Criando Ambientes Python e versionando código com o Git e Github via VSCode 🐍

Essa Documentação tem o objetivo de demonstrar o processo de criação de um ambiente de desenvolvimento para Python, bem como o uso do Git e Github para versionamento de código. Também mostra algumas dicas de como utilizar o Streamlit para geração de visualizações de dados.

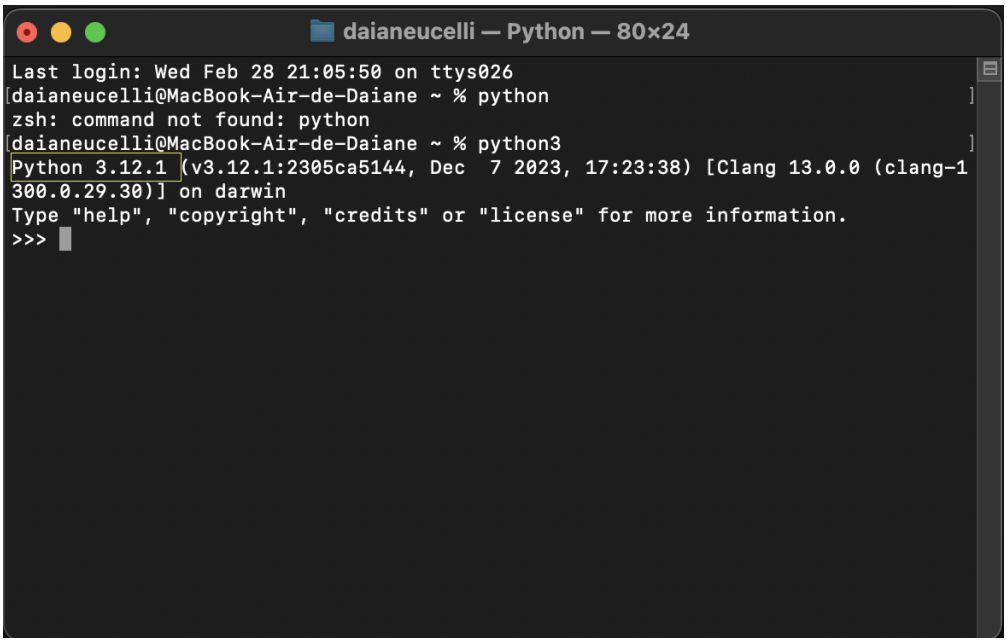
Configurando um ambiente Python (env):

1. Qual a importância de ter um ambiente Python criado no diretório do seu projeto?

A resposta mais simples é: isolar o ambiente criado de todo o ecossistema Python que existe na sua máquina. Fazemos isso porque projetos diferem entre si e utilizam bibliotecas diferentes para cada finalidade. Portanto, não seria necessário, por exemplo, tem uma biblioteca voltada para Ciência de Dados em um diretório/repositório em que estamos trabalhando com Análise de Dados.

Outro ponto é que em alguns projetos, utlizamos uma versão específica do Python ou até mesmo das bibliotecas utilizadas, essas que estão em constante desenvolvimento e atualizações, e o que você está usando agora - neste projeto, por exemplo - pode sofrer uma alteração crítica numa dessas atualizações e caso você atualize a versão de algumas dessas, essa funcionalidade mude e isso pode quebrar o seu código. Dito isso, é importante isolar seu projeto do ecossistema como um todo.

Estamos partindo do ponto em que o Python esteja corretamente instalado em sua máquina, para isso é fácil verificar abrindo um novo prompt de comando e digitar: python ou python3, vai depender do Sistema Operacional utilizado. Se estiver tudo certo, a versão mais recente do Python instalado em sua máquina, vai ser vista na resposta à linha de comando, assim:



No momento que escrevo esse texto a versão mais atual do Python é a 3.12 (que é a que estou usando aqui).

Criando um repositório:

Vamos criar um repositório para guardar todo o progresso e conseguir fazer o versionamento do nosso código. Utilizo o git, mas existem outras ferramentas para o mesmo propósito.

Para usar o git, é necessário instalar e configurar em sua máquina, aqui mais uma vez dependeremos do Sistema Operacional pois os comandos variam um pouco.

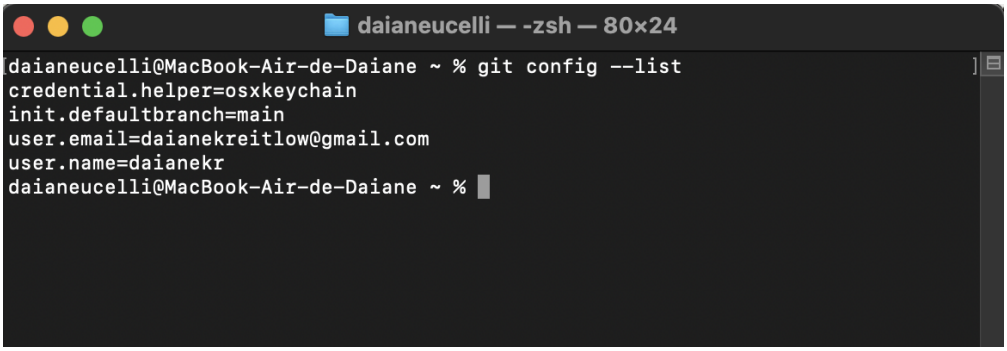
Link para baixar o git!

Na documentação do Git existe um passo a passo de como configurá-lo, o processo é bem simples.

Configurando o git!

Caso dê tudo certo com aconfiguração, conseguimos checar via prompt de comando, digitando:

```
git config --list
```



Agora já podemos nos comunicar com a nossa conta do Github.

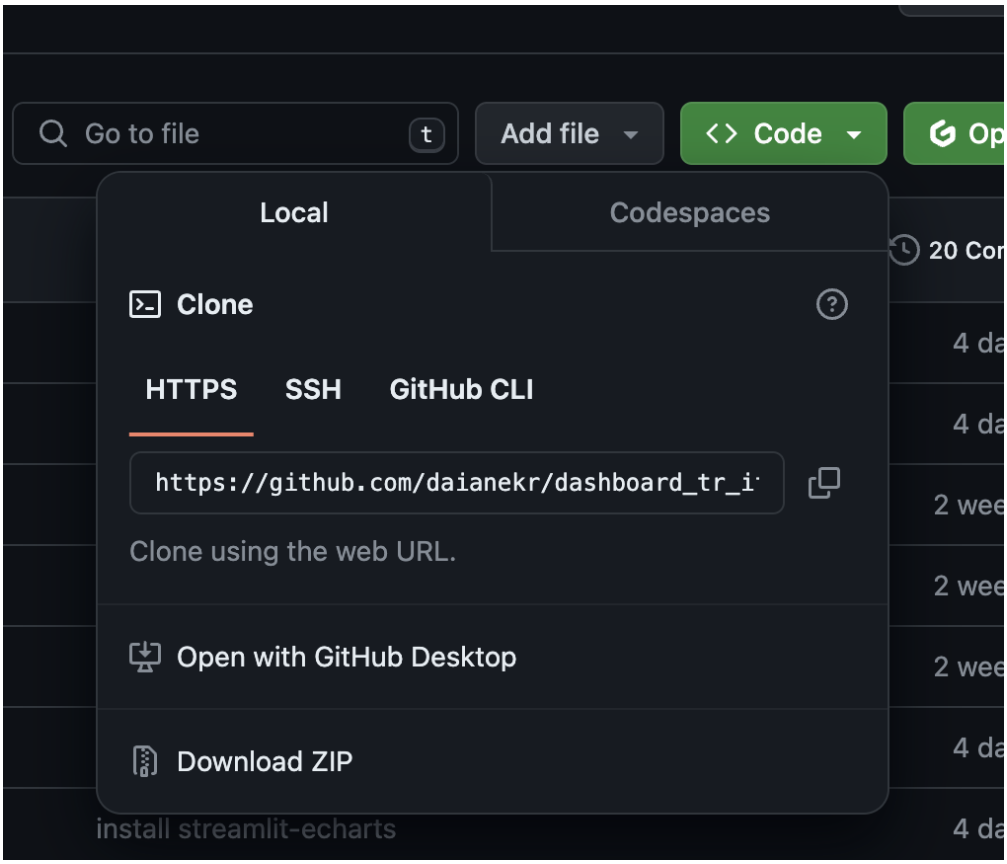
Esse artigo é bem interessante para entender como essa comunicação é feita.

Bom, existem várias maneiras de criar um repositório e ir versionando o código. Eu vou mostrar uma bem simples e que funcionará para o que queremos fazer.

1. Crie um repositório diretamente na interface do Github.

- Você vai entrar em sua conta e criar esse repositório vazio (como no tutorial da documentação do Github acima), com um arquivo readme e eu recomendo que deixe esse repositósio privado. Vamos trabalhar com dados e provavelmente dados sensíveis que não queremos expor (ainda).

- Como o repositório criado, vamos clonar o repositório vazio para nossa máquina com o comando: git clone <link do repositório criado>. Você vai acessar o link HTTPS do repositório pela interface do Github. Assim:



Há várias maneiras de fazer esse clone do projeto, via HTTPS como aqui, via SSH, via CLI, via github desktop. Esse mostrado aqui funciona bem e não é complexo.

Criando o Ambiente Virtual Python para o Projeto.

Para criar um ambiente Python, abra o seu projeto dentro do VSCode, abra um novo terminal pelo próprio VSCode e rode o comando*:

*nome_ambiente_virtual vai ser o nome do seu ambiente. Ex: env, venv, venv_nomedoprojeto.

substitua nome_ambiente_virtual, pelo nome do ambiente que você quer criar.

```
python3 -m venv nome_ambiente_virtual
```

com o ambiente criado, precisaremos ativá-lo. Ainda no terminal do próprio VSCode, vamos ativar o ambiente com um comando específico para cada SO.

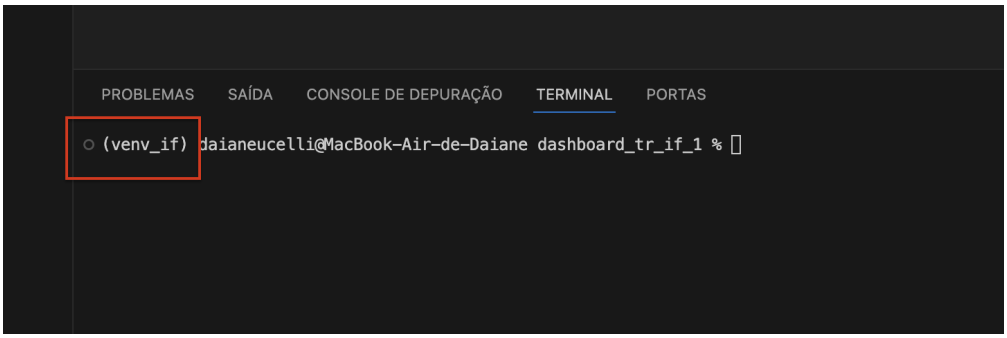
No MAC ou Linux:

```
source nome_ambiente_virtual/bin/activate
```

No Windows:

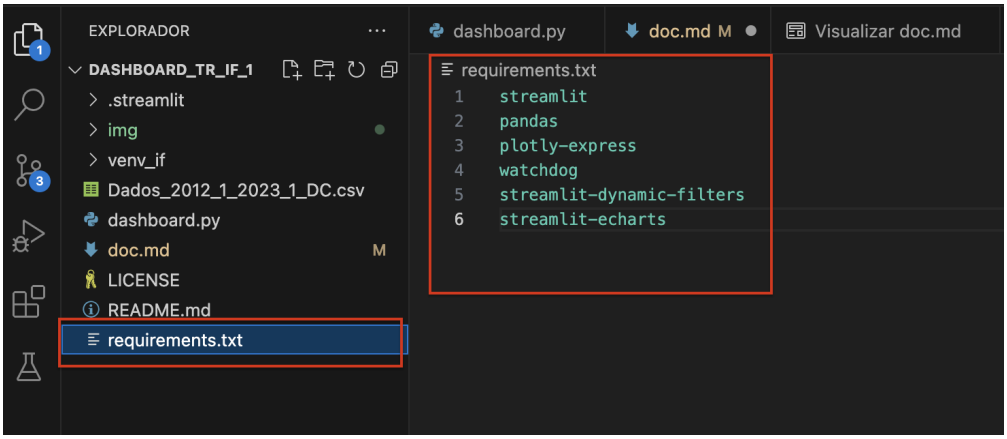
```
nome_ambiente_virtual/Scripts/Activate
```

Se o ambiente virtual for ativado com sucesso, será possível ver no terminal o nome_ambiente_virtual entre parênteses, assim, por exemplo:



Gerenciando pacotes usados no seu projeto:

Uma maneira simples de gerenciar todos os pacotes Python que utilizará em seu projeto, é criando um arquivo e o renomeando para requirements.txt. Desse maneira, você pode simplesmente listar todos os pacotes que quer instalar.



Você também pode especificar o versão do pacote que está usando. Aqui você pode ir atualizando os pacotes conforme for precisando/criando novos.

Para instalar, salve o arquivo (ctrl+S/cmd+S), e rode o comando a seguir, no mesmo terminal já aberto anteriormente e com a venv ativada! Isso vai garantir que todos os seus pacotes estarão sendo instalados somente nesse ambiente e que nenhum outro ambiente do seu Python local será afetado por eles.

MAC ou Linux:

```
pip3 install -r requirements.txt
```

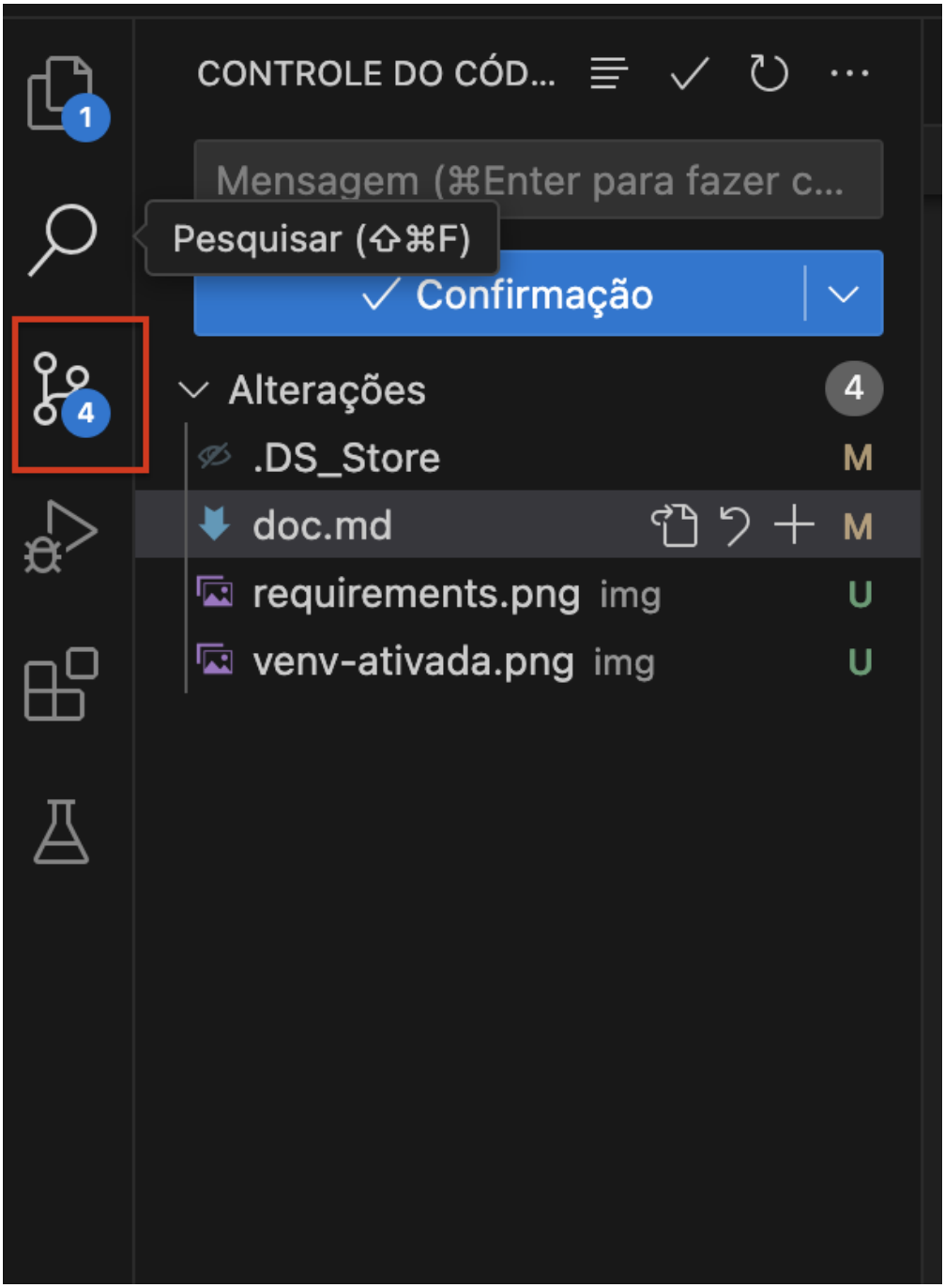
No Windows:

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

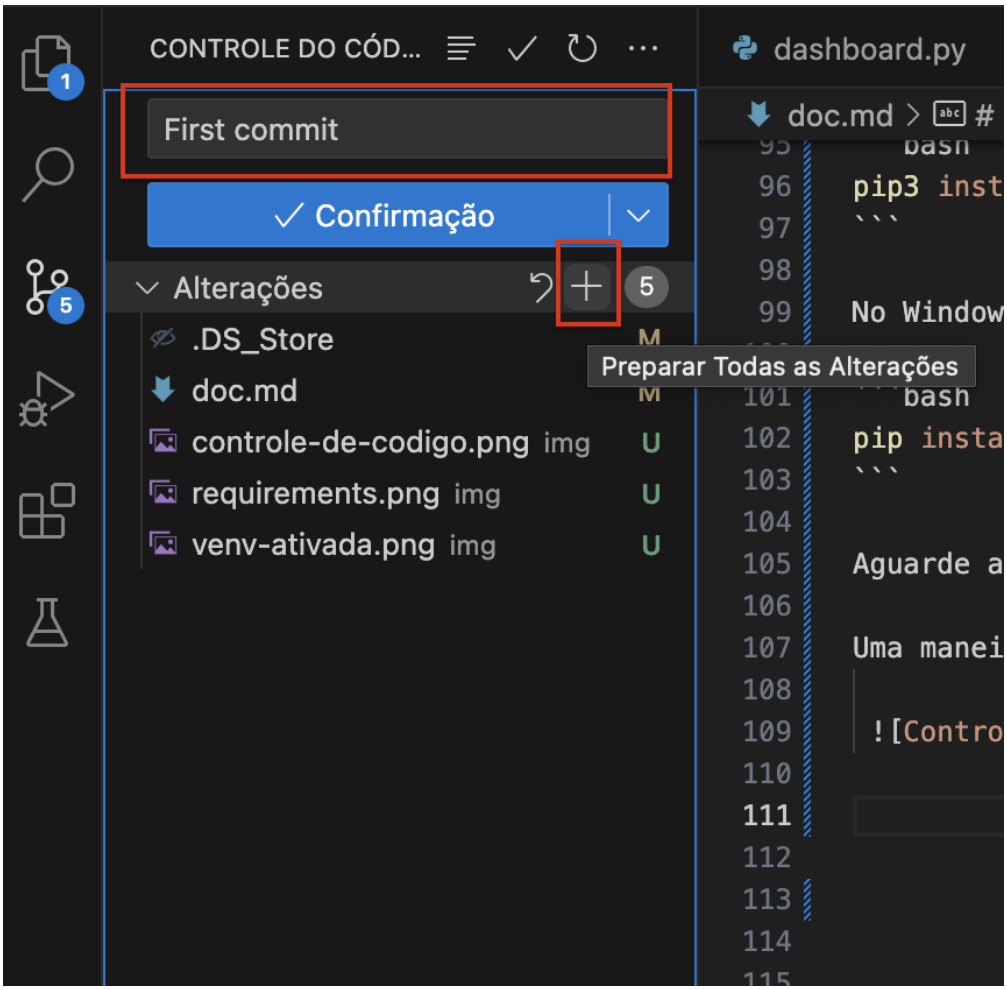
Aguarde a instalação. E o ambiente estará pronto pra uso.

Controle de Código Via Interface do VSCode:

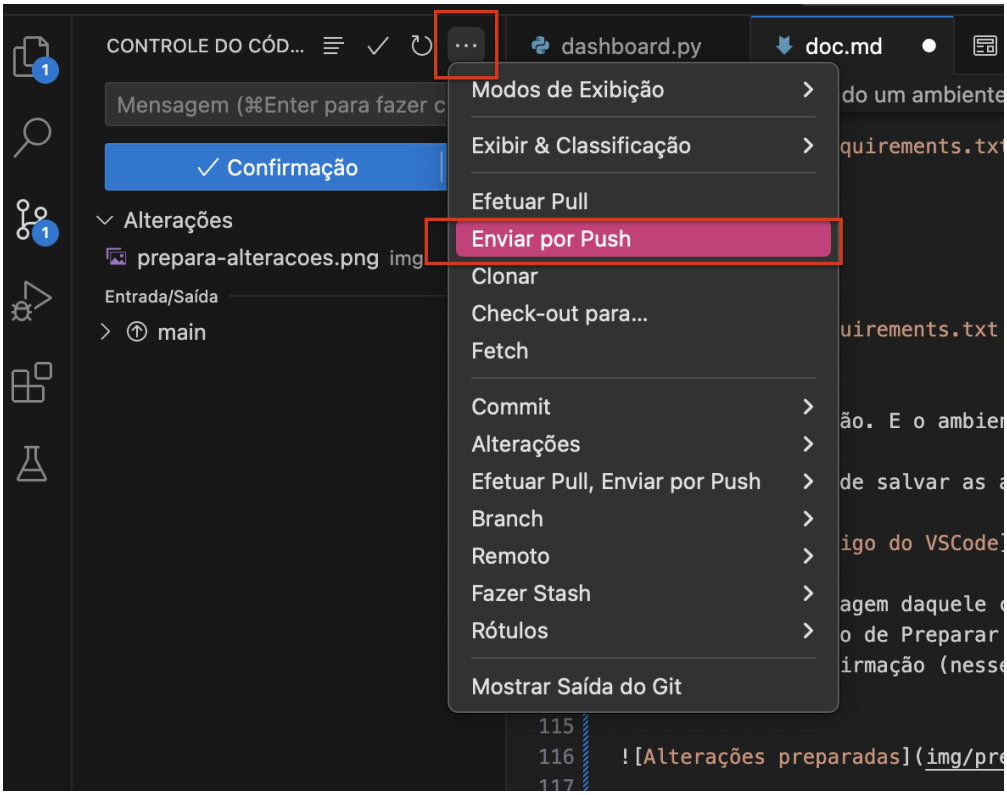
Uma maneira fácil de salvar as alterações do seu arquivo via git dentro do próprio VSCode é ir em controle de código:



1. Digitar a mensagem daquele commit (obrigatório),
2. clicar no botão de Preparar todas as alterações (obrigatório),
3. clicar em Confirmação (nesse ponto todos as alterações estão preparadas para "subir" para o Github)



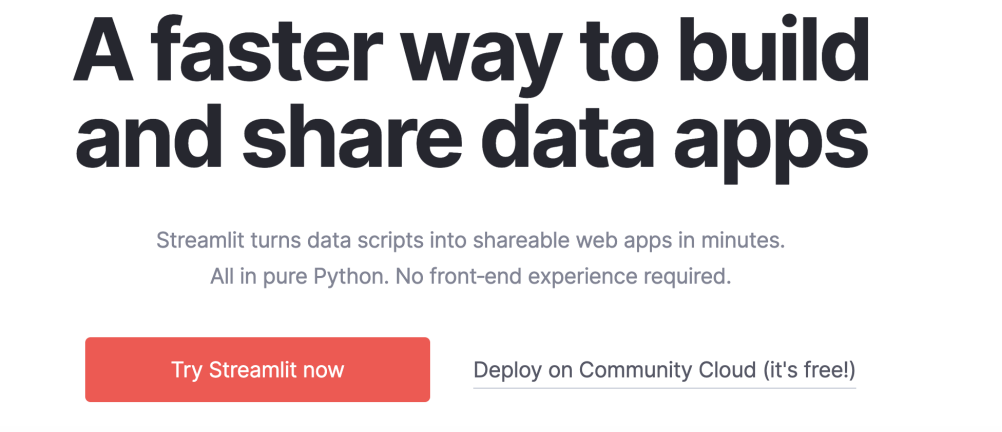
4. ir em Modos de Exibição e Mais ações (mais conhecido como 3 pontinhos rs) e clicar em Enviar por push.



Pronto! Suas alterações estarão todos no seu repositório do Github.

A Biblioteca Streamlit:

Podemos criar uma aplicação em minutos usando o Python com o apoio da biblioteca Streamlit. Ela também permite processo de deploy facilitado via Github.



Para começar a usar, precisamos instalar (pode ser via requirements que vimos acima) ou fazendo a instalação diretamente do terminal.

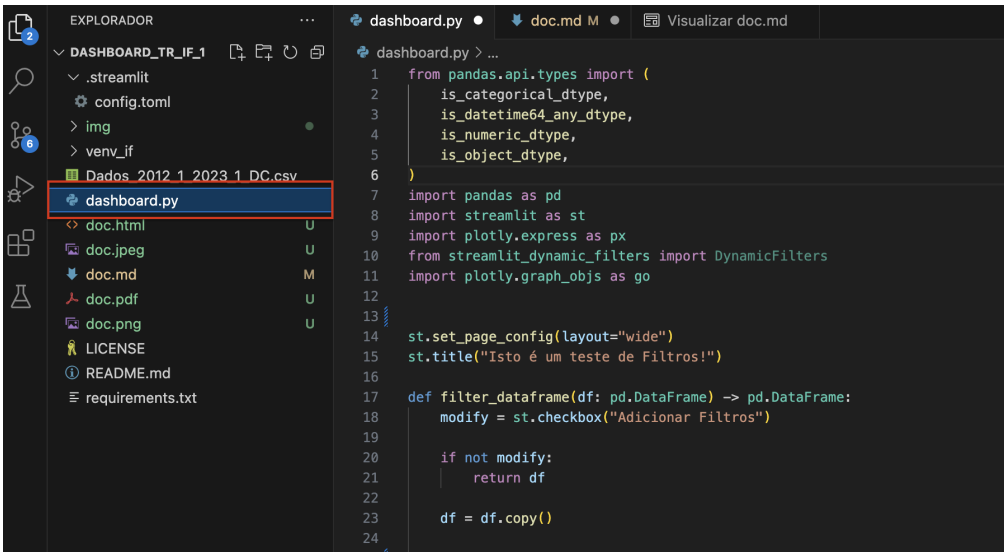
Linux e MAC:

```
pip3 install streamlit
```

Windows:

```
pip install streamlit
```

Para começar a usar, crie um arquivo que atuará como sua main, e vá criando seus códigos e visualizações nesse arquivo. Nesse exemplo, eu criei o arquivo com o nome dashboard.py, e nele fiz a importação das bibliotecas e as visualizações que eu queria ver.



Nesse exemplo foi usado o Pandas, o Streamlit, o Plotly. A documentação é bem vasta e com bastante exemplos prontos pra uso, necessitando das alterações que façam sentido no seu projeto.

Nesse exemplo abaixo, é gerado um título e a leitura de uma DataFrame com sua vizualização:

```
from pandas.api.types import (
    is_categorical_dtype,
    is_datetime64_any_dtype,
    is_numeric_dtype,
    is_object_dtype,
)
import pandas as pd
import streamlit as st
import plotly.express as px
from streamlit_dynamic_filters import DynamicFilters
import plotly.graph_objs as go


st.set_page_config(layout="wide")
st.title("Hello World! 🌞 ")

df = pd.read_csv("seus_dados_em_csv_aqui.csv", sep=",")

st.dataframe(df)
```

se salvamos o arquivo, e rodarmos no terminal da nossa venv o comando:

```
streamlit run dashboard.py
```

Ele vai executar e já abrir no navegador localmente sua nova página criada com Python.