

```
1 def Python(Fundamentos e
2 análise de dados):
3
4     print('Projeto guiado 2')
5
6
7
8
9     # Desenvolvimento Guiado por Testes
10
11
12
13
14
```

Vamos nos conhecer!

Escaneie o QR Code



Ou acesse o site e digite o código

<https://www.menti.com/>

7127 4456

Sumário

Planejamento	Slide 4
Testes de unidade vs TDD	Slide 5
Etapas do TDD	Slide 7
Padrão Triplo A	Slide 8
Bora pro código!	Slide 9
Atividade	Slide 10
Instruções para a entrega	Slide 11
Conclusão	Slide 12
Referências	Slide 13
Agradecimento e contato	Slide 14

Planejamento

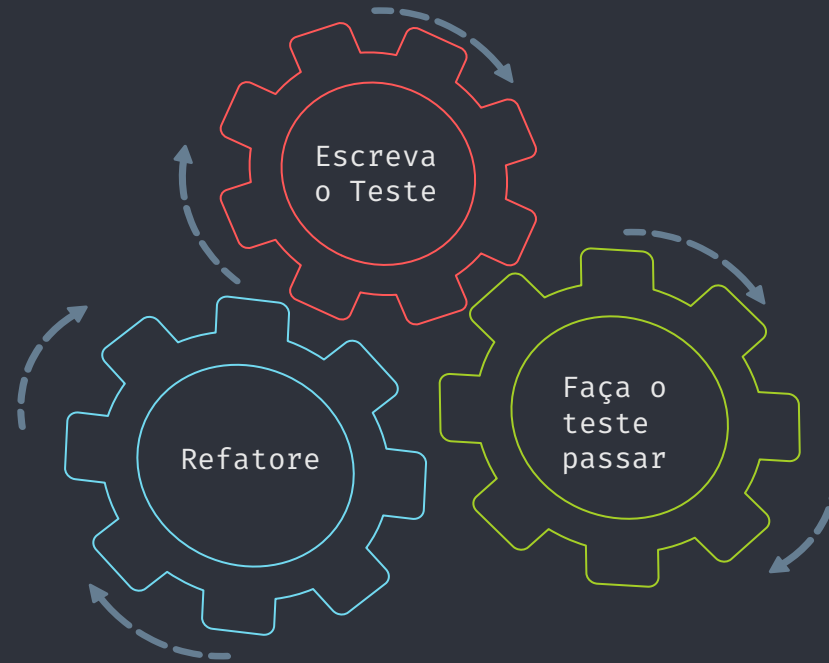
```
1  
2  
3  
4  
5      # Criar um sistema de gerenciamento de biblioteca  
6      usando TDD com as funcionalidades:
```

- ```
7 * Adicionar livros
8 * Listar livros
9 * Emprestar livros
10
11
12
13
14
```

```
1
2
3
4 print('Testes de unidade' == 'TDD')
5
6
7
8
```

```
9 >>> false
10
11
12
13
14
```

# Etapas do TDD



# O padrão Triplo A

## Arrange (Configuração)

- Configuração de tudo o que é necessário para que o nosso teste possa rodar (instâncias, mocks, fakes, stubs, spies etc)

## Act (Execução)

- Onde chamamos alguma função ou método que queremos por a prova. É comum chamar o objeto de teste de sut (System Under Test)

## Assert (Validação)

- É onde verificamos se a operação realizada na etapa anterior (Act) surtiu o resultado esperado. Assim sabemos se o teste passa ou falha.

# Bibliotecas de teste em Python

## Unittest

- Biblioteca nativa do Python
- Inicialmente chamado PyUnit
- Inspirado no JUnit

## PyTest

- Biblioteca mais utilizada
- Open source

## Nose2

- Unittest com plugins
- Substituto do Nose

## Testify

- Pensado para substituir unittest e o Nose



# Bora pro código!

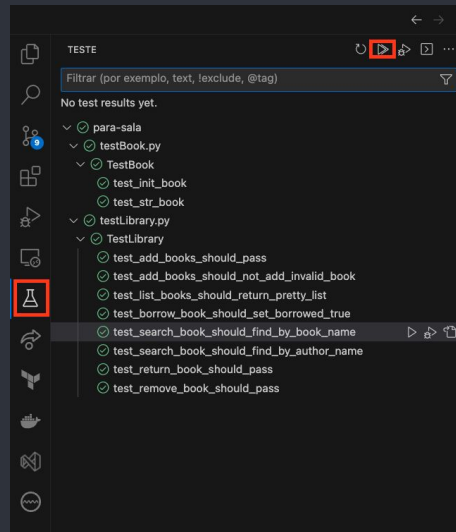
```
import webbrowser
```

```
webbrowser.open('https://github.com/reprograma/on26-python-s08-
projeto-guiado-II/tree/main')
```

# Executando os testes

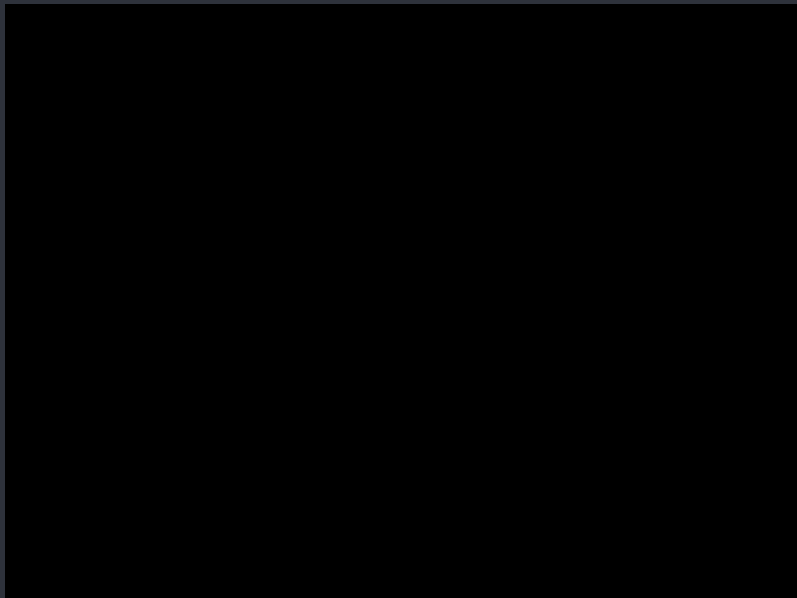
```
Pela linha de comando
> python3 -m unittest filename.py
```

```
Ou pelo teste explorer do VS Code
Python testing in Visual Studio Code
```



# Usando Test Explorer

# Vídeo demonstrativo de como configurar o Test Explorer



# Atividade até quarta-feira (27/9)

## 01 Criar método **exibir\_livros**

CrITÉRIOS de aceitação:

- - O método deve ter ao menos 1 (um) teste associado
- - O método deve não tem parâmetro além do self (contém apenas o self obrigatório)
- - O método deve exibir todos os livros que foram adicionados  
(dica retorne a lista e valide se contém todos os livros adicionados)

## 02 Criar método **emprestar\_livro**

CrITÉRIOS de aceitação:

- O método deve ter ao menos 1 (um) teste associado
- O método deve receber como parâmetro o nome do livro a ser emprestado
- O método deve marcar o valor de esta\_emprestado como `True`

# Atividade (opcional)

## 03 Criar método `remover_livro`

Critérios de aceitação:

- O método deve ter ao menos 1 (um) teste associado
- O método deve receber como parâmetro o nome do livro e remover da propriedade Books
- O método deve remover apenas 1 (um) livro por vez
- Caso o livro não seja encontrado o método não deve dar erro ou exceções
- Apenas livros não emprestados podem ser removidos (opcional)

## 04 Criar método `buscar_livro`

Critérios de aceitação:

- O método deve ter ao menos 1 (um) teste associado
- O método deve receber como parâmetro o nome do livro a ser buscado e retornar o nome do livro, autor e informação se livro está disponível ou emprestado
- Caso o livro não seja encontrado o método deve retornar a mensagem "Livro não encontrado"

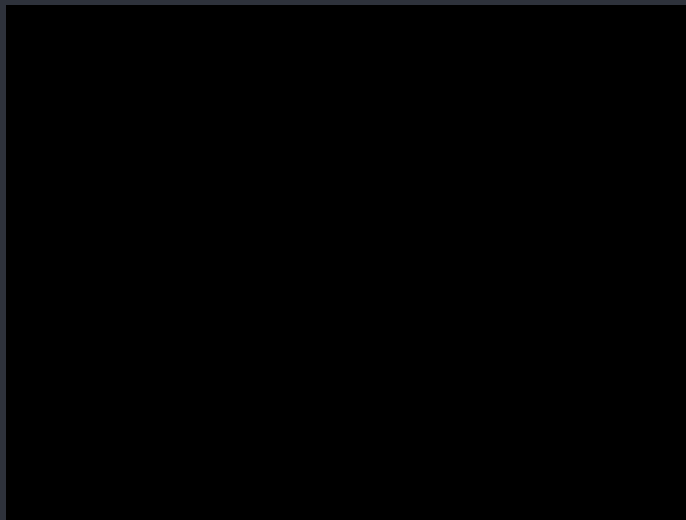
## 05 Criar método `devolver_livro`

Critérios de aceitação:

- O método deve ter ao menos 1 (um) teste associado
- O método deve receber como parâmetro o nome do livro a ser devolvido e mudar o status do livro para não emprestado na propriedade Books
- Caso o livro não seja encontrado o método deve apenas exibir a mensagem "Livro não encontrado"

# Instruções para a entrega

```
Vídeo demonstrativo de como abrir um Pull Request ou confira
as instruções escritas aqui
```



# Conclusão

- \* Os testes escritos com unittest devem ter o prefixo `test`
- \* Os testes podem ser executados pela linha de comando ou pelo `Test Explorer`
- \* As pastas devem ter o arquivo `__init__.py` para que os arquivos dentro da pasta sejam reconhecidos como `pacotes`
- \* Para auxiliar na estruturação do teste pode-se utilizar o padrão Triplo A (Arrange, Act, Assert)

# Referências

## # Vídeos:

[Quero ser Dev - O que é TDD?](#)

## # Livros:

[TDD Desenvolvimento Guiado Por Testes - Kent Beck \(pt-br\)](#)

[Test-Driven Development By Example - Kent Beck \(inglês\)](#)

[The Hitchhiker's Guide to Python!\(inglês\)](#)

## # Artigos:

[Happy & Unhappy Paths: Why You Need to Test Both](#)

[Python Iluminado - Erros, Exceções e Testes](#)

[7 Popular Unit Test Naming Conventions](#)

[Arrange-Act-Assert: A Pattern For Writing Good Tests](#)

[O Padrão Triple A \(Arrange, Act, Assert\)](#)

[Mocking, stubs, & spies in your unit tests](#)

[Mocks Aren't Stubs](#)

[SUT](#)

[Testando seu código](#)

[5 Melhores frameworks para testes com Python](#)

[Python Developers Survey 2019 Results](#)

## # Podcasts:

[QuebraDev #20 - \[Tech\] Python com Nana Raythz](#)

[Estratégias de Testes e Dev Leaders - Hipsters Ponto Tech #367](#)

[Python - Hipsters #122](#)



# Referências (continuação)

# Documentação:

[Nose2](#)

[Testify](#)

[PyTest](#)

[Unittest](#)

[Built-in Functions](#)

[Built-in Exceptions](#)

[Python testing in Visual Studio Code](#)

[Python docstring](#)

# Tutoriais:

[Real Python Tutorials](#)

[PyTutor \(para fazer teste de mesa\)](#)

[Mocking print\(\) in Unit Tests](#)

# Repositórios com exemplos de testes

[Py-sorting](#)

[Awesome Python Testing](#)

1 **Obrigada!**

2 #Tem alguma dúvida ou feedback?



4 mayaracsferreira@gmail.com



6 <https://www.linkedin.com/in/mayaracsf/>



8 <https://github.com/mayaracsferreira>



10 <https://reprograma.slack.com/team/U056MB4U5T2>

11 CREDITS: This presentation template was  
12 created by **Slidesgo**, including icons by  
13 **Flaticon**, and infographics & images by **Freepik**  
14