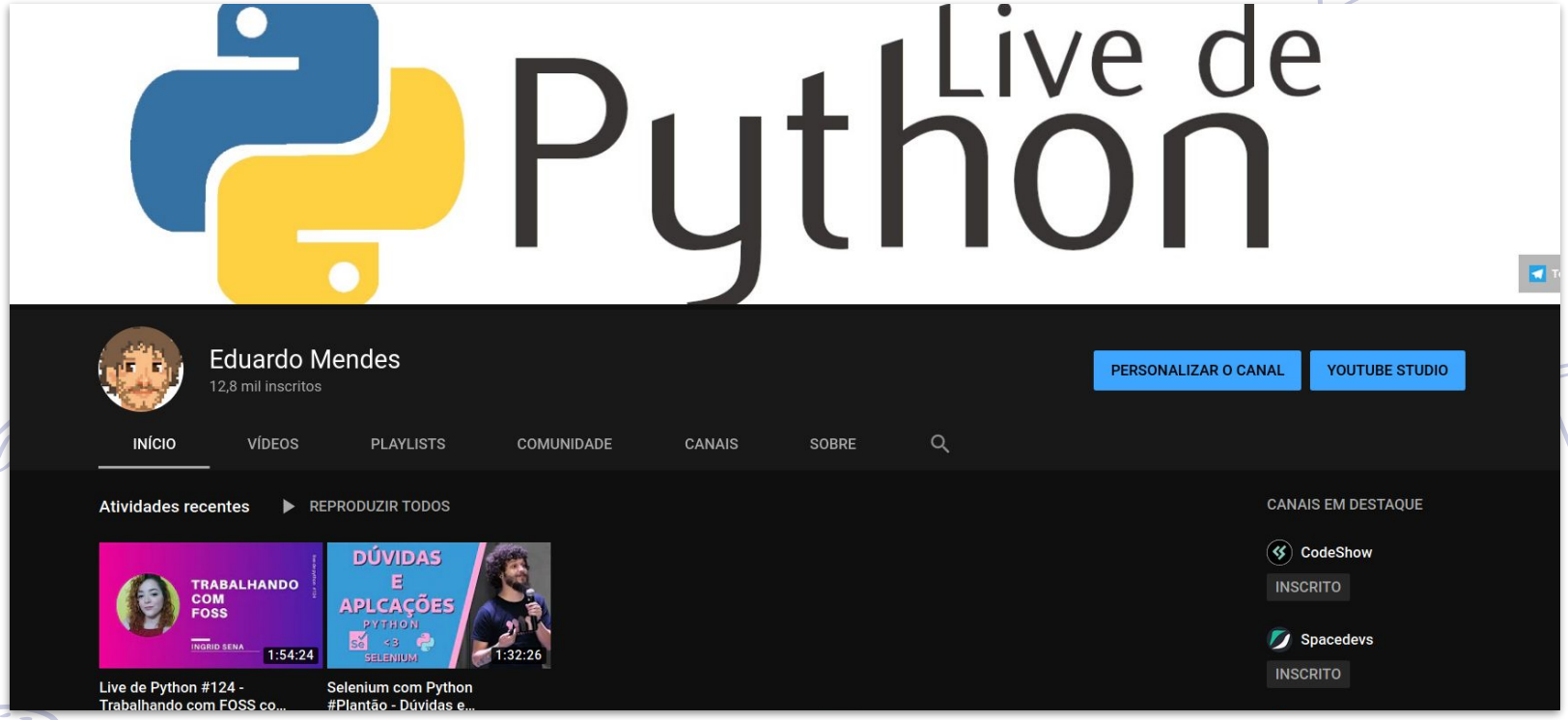
# Saindo do zero em testes


Antes de tudo

# Olar bebês!

**Sou Eduardo Mendes (@dunossauro)**

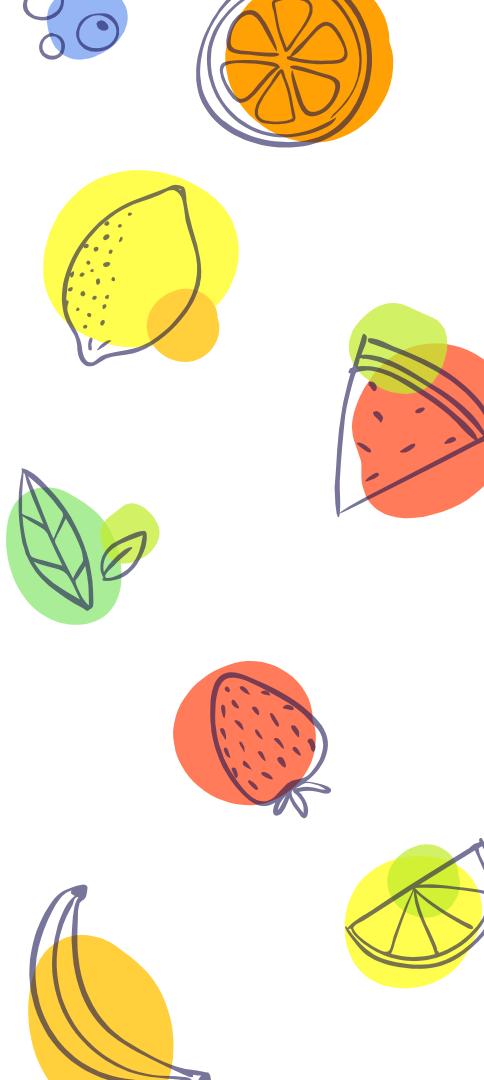
Apto fazedor de lambdas, pythonista, apaixonado por software **livre** e ciência.

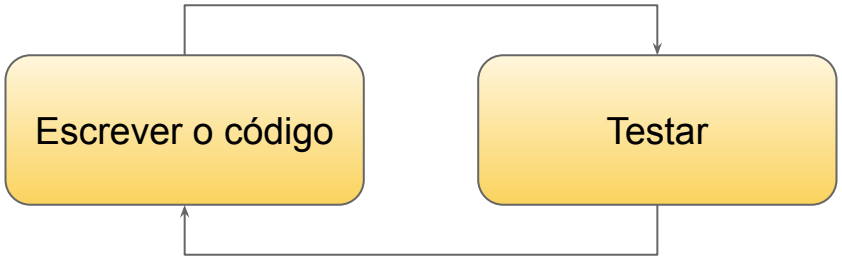


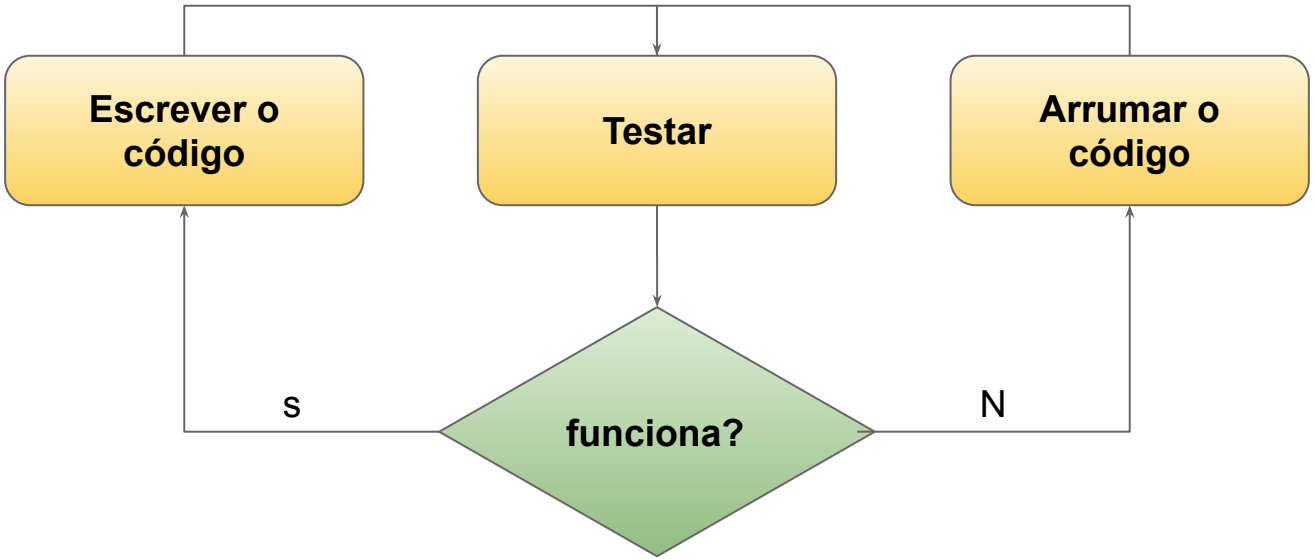

**youtube.com/eduardomendes**

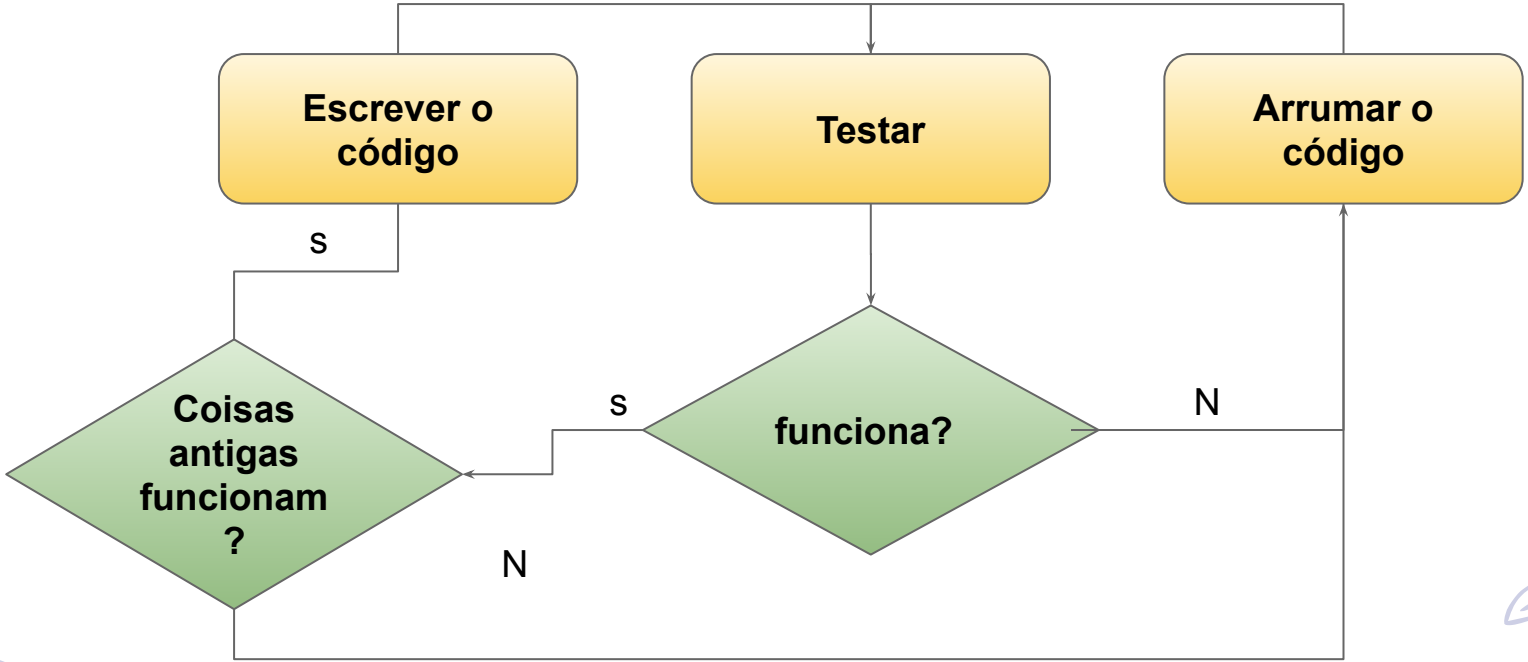
# Roteiro

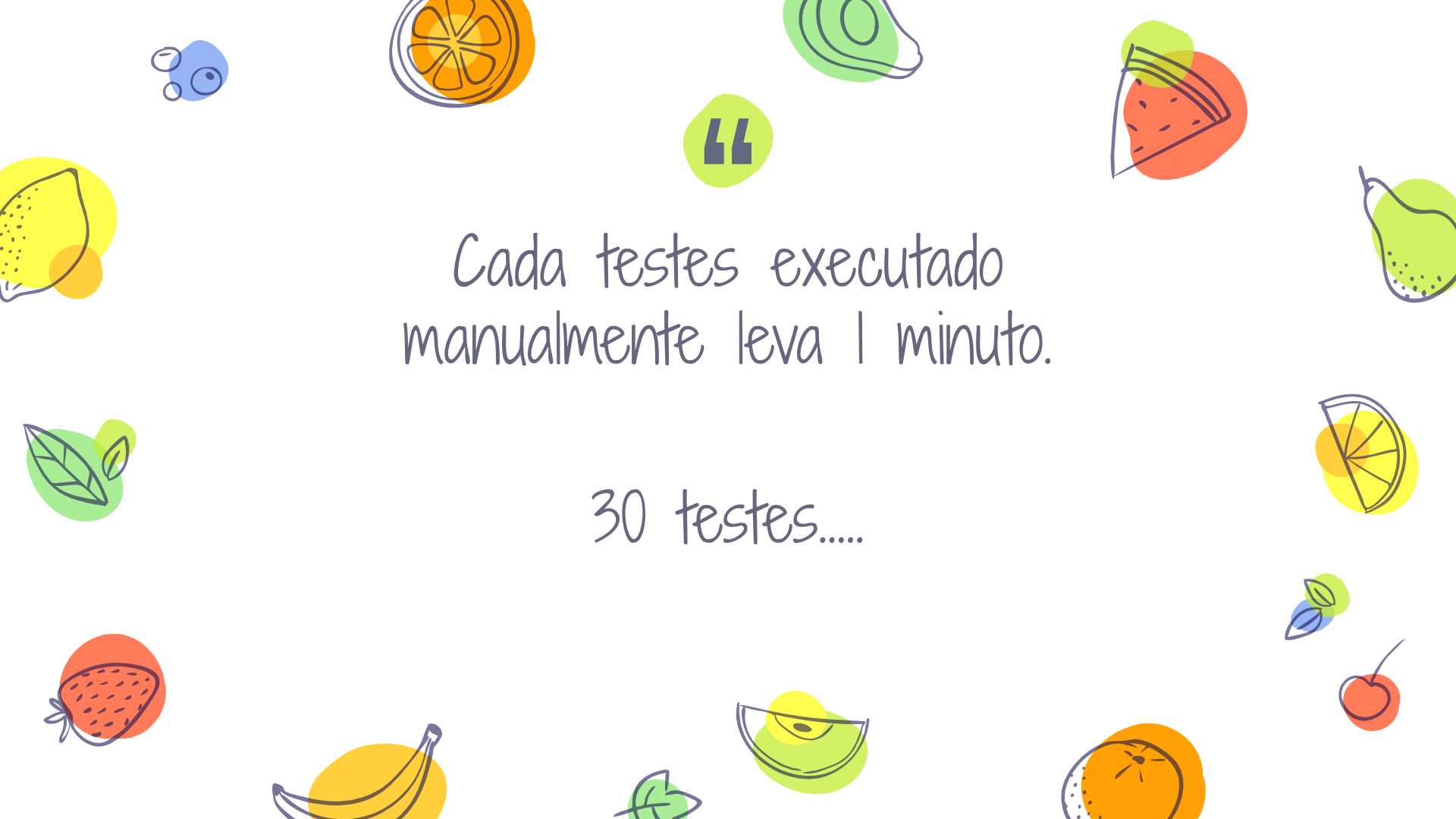
- Testes, para que testes?
- Tipos de teste
- Bibliotecas de teste
- Testes de unidade

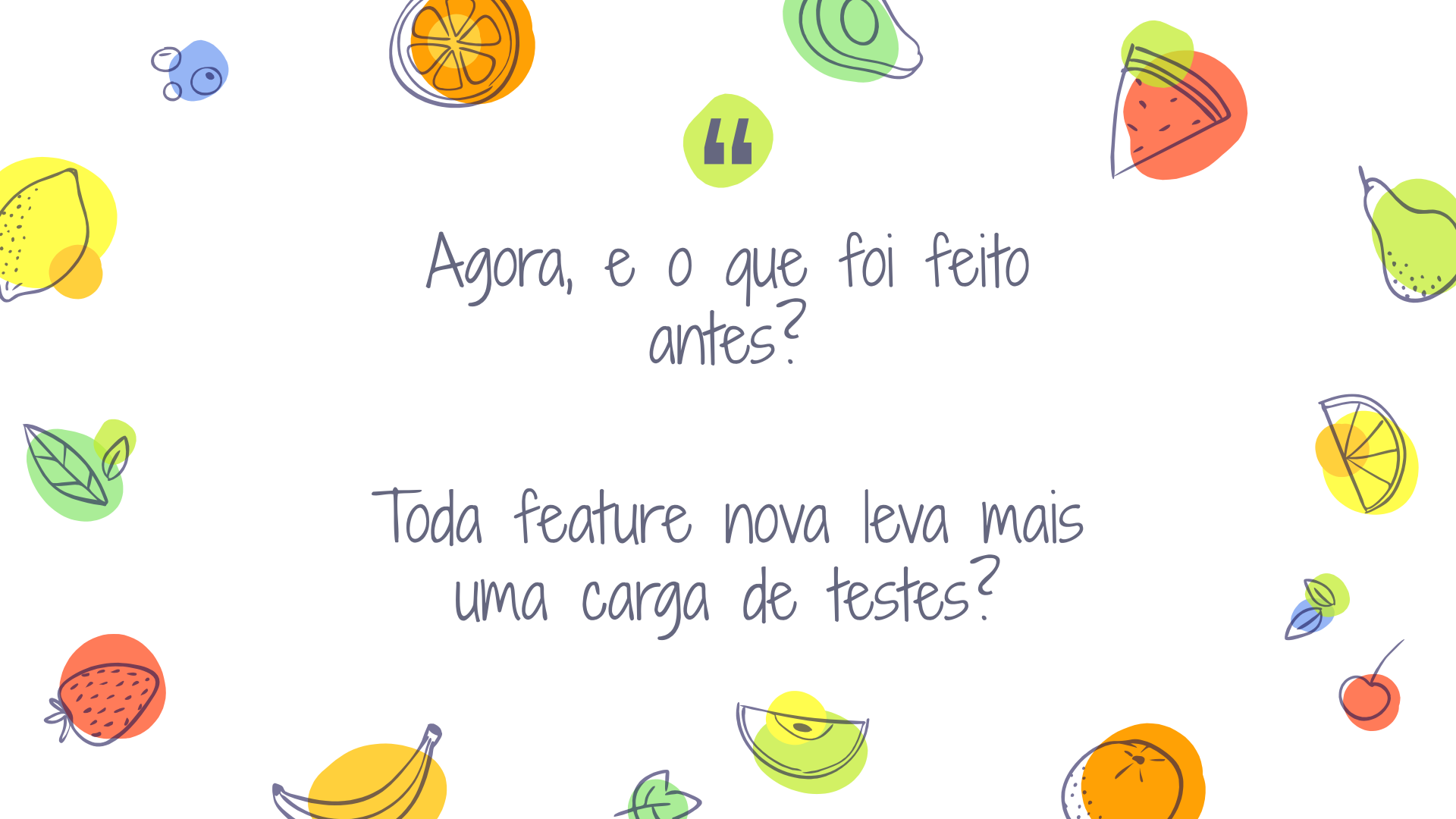



Escrever o código
Testar

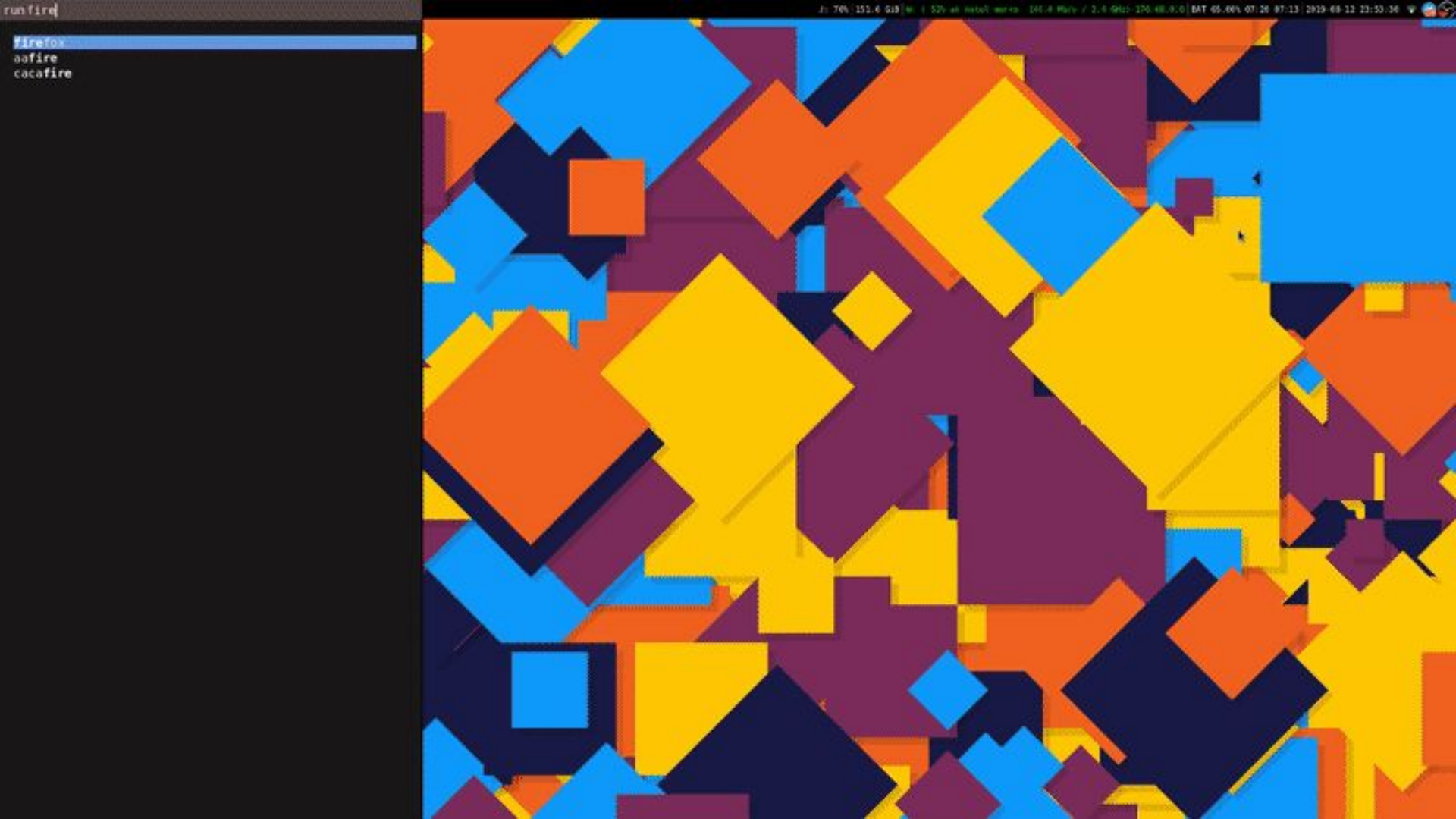

Escrever o código
Testar
Arrumar o código
funciona?
s
N


Escrever o código
Testar
Arrumar o código
s
Coisas antigas funcionam ?
s
funciona?
N
N


“
Cada testes executado manualmente leva 1 minuto.
30 testes.....


“
Agora, e o que foi feito antes?
Toda feature nova leva mais uma carga de testes?

# Então vamos aprender testes


runfire
firefox
aafire
cacafire

# Tipos de teste

- Testes unitários
- Testes de Integração
- Testes de sistema
- Testes de regressão
- Testes de regressão visual
- Testes de caixa preta/branca/cinza
- Testes de mutação

# Tipos de teste


- Testes unitários
- Testes de Integração
- Testes de sistema
- Testes de regressão
- Testes de regressão visual
- Testes de caixa preta/branca/cinza
- Testes de mutação

# Bibliotecas de teste

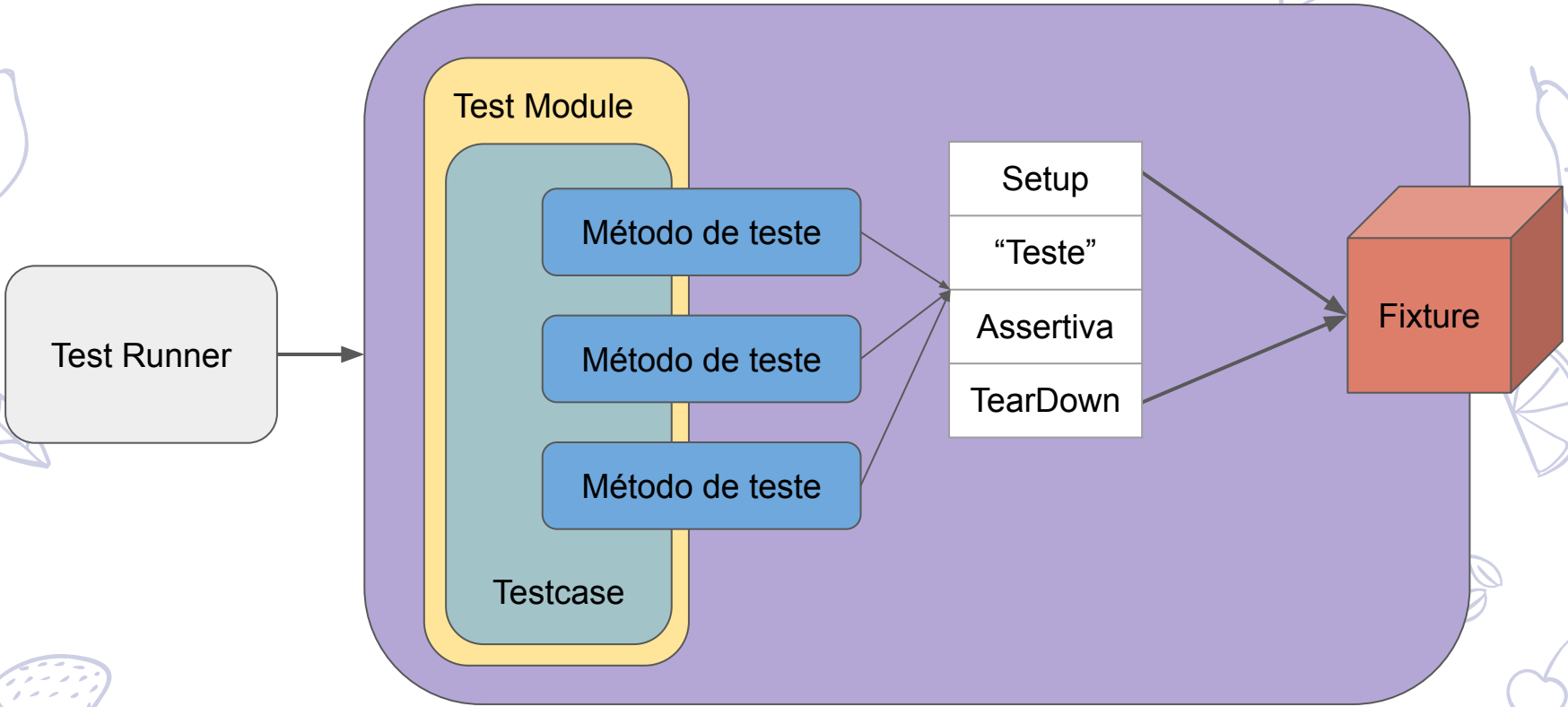
# Bibliotecas

- Unittest
  - Nativo
- Pytest
  - https://docs.pytest.org/
- Mamba
  - https://nestorsalceda.com/mamba
- Behave
  - https://github.com/behave/behave

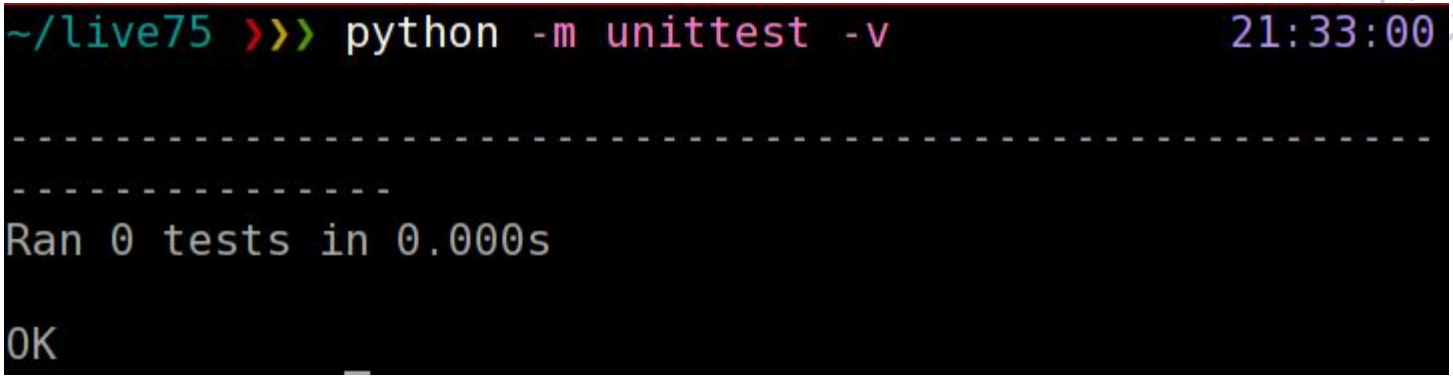
# Bibliotecas

- Unittest
- Pytest
- Mamba
- Behave

# Um belo dia com X-Unit


Test Runner
Test Module
Testcase
Método de teste
Método de teste
Método de teste
Setup
“Teste”
Assertiva
TearDown
Fixture

Test Runner

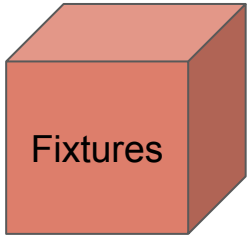


Testcase

```
class TestCase(unittest.TestCase):
    def setUp(self):
        print('No setUp')

    def tearDown(self):
        print('No tearDown')

    def test_01(self):
        print('No test_01')
        self.assertEqual(1, 1, msg='1 != 1')
```

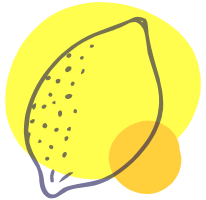


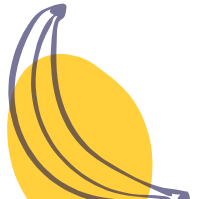

Método de teste

Assertiva

# Nosso primeiro desafio

```
def soma(x, y):
    """Retorna uma soma de 2 números."""
    return x + y
```

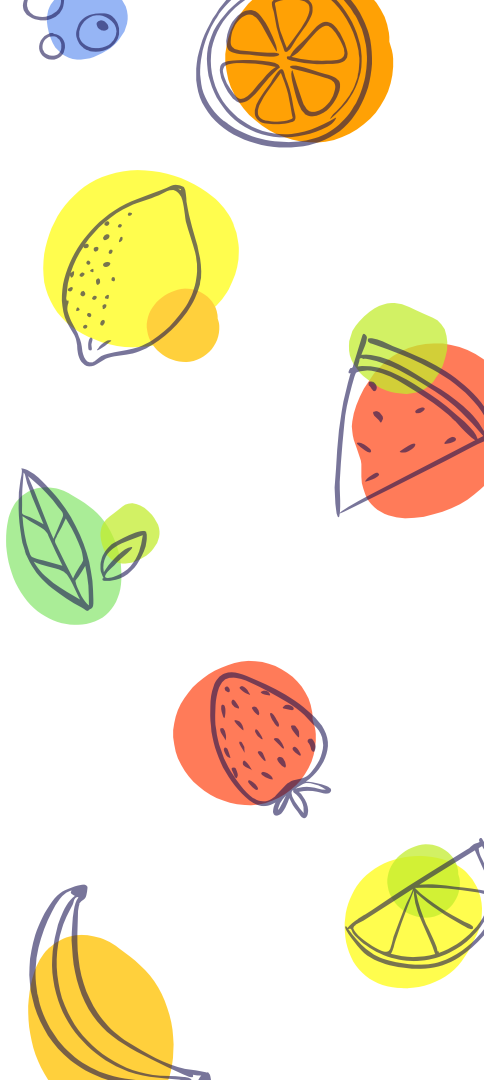




# SUT (System Under Test)

Geralmente, “o que está sendo testado” na bibliografia aparece como SUT, mas pode aparecer com nomes diferentes, como:

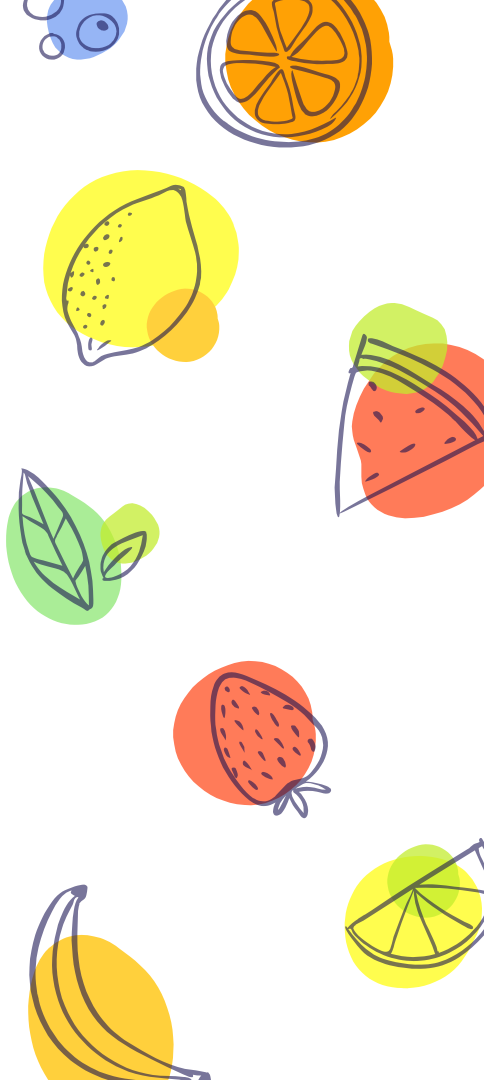
- SUT: Sistema em teste (A coisa em si)
- DOC: Componente de quem o SUT depende



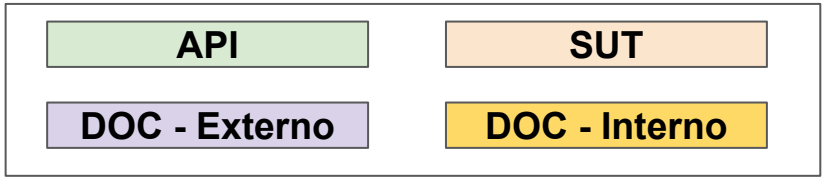
# Nosso segundo desafio

```
def expr(x, y, z):
    """
    Resolve a expressão.

    (x + y) - z
    """
    return subtracao(soma(x, y), z)
```

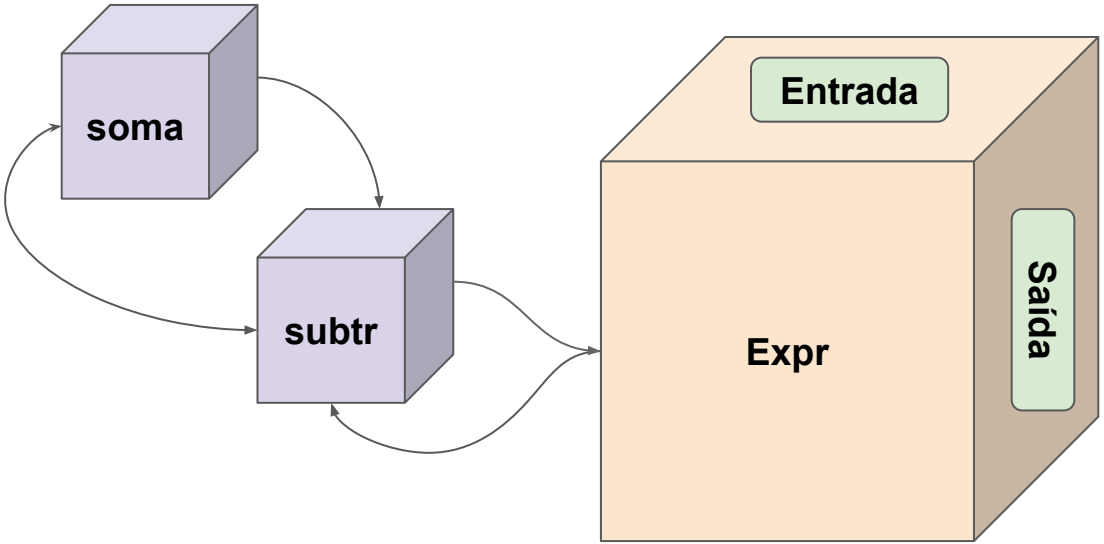


# Acoplamento




Não é tão complicado entender o sentido de acoplamento.

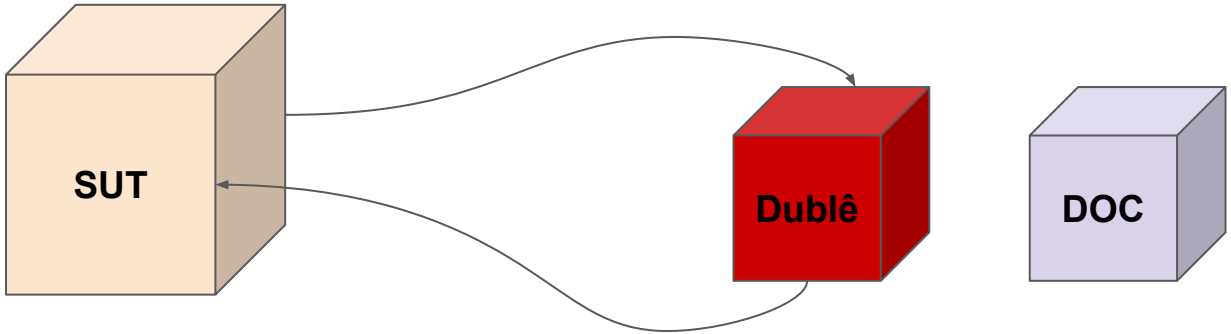
O quanto uma coisa depende de outra.



Dublês de teste

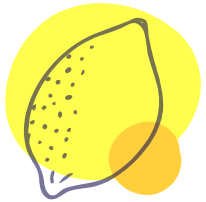
# Dublês

Dublês tornam seus testes mais **determinísticos**, pois eles vão remover as dependências do jogo.

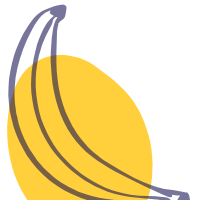


# Dublês

```
def teste_expressao_com_1_1_1_deve_retonar_1(self):
    x = 1
    y = 1
    z = 1
    esperado = 1
    with patch('code.subtracao', return_value=1):
        self.assertEqual(expr(x, y, z), esperado)
```

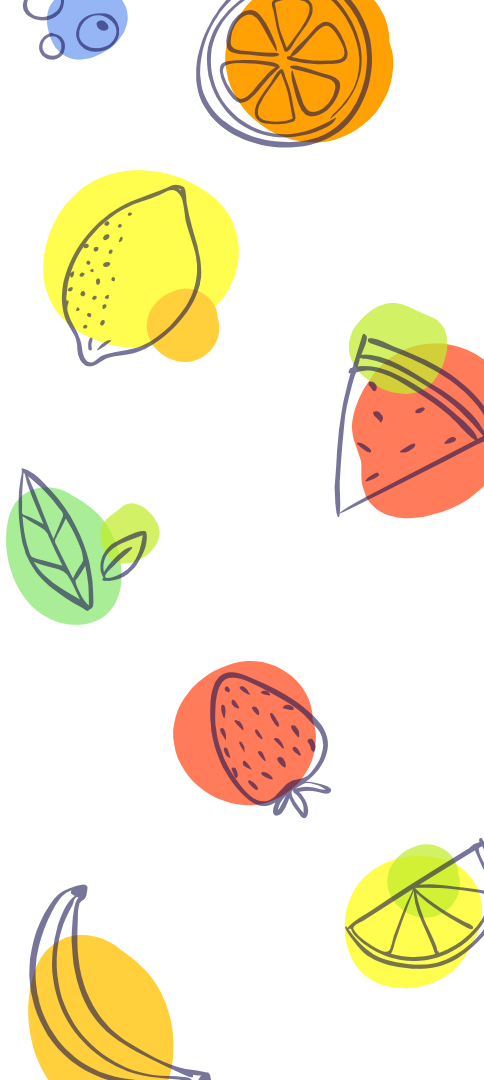





# Aprenda comigo

- Live de python #75
- Live de python #76
- Live de python #79
- Live de python #80
- Live de python #81
- Live de python #83



# XOXO

**Perguntas?**

gist.github.com/dunossauro/ff9c2d346ba05ee66887346ccd36c20a

Você pode me achar em

@dunossauro

mendesxeduardo@gmail.com

**youtube.com/eduardomendes**