

Aula #11: Revisão dos Principais Conceitos e Comandos Básicos

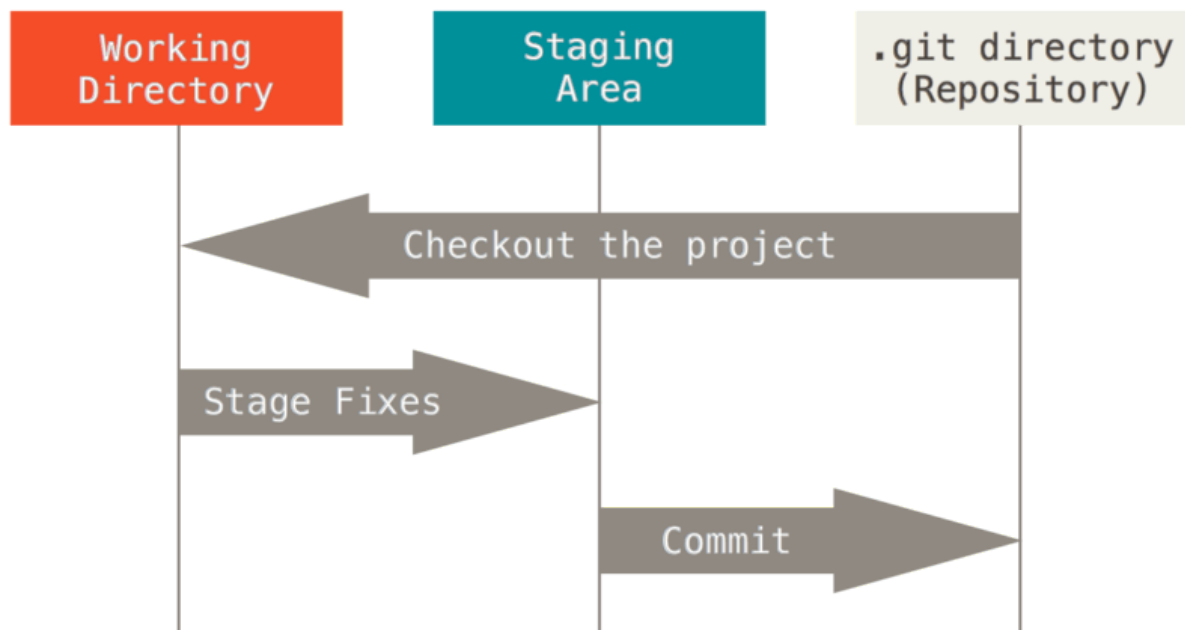
Três Áreas do Git	1
Principais comandos básicos	2
Inicializar um repositório local	4
Adicionar novos arquivos para a Área de Stage - Marcar como prontos	4
Salvar alterações que estejam marcadas como prontas	5
Desfazendo uma alteração	6
Desmarcando alterações marcadas como prontas	8
Exercícios	9
Próxima Aula	9
Fontes e Links Complementares	10

Três Áreas do Git

Como vimos nas aulas anteriores, o Git possui três grandes áreas de trabalho:

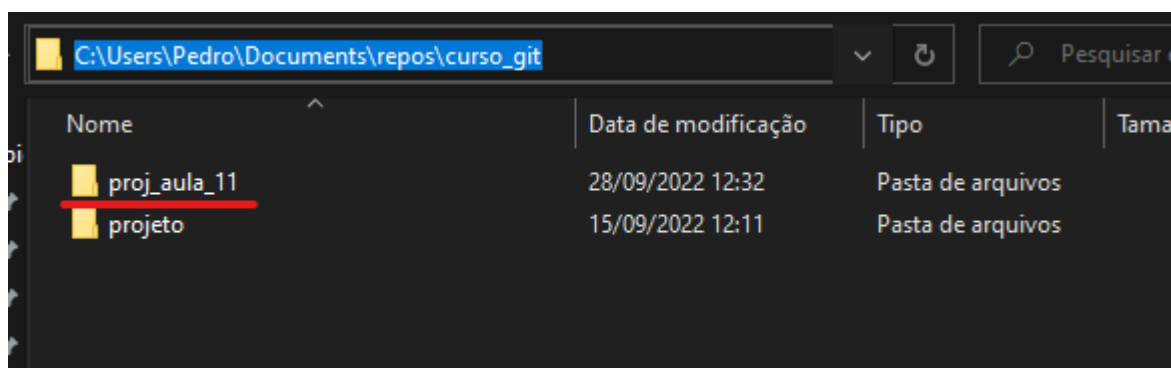
- Área de Repositório (.git Repositório)
- Área de Trabalho (Working Directory)
- Área de Staging (Staging Area)

Essas três áreas interagem entre si conforme utilizamos os comandos para adicionar, salvar ou remover os arquivos do nosso repositório.

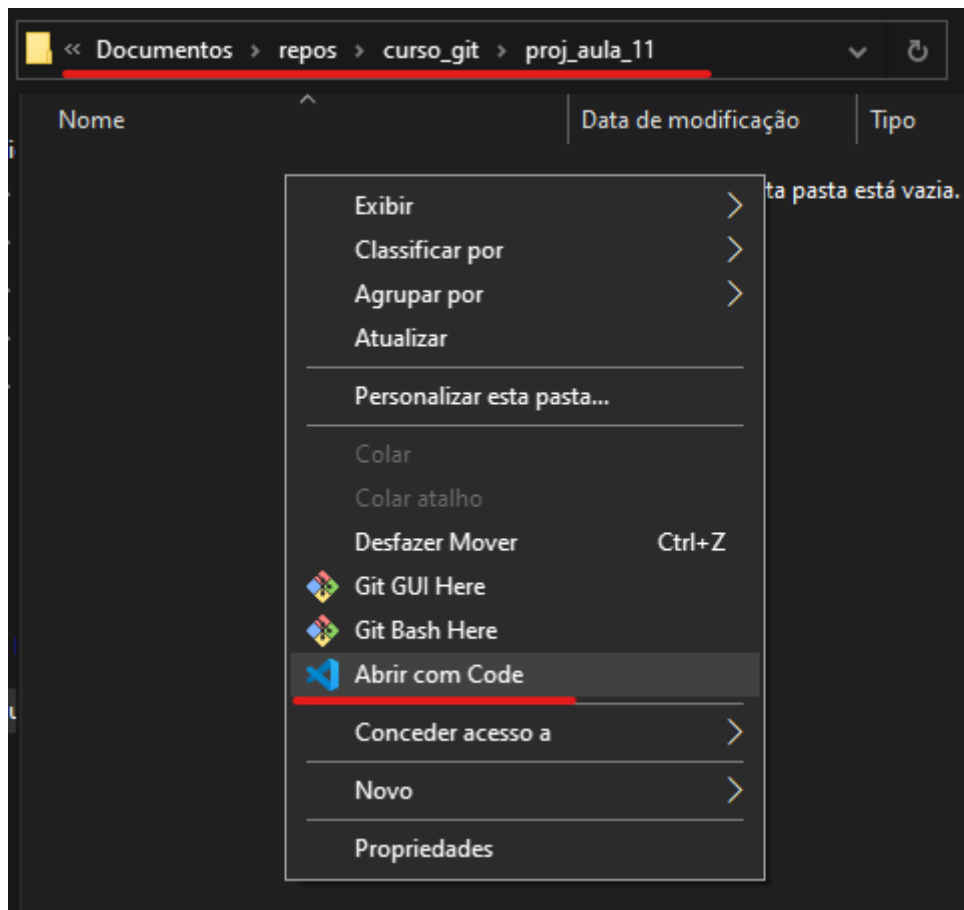


Principais comandos básicos

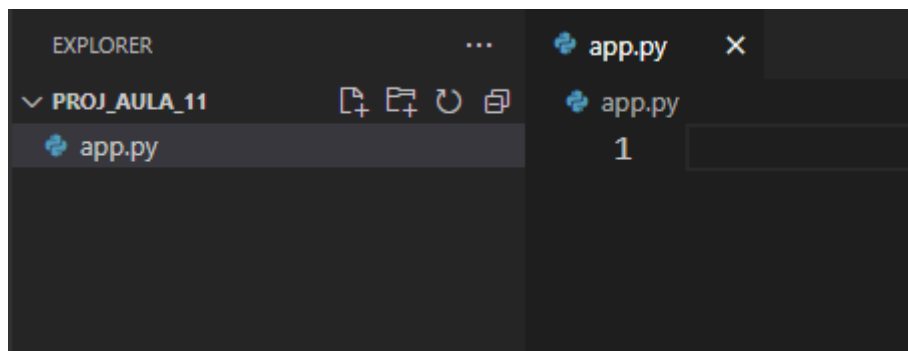
Para começar, vamos criar uma nova pasta dentro da pasta de projetos do curso chamada `proj_aula_11`:



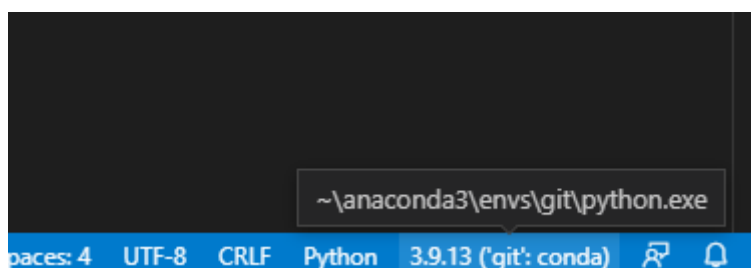
Com a pasta criada, entre nela, clique com o botão direito e selecione Abrir com Code:



Dentro da IDE VS Code, crie um arquivo Python chamado `app.py`:



E selecione o ambiente de desenvolvimento `curso_git`, no canto inferior direito:



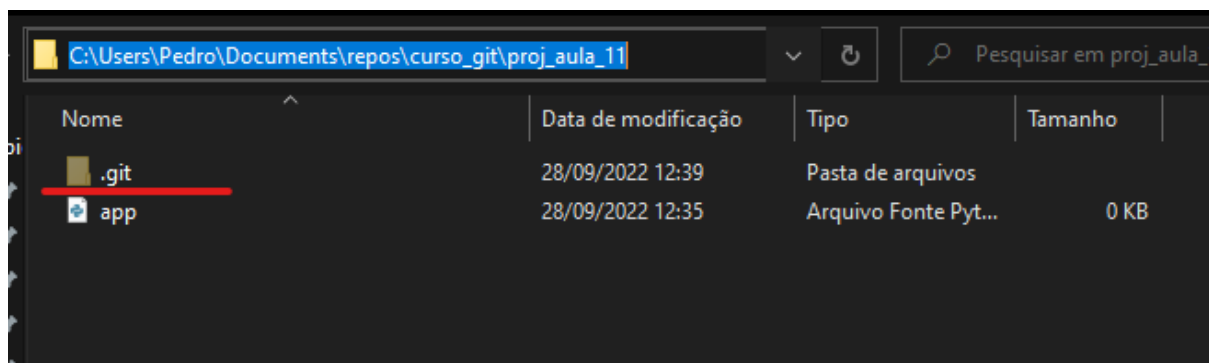
Feito isso, vamos começar com a revisão dos principais comandos vistos no ciclo anterior.

Inicializar um repositório local

Para inicializar um repositório local, temos que utilizar o comando `git init`, dentro do diretório raiz do projeto. Para isso, basta acessar o terminal do Git dentro do VS Code e digitar o comando `git init`:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11
$ git init
Initialized empty Git repository in C:/Users/Pedro/Documents/repos/curso_git/proj_aula_11/.git/
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$
```

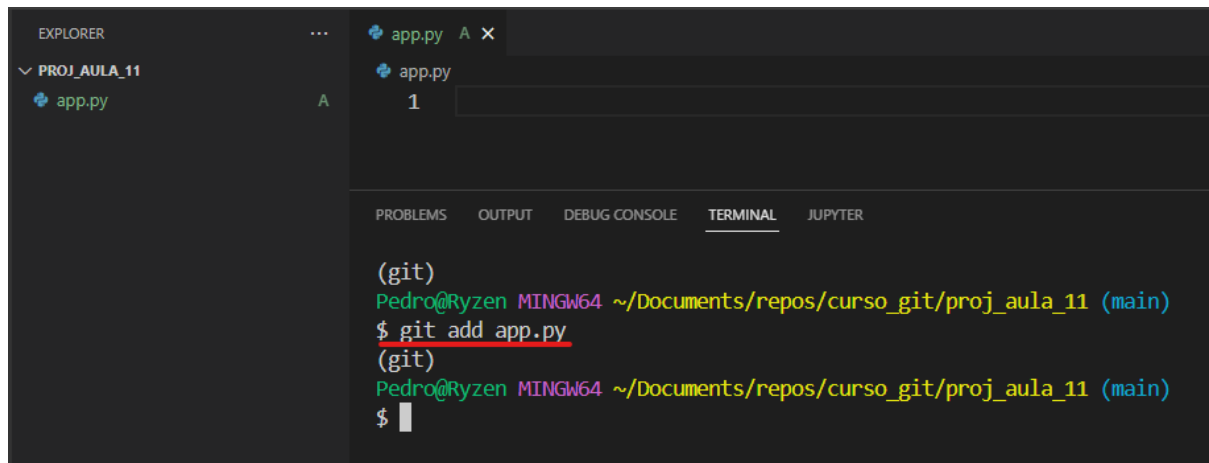
Esse comando irá inicializar um novo repositório local na pasta `proj_aula_11`, conforme podemos verificar no explorador de arquivos:



Adicionar novos arquivos para a Área de Stage - Marcar como prontos

Para marcarmos arquivos adicionados ou alterados como prontos para serem salvos, ou seja, colocarmos essas alterações do repositório dentro da `Área de Stage`, utilizamos o comando `git add <nome_arquivo>`. Lembrando que podemos utilizar o atalho `git add .` quando desejamos adicionar todos os

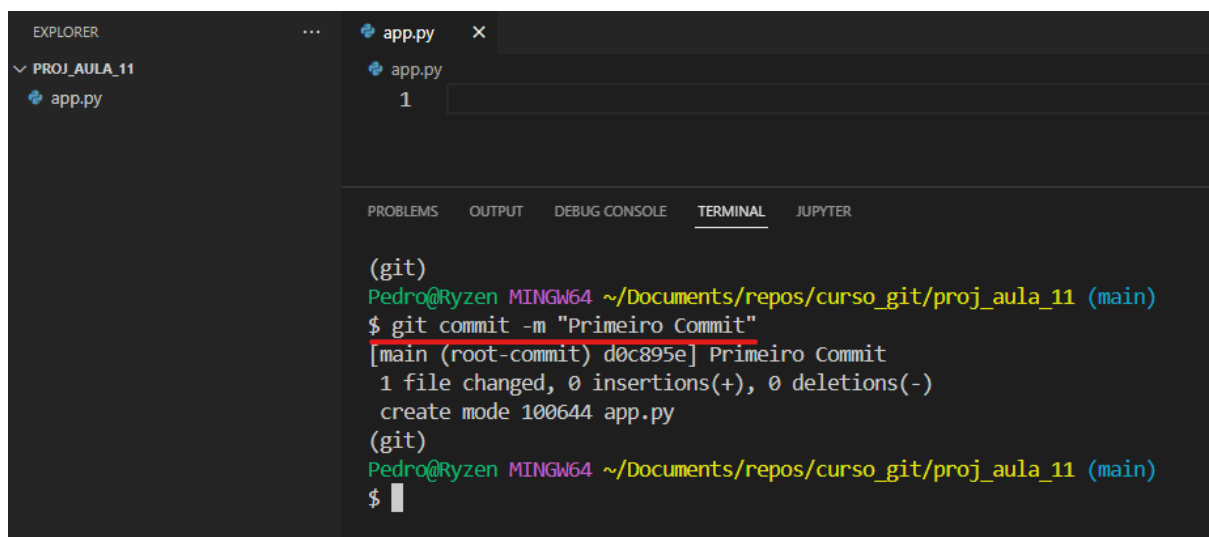
arquivos que estão na **Área de Trabalho (Working Directory)** para a **Área de Stage**:



```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git add app.py
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$
```

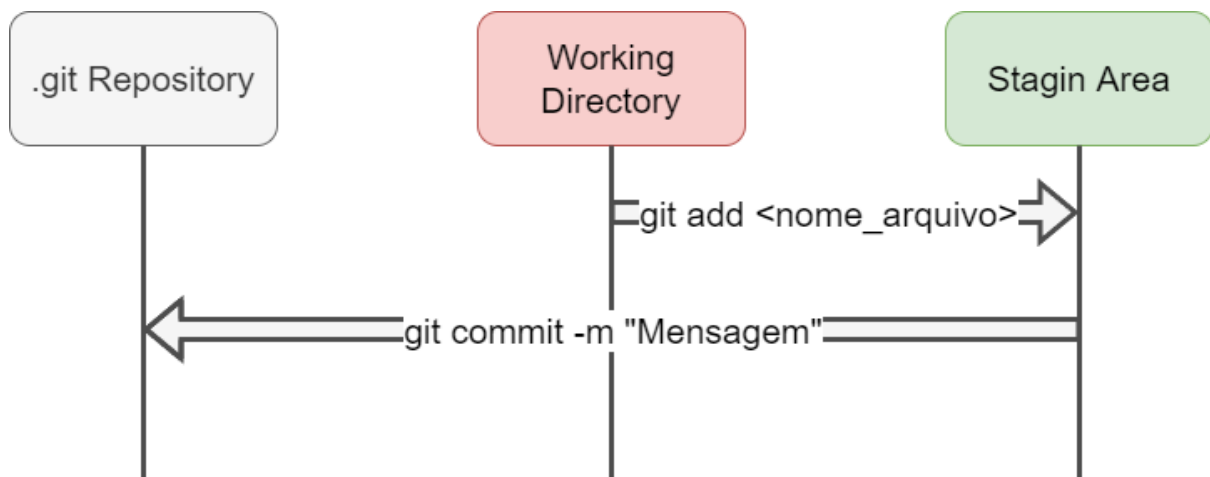
Salvar alterações que estejam marcadas como prontas

Uma vez que marcamos as alterações dos arquivos como prontas com o comando `git add`, utilizamos o comando `git commit` para salvarmos o estado dessas alterações, informando o Git que agora esse é o novo estado do arquivo.



```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git commit -m "Primeiro Commit"
[main (root-commit) d0c895e] Primeiro Commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 app.py
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$
```

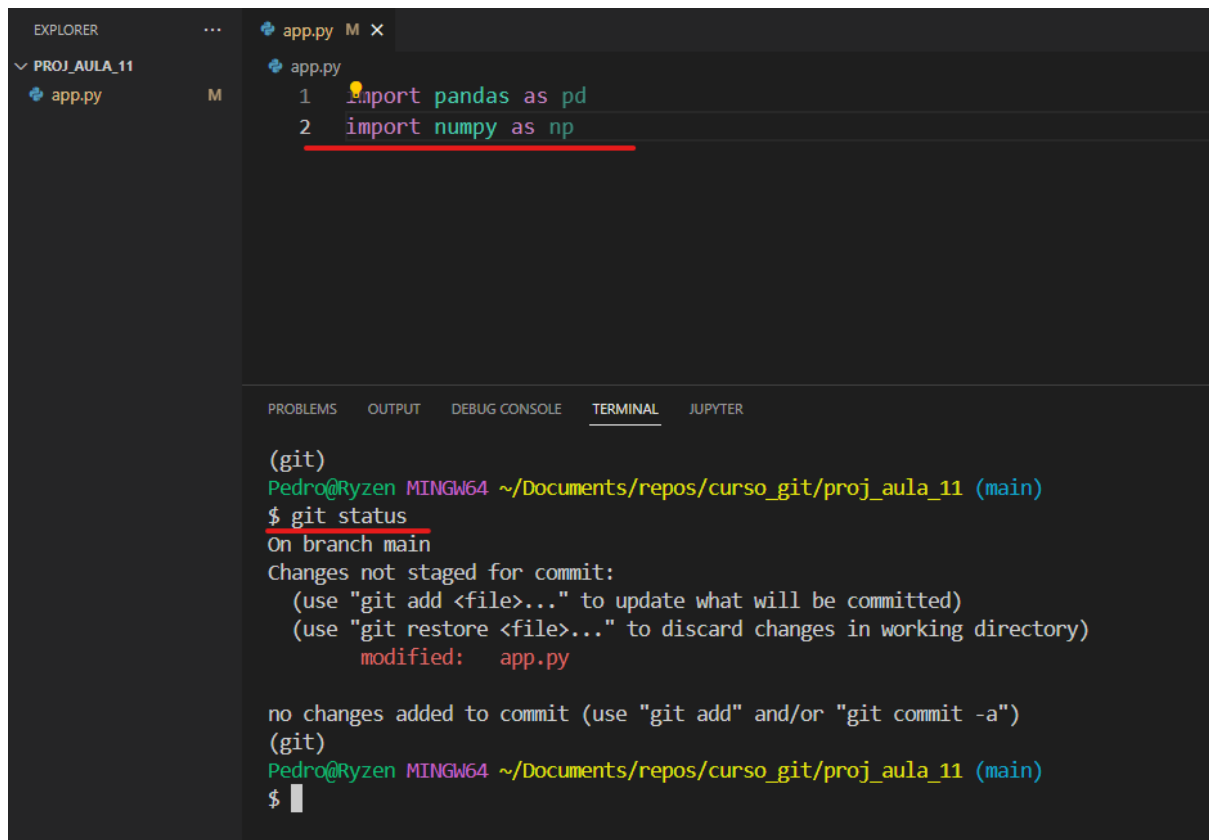
Com isso, passamos por todo o ciclo de vida de um arquivo recém adicionado.



Esse processo, adicionar à **Área de Stage** com o comando `git add` e salvar as alterações com o comando `git commit`, pode ser executado tanto para arquivos recém adicionados como para arquivos já versionados pelo Git.

Desfazendo uma alteração

Quando um arquivo está versionado, como é o caso do nosso arquivo `app.py`, caso façamos alterações nele e precisemos desfazer e voltar ao estado original, podemos utilizar o comando `git restore`. Para exemplificar esse processo, vamos adicionar o uso das bibliotecas `Pandas` e `Numpy` no arquivo:

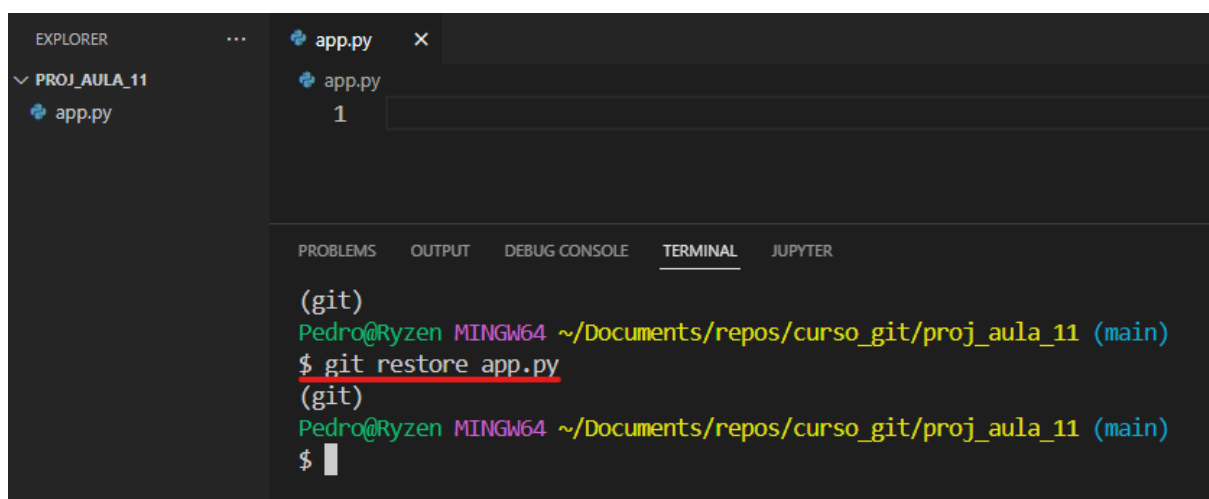


The screenshot shows the Visual Studio Code interface. In the Explorer panel on the left, a project named 'PROJ_AULA_11' is open, containing a file 'app.py'. The editor window shows the content of 'app.py' with two lines of code: `import pandas as pd` and `import numpy as np`. The second line is underlined in red. Below the editor, the TERMINAL panel is active, displaying the output of the `git status` command. The output indicates that the file 'app.py' has been modified and is not staged for commit.

```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app.py

no changes added to commit (use "git add" and/or "git commit -a")
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$
```

Lembrando que utilizamos o comando `git status` para verificar como estão as áreas de trabalho do nosso repositório local. Uma vez que as alterações foram feitas, podemos utilizar o comando `git restore <nome_arquivo>` para desfazê-las:



This screenshot shows the same VS Code interface after running the `git restore app.py` command. The editor window still shows the 'app.py' file, but the terminal panel now displays the output of the restore command, which is empty, indicating that the file has been successfully restored to its original state.

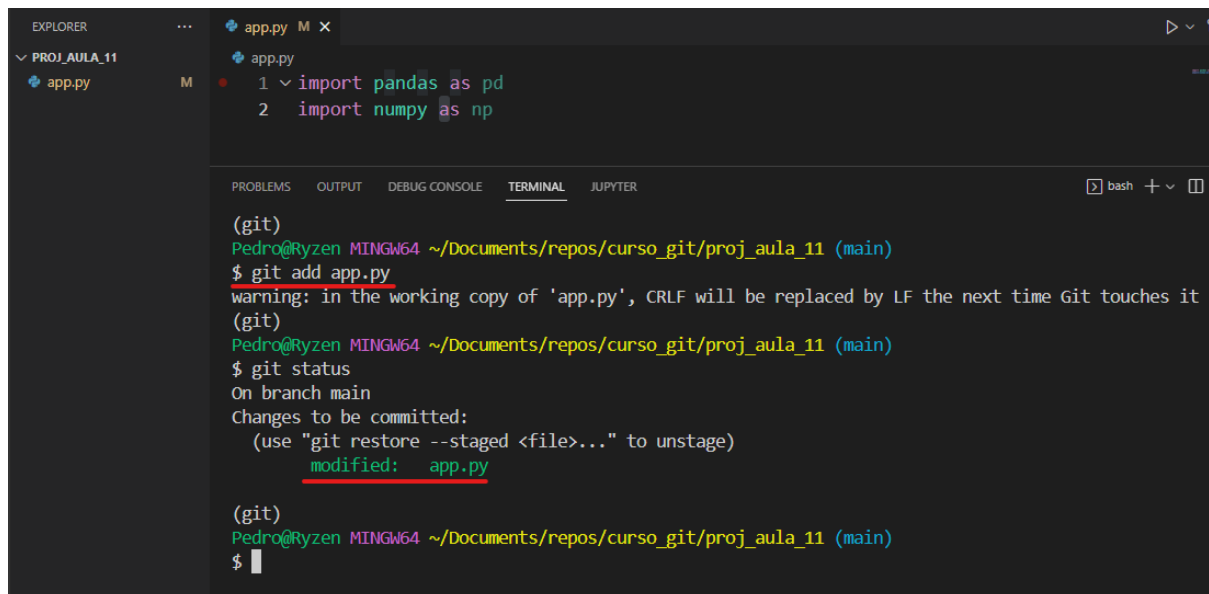
```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git restore app.py
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$
```

Observe que ao utilizarmos o comando `git restore app.py`, as alterações feitas foram removidas e o arquivo voltou ao seu estado original.

Desmarcando alterações marcadas como prontas

Caso tenhamos marcado as alterações de um arquivo como prontas por engano, utilizando o comando `git add`, podemos desmarcar essas alterações utilizando novamente o comando `git restore --staged <nome_arquivo>`.

Para exemplificar esse processo, vamos inserir novamente as bibliotecas `Pandas` e `Numpy` no arquivo `app.py` e marcar ele como pronto para ser commitado com o comando `git add`:

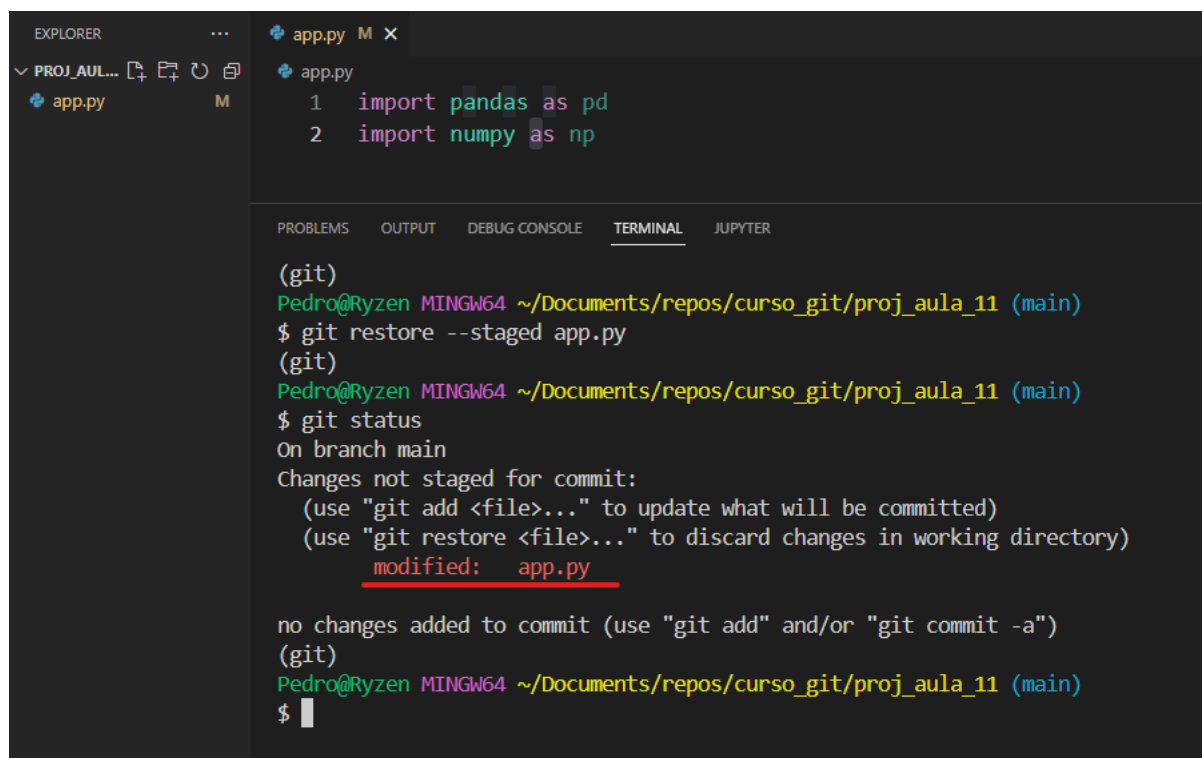


```
EXPLORER
...
PROJ_AULA_11
  app.py
  M
  1 import pandas as pd
  2 import numpy as np

TERMINAL
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git add app.py
warning: in the working copy of 'app.py', CRLF will be replaced by LF the next time Git touches it
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   app.py

(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$
```

Para desmarcar o arquivo, basta utilizarmos o comando `git restore --staged app.py`:



The screenshot shows a VS Code interface. The Explorer sidebar on the left shows a project named 'PROJ_AUL...' with a file 'app.py'. The main editor area shows the content of 'app.py':

```
1 import pandas as pd
2 import numpy as np
```

Below the editor is a terminal window with the following output:

```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git restore --staged app.py
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
       modified:   app.py

no changes added to commit (use "git add" and/or "git commit -a")
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git/proj_aula_11 (main)
$
```

Observe que agora o arquivo está dentro da **Área de Trabalho (Working Directory)** do Git e não mais na **Área de Stage**. Além disso, as alterações feitas ainda estão dentro do arquivo, o que nos permite fazer os ajustes necessários para marcarmos o arquivo como pronto novamente, ou podemos realizar novamente o comando `git restore` e remover todas as alterações feitas e que estão na **Área de Trabalho** do Git.

Exercícios

Link para o formulário de exercícios de fixação de conteúdo: [Exercícios](#)

Próxima Aula

Na próxima aula iremos verificar como visualizar e arrumar o histórico de commits feitos em um repositório local.

Fontes e Links Complementares

[Livro Pro Git - Capítulo 1.3 - O que é o Git](#)

[Livro Pro Git - Capítulo 2.2 - Salvando alterações no repositório](#)