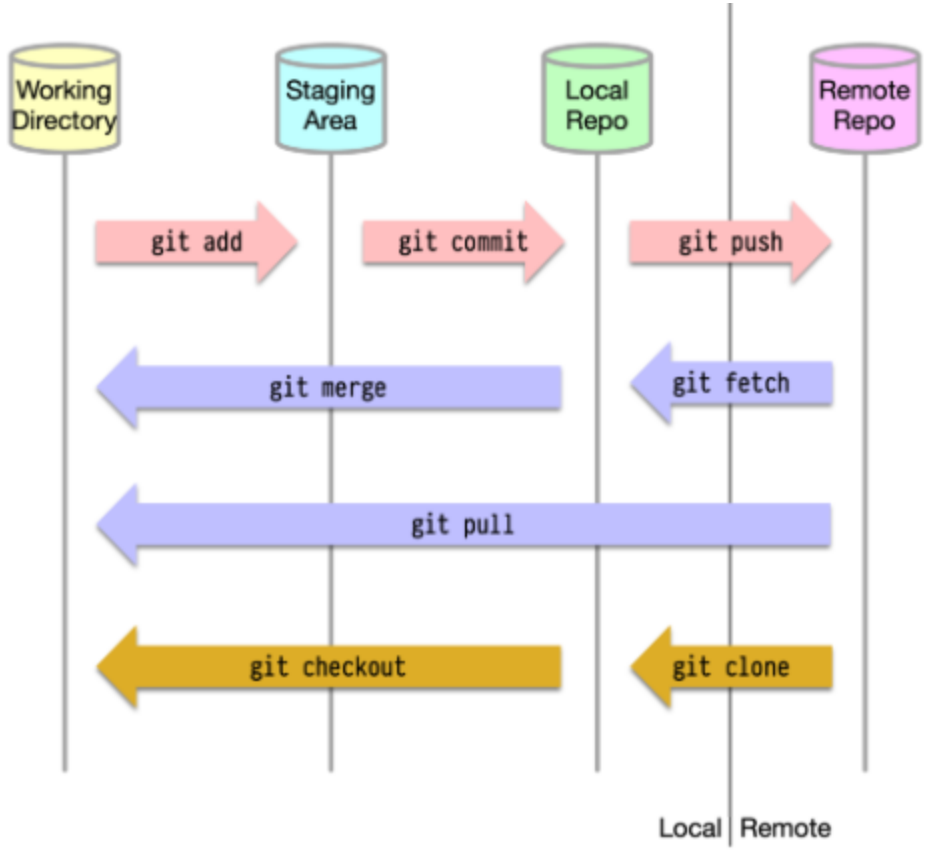
Instrutor: Pablo Araujo

(21) 97172-1697

Aula 6

Revisão GIT

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 - GIT

Objetivo

Apresentar dois fluxos comuns para começar um repositório Git:

- **Método A:** Criar o repositório no GitHub e clonar para a máquina local.
- **Método B:** Criar o repositório localmente e depois subir para o GitHub (remote).

Revisão - GIT

Método A — Criar no GitHub e clonar (fluxo rápido)

Quando usar: você quer que o repositório exista primeiro no GitHub, ou já solicitar para que o exista um readme ou gitignore.

Passo a passo:
1. No GitHub → botão **New repository** → escolha nome, descrição, visibilidade (Public/Private).
2. Copie a URL do repositório.
3. No terminal local: clonar o repositório

Revisão - GIT

Método B — Criar localmente e subir para o GitHub
Quando usar: você começa o projeto na sua máquina e só depois quer criar o repositório remoto.

Passo a passo:
1.No terminal, crie a pasta do projeto e inicialize o git:
git init

2. Crie arquivos básicos, pode ser um txt.

3. Adicione e faça o commit inicial:
git add .
git commit -m "chore: initial commit"

4. No GitHub, crie um novo repositório **sem** inicializar com README:

5. Adicione o remoto e faça o push:
git remote add ENDERECO_REPOSITORIO
git branch -M main
git push -u origin main

Revisão - GIT

Primeira atividade:
Crie um repositório remoto(github) chamado ClonandoProjeto e clone para sua máquina.
Para reforçar o aprendizado utilize o git clone com e sem “.” no final do comando via terminal.

Segunda atividade:
Crie uma pasta SubindoProjeto, inicialize o git e suba para o repositório remoto.

Revisão Classes

Revisão - Classe

Objetivo
Revisar os conceitos fundamentais de orientação a objetos em Python:
classes, atributos, métodos e herança.

Revisão - Classe

Atributos

Instância

• Pertencem a cada objeto.
• Definidos dentro do método __init__.

```
class Carro:
    def __init__(self, marca):
        self.marca = marca
```

Classe

• Compartilhados por todas as instâncias.
• Declarados diretamente dentro da classe.

```
class Carro:
    rodas = 4
```

Revisão - Classe

Herança

- Permite criar classes que herdam atributos e métodos de outra.
- **Superclasse → Subclasse**

```
class Veiculo:
    def mover(self):
        print("O veículo está em movimento")


class Carro(Veiculo):
    pass
```

Revisão - Classe

Desafio:

1. Crie uma classe Pessoa com atributos de instância nome e idade.
2. Adicione um método de instância que mostre nome e idade.
3. Crie uma subclasse Aluno que herda de Pessoa e tenha um atributo adicional curso.
4. Na classe Aluno, sobrescreva o método de instância para também mostrar o curso.
5. Crie objetos de Pessoa e Aluno e teste os métodos.

Revisão PIP e Venv

Revisão – PIP e Venv

Objetivo
Revisar como gerenciar pacotes e ambientes virtuais em Python usando pip e venv.

Revisão – PIP e Venv

PIP — Gerenciador de pacotes
• **O que é:** ferramenta padrão do Python para instalar bibliotecas externas.
• **Uso principal:** instalar, atualizar ou remover pacotes.

Comandos mais comuns:
- Pip install nome_projeto
- Pip install nome_projeto==versão
- Pip install --upgrade nome_projeto
- Pip uninstall nome_projeto
- Pip list
- Pip freeze > requirements.txt
- Pip install –r requirements.txt

Revisão – PIP e Venv

VENV — Ambiente virtual

• O que é: cria um ambiente isolado de pacotes para cada projeto.
• Por que usar**:** evita conflitos de versões entre projetos.

Criando e ativando um ambiente virtual:
python -m venv .venv
.venv\Scripts\Activate

Desativar ambiente virtual:
Deactivate

Excluir ambiente virtual:

• Basta deletar a pasta .venv

Revisão – PIP e Venv

Atividade:
1 – Vamos simular que um projeto foi criado e ao finalizar o projeto o desenvolvedor salvou a lista dos pacotes utilizados no requirements. Então para utilizar o projeto apenas é necessários instalar o venv e instalar os pacotes via requirements.
Vamos fazer esse fluxo:

• Crie uma pasta chamada meu_projeto.
• Dentro dela, crie um ambiente virtual usando venv e ative-o.
• Instale os seguintes pacotes usando pip: Django e pandas
• Liste os pacotes instalados no ambiente e confirme que foram instalados.
• Gere um arquivo requirements.txt com as dependências do projeto.
• Delete a pasta do ambiente virtual.

“**Nesse ponto imagine: Você é um desenvolvedor novo no projeto e acabou de cloná-lo”**

• Recrie-o. Em seguida, use: pip install -r requirements.txt

Revisão Django

Revisão - Django

Objetivo
Revisar os conceitos fundamentais do **Django**, um dos frameworks web mais usados em Python.

Revisão - Django

Instalando o Django

1. Crie e ative um ambiente virtual (boa prática).
2. Instale o Django com pip: pip install django
3. Verifique a instalação: django-admin --version

Revisão - Django

Projeto Django
•Projeto**:** estrutura principal que configura todo o ambiente da aplicação web.
•Contém arquivos de configuração, banco de dados e mapeamento de URLs.

Criando um projeto:
django-admin startproject meu_projeto
cd meu_projeto
python manage.py runserver

App Django
Módulo dentro do projeto responsável por uma parte da aplicação (ex.: blog, autenticação, produtos).
python manage.py startapp meu_app

Registrando apps no projeto
1. Abra o arquivo settings.py do projeto.
2. Localize a lista INSTALLED_APPS.
3. Adicione o nome do app

Revisão - Django

Criando um superusuário

O superusuário é usado para acessar o Django Admin (painel administrativo).

1. Rodar migrações iniciais do banco: python manage.py migrate
2. Criar superusuário: python manage.py createsuperuser
3. Informe usuário, e-mail e senha.
4. Inicie o servidor e acesse: http://127.0.0.1:8000/admin/
5. Faça login com o superusuário criado.

Revisão - Django

Atividade Integrada — Git, GitHub, venv, pip e Django

Objetivo

Simular o ciclo real de criação de um projeto Python com Django, controlado por versionamento no Git/GitHub, usando ambientes virtuais e pacotes externos.

1. Criar repositório no Github
2. Clonar o repositório do GitHub
3. Criar um ambiente virtual
4. Instalar o Django
5. Criar o Projeto
6. Criar o App
7. Registrar o App
8. Criar um superuser
9. Testar o Login do superuser

Desafio:

1. Criar a model Cliente.
2. Registrar a model no admin
3. Cadastrar pela interface admin 5 clientes.

Dúvidas ?

Obrigado!

Instrutor: Pablo Araujo