

# Aula #25: Resolvendo Conflitos de Merge em Jupyter Notebooks

---

<a href="#">Revisão: O que é um conflito de Merge?</a>	1
<a href="#">Preparando o Projeto</a>	2
<a href="#">Resolvendo Conflitos de Merge em Arquivos Jupyter Notebook</a>	6
<a href="#">O Problema de arquivos Jupyter Notebook</a>	7
<a href="#">Pacote nbtime</a>	9
<a href="#">Próxima Aula</a>	14
<a href="#">Exercícios</a>	14
<a href="#">Fontes e Links Complementares</a>	14

---

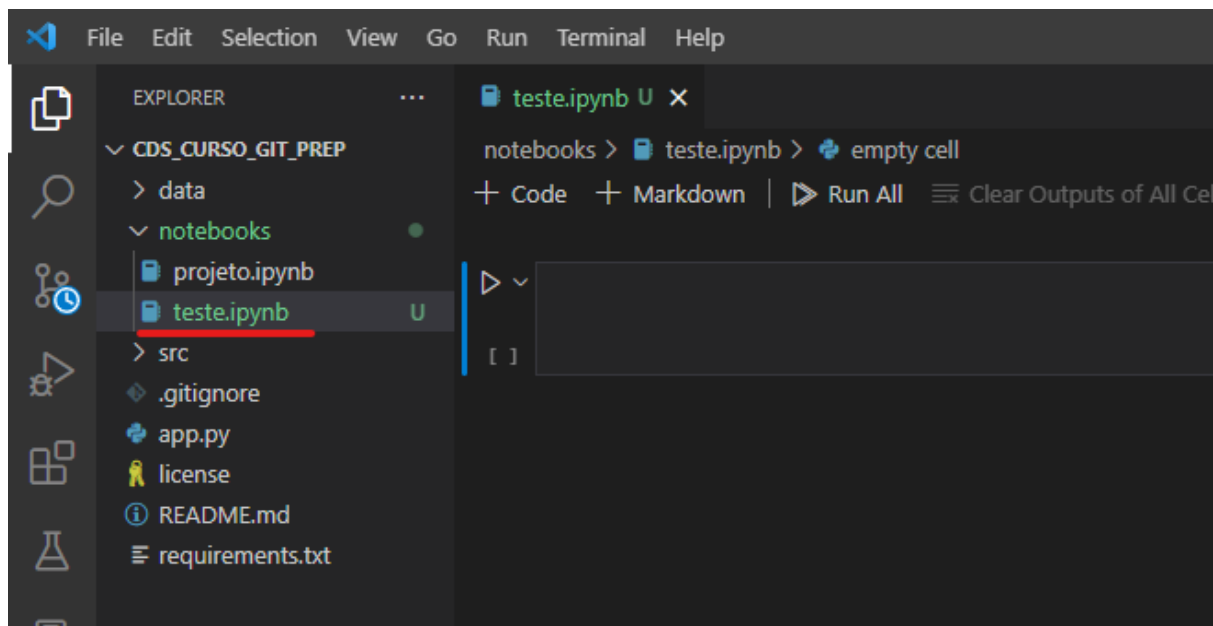
## Revisão: O que é um conflito de Merge?

Antes de iniciarmos os assuntos da aula em si, vamos revisar o que é e quando um conflito de merge acontece dentro do Git.

Conflitos de Merge ocorrem quando o Git não consegue fazer a união de dois commits de forma automática, independente de esses commit serem de branches diferentes ou de repositórios diferentes, ou ambos. Ou seja, toda vez que estamos tentando combinar commits feitos em um repositório, ou juntar os commits de uma branch na outra, estamos executando um merge, e quando o Git não consegue fazer isso de forma automática, é gerado um conflito.

# Preparando o Projeto

Antes de iniciar a explicação, vamos fazer alguns ajustes no nosso projeto. Com o projeto principal do curso aberto, crie um novo arquivo Jupyter notebook chamado `teste.ipynb` dentro da pasta chamada notebooks:



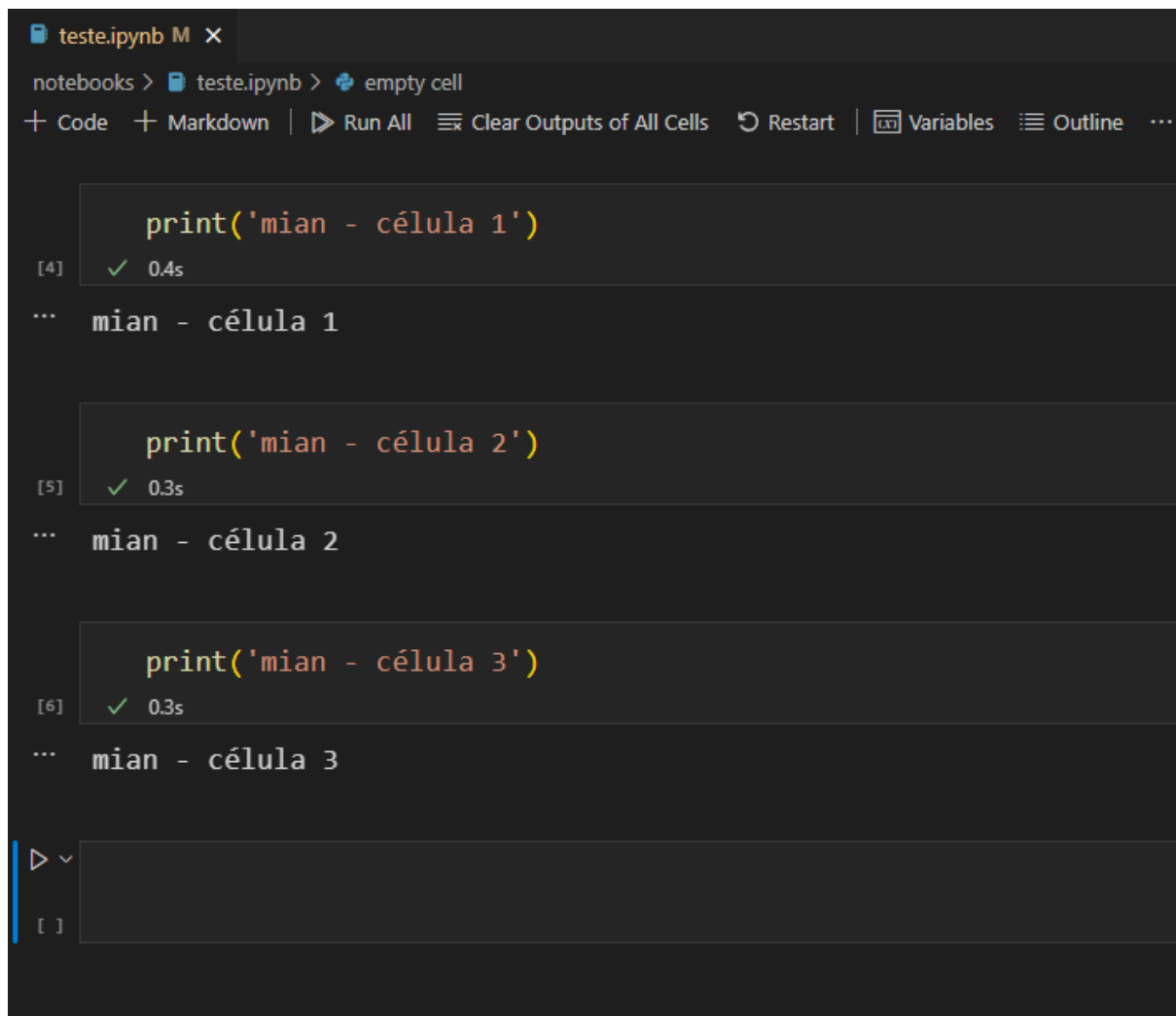
Com o arquivo criado, vamos commitar a alteração com a mensagem “Criado arquivo teste.ipynb”:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git add .
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git commit -m "Criado arquivo teste.ipynb"
[main 79ae92e] Criado arquivo teste.ipynb
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 notebooks/teste.ipynb
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$
```

Agora vamos criar uma branch chamada `dev` a partir da branch master:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git branch dev
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git branch -a
dev
* main
  remotes/origin/main
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$
```

Dentro da branch main, vamos criar três células no notebook teste:



```
teste.ipynb M X
notebooks > teste.ipynb > empty cell
+ Code + Markdown | Run All Clear Outputs of All Cells Restart | Variables Outline ...

[4] ✓ 0.4s
... mian - célula 1

[5] ✓ 0.3s
... mian - célula 2

[6] ✓ 0.3s
... mian - célula 3

[ ]
```

Vamos commitar essa alteração com a mensagem “main - Adicionado textos no notebook de teste”:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git add .
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git commit -m "main - Adicionado textos no notebook de teste"
[main 9d72eee] main - Adicionado textos no notebook de teste
1 file changed, 89 insertions(+)
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$
```

Agora vamos para a branch dev e fazer a mesma coisa: Criar três células com textos dentro:

```
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git checkout dev
Switched to branch 'dev'
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$
```

```
teste.ipynb M X
notebooks > teste.ipynb > empty cell
+ Code + Markdown | ▶ Run All ☰ Clear Outputs of All Cells ↺ Restart | [LX] Variables ☰ Out

print('dev - linha 1')
[2] ✓ 0.4s
... dev - linha 1

print('dev - linha 2')
[3] ✓ 0.3s
... dev - linha 2

print('dev - linha 3')
[4] ✓ 0.4s
... dev - linha 3

▶ ~
[ ]
```

E vamos commitar essas alterações com a mensagem “dev - criado celulas de texto”:

```
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$ git add .
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$ git commit -m "dev - criado celulas de texto"
[dev af7482a] dev - criado celulas de texto
1 file changed, 89 insertions(+)
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$
```

Verificando o nosso histórico de commits, teremos o seguinte resultado:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$ git log --oneline --all
af7482a (HEAD -> dev) dev - criado celulas de texto
9d72eee (main) main - Adicionado textos no notebook de teste
79ae92e Criado arquivo teste.ipynb
fce2e41 (origin/main) Ajustado conflito de merge
5ab9237 Ajustado app.py e reorganizado perguntas
06b7c96 ajustado funcao main
152484e Criado funcao da secao de resposta das perguntas do CEO
63a331a Adicionado secao de Descricao dos Dados
79f2c08 Ajustado arquivo .gitignore
75a81c7 first commit
d039b70 Respondido perguntas da primeira rodada de perguntas
429f3ca Criado arquivo de respostas das perguntas de negocio - answers.py
e532a7e Criado arquivo de extracao e ajustado arquivo principal
6f83216 Estrutura de pastas Ajustadas e removido arquivos de dados nao utilizado
6559189 Arquivo app.py criado e inicializado
53f930a Removido o arquivo app.py
35faa5d Adicionado bibliotecas pandas e numpy
178c9cb Adicionado arquivo app.py e rodado arquivo do projeto novamente
ca6e4b9 Adicionado licenca de uso
bf0844a Removido arquivo de licenca
06ab72d Adicionado licenca de uso
085ae1c Primeiro Commit
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$
```

Feito isso, estamos prontos para começar o assunto da aula de hoje

## Resolvendo Conflitos de Merge em Arquivos Jupyter Notebook

Imagine a seguinte situação: Terminamos o trabalho que estávamos fazendo na branch dev e agora queremos juntar tudo que fizemos dentro da branch dev para a branch main.

Mas antes, porque iremos ter um conflito de merge no nosso repositório? Porque quando criamos a branch `dev`, ela foi criada a partir do commit `79ae92e`.

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$ git log --oneline --all
af7482a (HEAD -> dev) dev - criado celulas de texto
9d72eee (main) main - Adicionado textos no notebook de teste
79ae92e Criado arquivo teste.ipynb
fce2e41 (origin/main) Ajustado conflito de merge
5ab9237 Ajustado app.py e reorganizado perguntas
```

E depois disso, tivemos um commit feito tanto na branch main quanto na branch dev. Além disso, esse commit movimenta o mesmo arquivo e as mesmas linhas. E por conta disso, haverá um conflito.

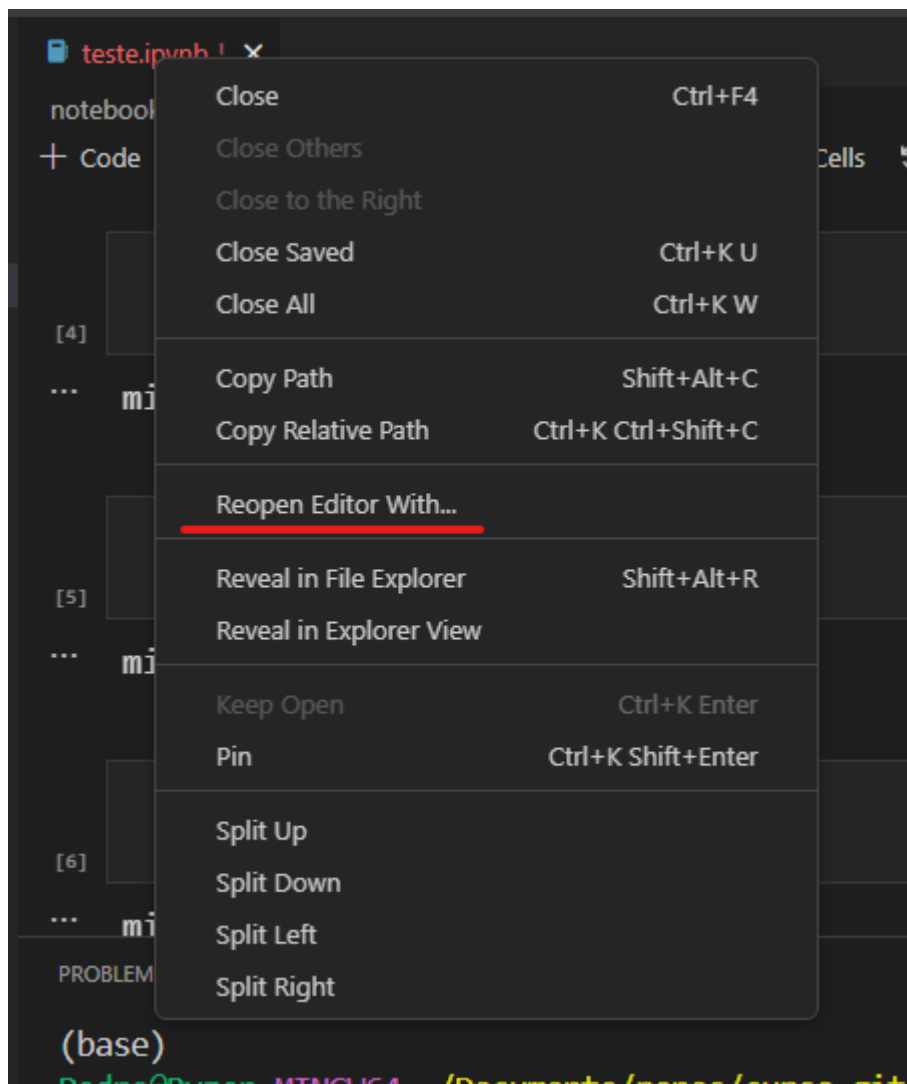
Dito isso, vamos fazer o merge dos commits dentro da main:

```
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (dev)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git merge dev
Auto-merging notebooks/teste.ipynb
CONFLICT (content): Merge conflict in notebooks/teste.ipynb
Automatic merge failed; fix conflicts and then commit the result.
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$
```

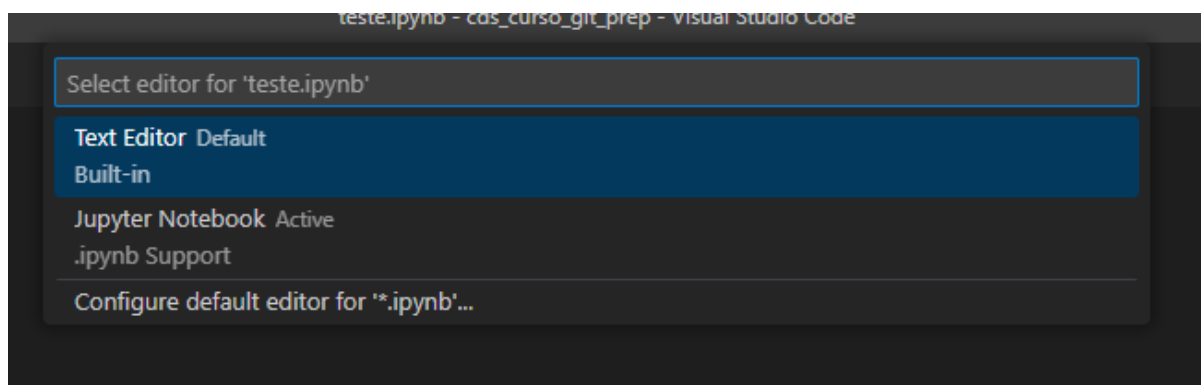
Feito isso, temos um conflito de merge.

## O Problema de arquivos Jupyter Notebook

Essencialmente, um arquivo Jupyter Notebook é um arquivo **JSON** que quando é interpretado, é criado as células do Jupyter Notebook. Para demonstrar isso, basta clicar com o botão direito do mouse sobre a aba do arquivo teste.ipynb e selecione a opção **"Reopen Editor With"**:



E selecione a opção Text Editor na janela aberta



Com isso feito, será aberto o “código fonte” do arquivo Jupyter Notebook:



```
1 {
2   "cells": [
3     {
4       "cell_type": "code",
5       <<<<<<< HEAD (Current Change)
6       =====
7       "execution_count": 2,
8       "metadata": {},
9       "outputs": [
10        {
11          "name": "stdout",
12          "output_type": "stream",
13          "text": [
14            "dev - linha 1\n"
15          ]
16        }
17      ],
18      "source": [
19        "print('dev - linha 1')
20      ]
21    },
22    {
23      "cell_type": "code",
```

Por conta disso, quando há conflitos em arquivos Jupyter Notebooks, não conseguimos ver as marcações de diferença entre branches.

## Pacote nbdime

Em vista disso, foi criado um pacote chamado [nbdime](#). Ele irá nos ajudar a visualizar essas diferenças dentro dos arquivos Jupyter Notebooks, de forma que texto. Vamos começar a utilizar esse pacote. Primeiro, temos que instalá-lo. Dentro do Anaconda, esse pacote não existe, por conta disso, teremos que instalar utilizando o pip, ao invés do conda.

Confirme se o seu terminal está com o ambiente do curso ativo.

```
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds curso_git_prep (main|MERGING)
$
```

Caso não esteja, utilize o comando para ativar conda `activate C:\\Users\\Pedro\\anaconda3\\envs\\git` trocando o `git` pelo nome da sua variável de ambiente:

```
(base)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$ conda activate C:\\Users\\Pedro\\anaconda3\\envs\\git
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$
```

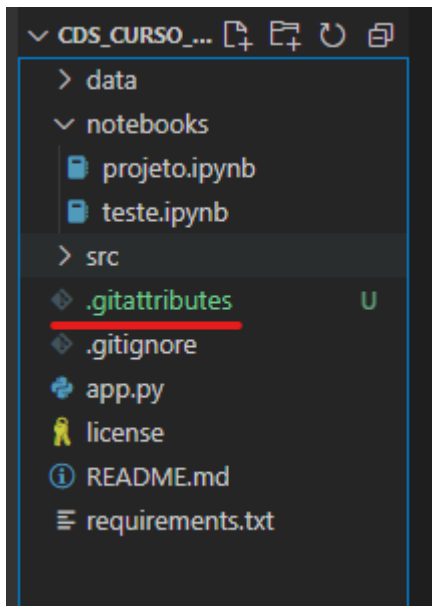
Com o ambiente ativo, utilize o comando `pip install -U nbdtimer`:

```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$ pip install -U nbdtimer
```

E vamos agora usar o comando `git-nbmergedriver config --enable` para configurar o git para que, todo conflito que houver em arquivo `.ipynb`, use o driver de verificação de diferença do pacote que acabamos de instalar

```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git-nbmergedriver config --enable
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$
```

Isso irá criar um arquivo chamado `.gitattributes` dentro do diretório principal do projeto:



Com o pacote instalado e configurado, precisamos abortar a tentativa de merge para tentar novamente, mas agora com o pacote instalado. Para abortar um merge, vamos usar o comando `git merge --abort`:

```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$ git merge --abort
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$
```

Feito isso, vamos tentar novamente:

```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$ git merge dev
[W strategies:468] Recording unresolved conflicts in /metadata/nbdime-conflicts.
[W nbmergeapp:55] Conflicts occurred during merge operation.
[E __init__:130] Notebook JSON is invalid: data.cells[{{data__cells_x}}] must be valid exactly by one definition (0 matches found)

Failed validating <unset> in notebook['data']['cells']:

On instance:
<unset>
[I nbmergeapp:68] Merge result written to .merge_file_kgxSvW
Auto-merging notebooks/teste.ipynb
CONFLICT (content): Merge conflict in notebooks/teste.ipynb
Automatic merge failed; fix conflicts and then commit the result.
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$
```

Observe que agora temos mensagens diferentes e, se abrirmos o arquivo teste.ipynb, iremos ver que as marcações do conflito estarão dentro do arquivo Jupyter Notebook:



```
teste.ipynb ! X
notebooks > teste.ipynb > M***&lt;&lt;&lt;&lt;&lt;&lt; local**
+ Code + Markdown | ▶ Run All ≡ Clear Outputs of All Cells | ≡ Outline ...

<<<<<< local

[4] print('mian - célula 1')
... mian - célula 1

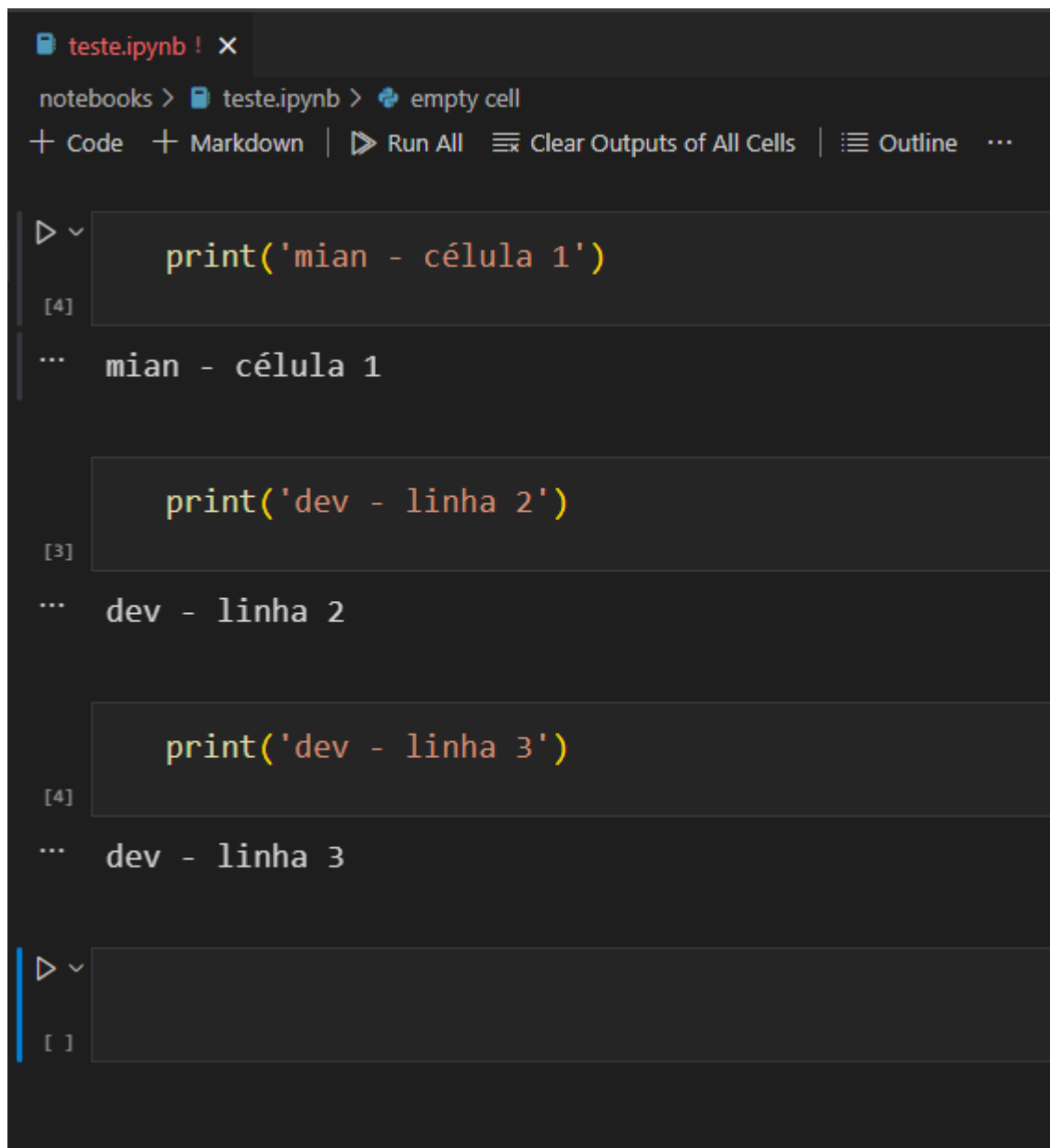
[5] print('mian - célula 2')
... mian - célula 2

[6] print('mian - célula 3')
... mian - célula 3

=====

[2] print('dev - linha 1')
... dev - linha 1
```

Dessa forma, vamos agora fazer os ajustes, commitar os ajustes e arrumar o merge:



```
teste.ipynb ! X
notebooks > teste.ipynb > empty cell
+ Code + Markdown | ▶ Run All ☰ Clear Outputs of All Cells | ☰ Outline ...

▶ ▼
    print('mian - célula 1')
[4]
... mian - célula 1

    print('dev - linha 2')
[3]
... dev - linha 2

    print('dev - linha 3')
[4]
... dev - linha 3

▶ ▼
[ ]
```

E commitar o ajuste com a seguinte mensagem “Ajustado driver de arquivos notebooks e ajustado conflito de merge”:

```
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$ git add .
warning: in the working copy of '.gitattributes', CRLF will be replaced by LF the next time Git touches it
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main|MERGING)
$ git commit -m "Ajustado driver de arquivos notebooks e ajustado conflito de merge"
[main ec0ea81] Ajustado driver de arquivos notebooks e ajustado conflito de merge
(git)
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/cds_curso_git_prep (main)
$
```

E é dessa forma que resolvemos conflitos de merge com arquivos Notebooks

## Próxima Aula

Na próxima aula vamos iniciar o novo ciclo do curso. E na primeira aula, vamos terminar de ajustar o projeto final do curso.

## Exercícios

Exercícios da Aula: [Link](#)

Sessão Cross-Validation: [Link](#)

## Fontes e Links Complementares

[Documentação - git merge](#)

[Artigo - Jupyter Notebook - Verificando Diferenças e Merges em Arquivos Jupyter Notebook](#)

[Documentação - nbdime](#)

[Artigo - Medium - Como fazer merges em Arquivos Jupyter Notebooks sem dor de cabeça](#)