

# TDD - Test-Driven Development



# JESSYKA LAGE

## DEVELOPER | SÃO PAULO



Atua como desenvolvedora desde 2012, formada em Ciência da Computação e pós-graduanda em Inteligência Artificial. Atualmente consultora de software na ThoughtWorks.

Twitter: @jessykalage

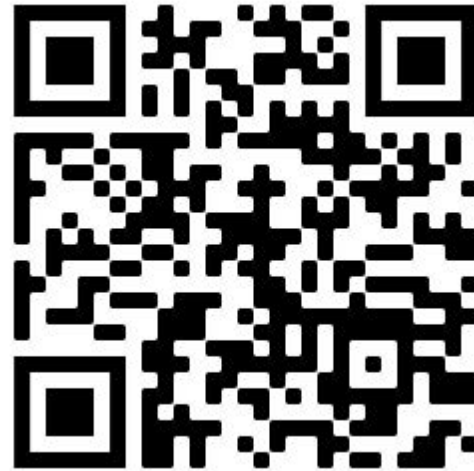
Instagram: @jessykalage

LinkedIn: Jessyka Lage

Github: @jessyka

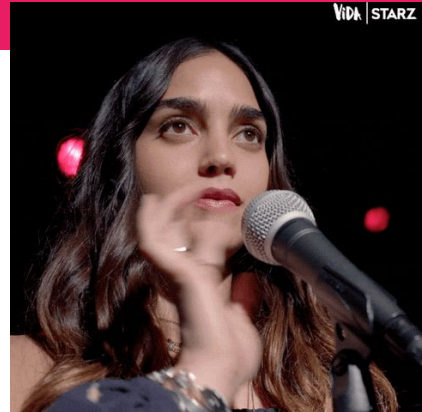
**Dê seu Feedback! <3**

---



<https://bit.ly/2ATDFAA>

# Alô, alô... testando!



# O que é teste?

— — —

- "Qualquer meio para verificar ou testar a **qualidade** ou a **veracidade** de algo; prova, **experimento**, **verificação**"
- "Prova para verificação da **eficiência** ou do **bom funcionamento** de equipamento, organização, material, etc"
- "Todo mecanismo que busca **verificar** ou provar a **verdade**"

# Teste no nosso cotidiano

— — —



# Benefícios do teste

— — —

- Dá os primeiros sentidos das nossas observações
- Feedback
- + Segurança
- + Confiança

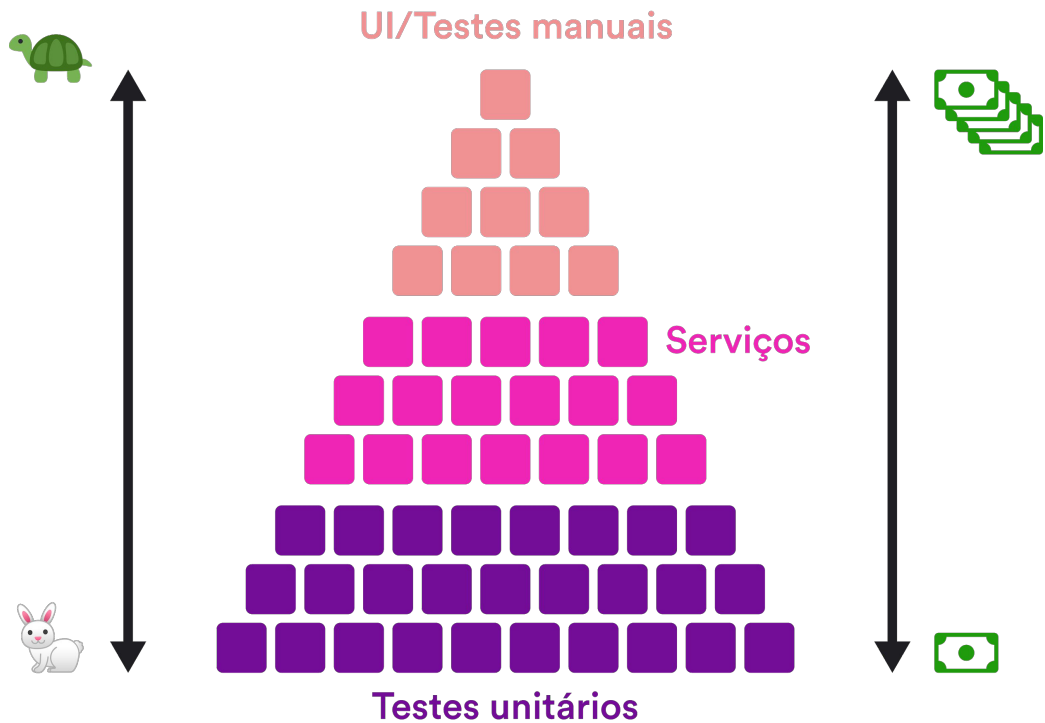


# E o que isso tem a ver com software?





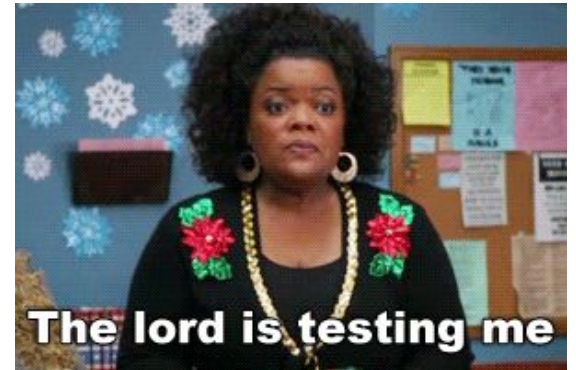
# Pirâmide de Testes



*"Escreva testes até que o medo  
seja transformado em tédio".*

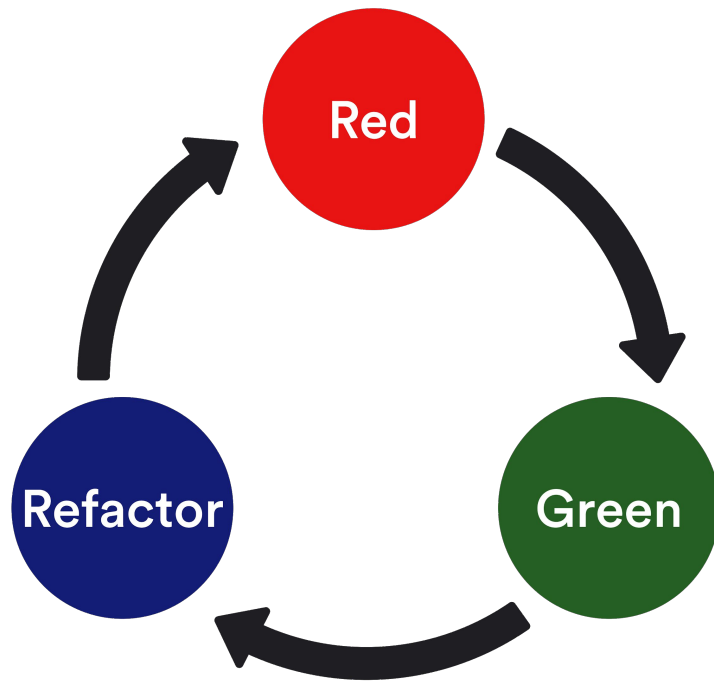
*Kent Beck*

# Falando de TDD



# O que é TDD?

Uma metodologia de desenvolvimento de software que prima por ciclos curtos de teste, implementação e refatoração, nesta ordem.



---

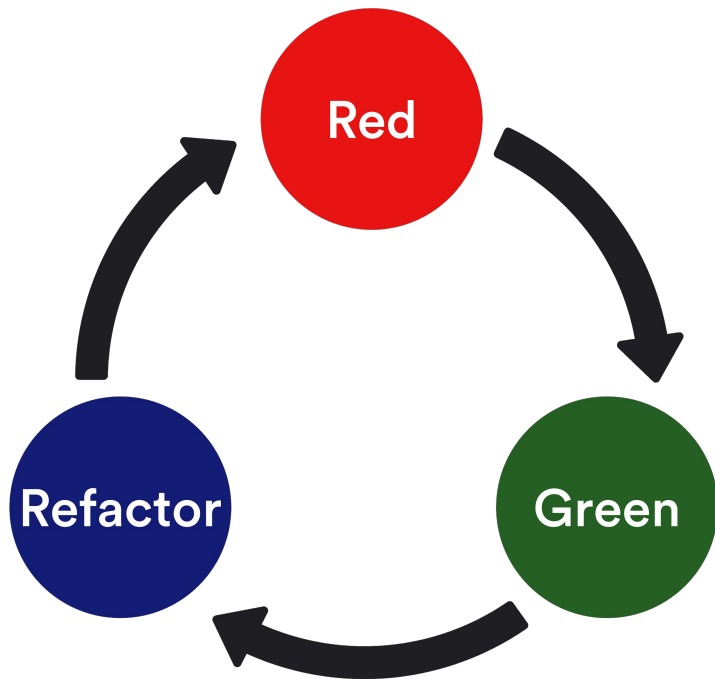
# O que é TDD?

— — —

1- Não se deve escrever código de produção até criar um teste de unidade de falha.

2- Não se deve escrever mais de um teste de unidade necessário para falhar.

3- Não se deve escrever mais código de produção do que o necessário para resolver o teste.



# Razões para TDD

— — —

- Escrever o **mínimo necessário** de código para atender os requisitos do negócio.
- Criar uma coleção **executável** de cenários da especificação do software.
- Dar a segurança necessária para introduzir **mudanças** no software.

# Benefícios TDD

— — —

- Faz com que **cada linha** do software seja testada
- Proporciona código **limpo e legível**
- Mantém a implementação **simples e testável**
- Testes unitários são **documentos atualizados** do seu código
- Menos Bugs em produção
- Facilidade de refatoração

# Python com TDD





# Python com TDD - Pytest

— — —

- Ferramenta de teste python.
- Rápido e com boa legibilidade.

# Exemplo de teste:

```
1
2
3  def incrementar(x):
4      return x + 1
5
6
7  def test_incrementar():
8      assert incrementar(3) == 4
```



# Python com TDD- Pytest

- Como instalar o pytest no meu projeto?

- Executando o comando:

```
pip install pytest
```

- Onde escrever os testes?

- Mantenha seu código de teste separado do código da aplicação:

```
setup.py  
mypkg/  
  __init__.py  
  app.py  
  view.py  
tests/  
  test_app.py  
  test_view.py  
  ...
```

# Python com TDD - Convenções para Testes

— — —

- Como escrever testes unitários:
  - O nome do arquivo deve iniciar com Test. ex: *test\_fizzbuzz.py*
  - O nome do método deve iniciar com test.  
ex: *test\_deve\_retornar\_isso()*
- Assertions
  - `assert` esperado == saída
- Como rodar os testes:
  - Executar o comando `pytest` no terminal
  - `pytest`

# Padrões e Práticas de Teste



- Pequenas unidades →  
Separação de conceitos
- Divisão de complexidade
- Faça o mais simples possível
- Nomeie seus testes de forma descritiva, [aqui dicas de como nomear seus testes](#)



- Nomes de testes ruins
- Testes vazios (no assertions)
- Testes não determinísticos  
(causes flaky tests)

# Python com TDD - Estudo de Caso

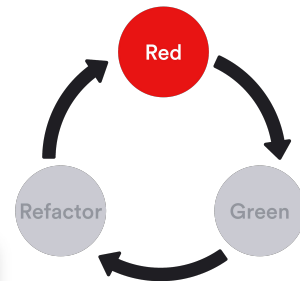
— — —

## **FizzBuzz**

Dado um número, o programa deve:

- Retornar "Fizz" se o número é um múltiplo de 3;
- Retornar "Buzz" se o número é um múltiplo de 5;
- Retornar "FizzBuzz" se o número é um múltiplo de 3 e 5 simultaneamente;
- Retornar o próprio número se não for múltiplo de nenhum.

- Retornar "Fizz" se o número é um múltiplo de 3;

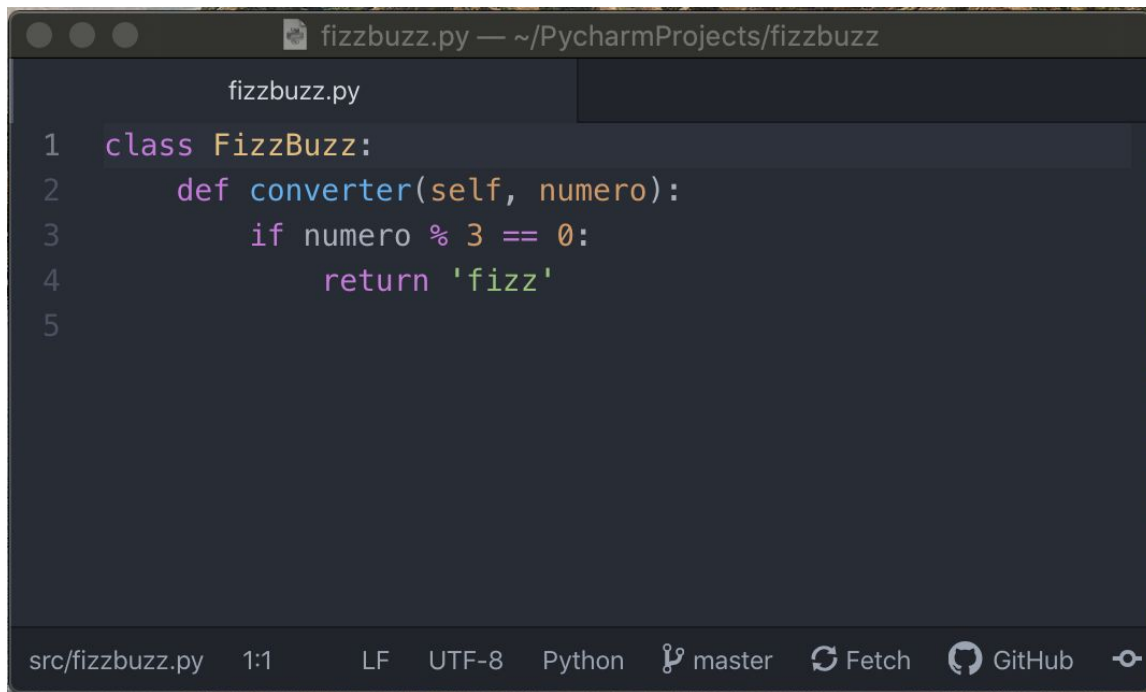


```
test_fizzbuzz.py — ~/PycharmProjects/fizzbuzz

test_fizzbuzz.py
1  from src.fizzbuzz import FizzBuzz
2
3
4  def test_deve_retornar_fizz_quando_multiplo_de_3():
5      fizzbuzz = FizzBuzz()
6      assert 'fizz' == fizzbuzz.converter(3)
7
```

test/test\_fizzbuzz.py 7: LF UTF-8 Python master Fetch GitHub

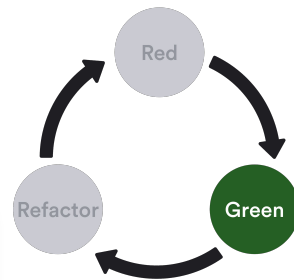
- Retornar "Fizz" se o número é um múltiplo de 3;



```
fizzbuzz.py — ~/PycharmProjects/fizzbuzz

fizzbuzz.py
1 class FizzBuzz:
2     def converter(self, numero):
3         if numero % 3 == 0:
4             return 'fizz'
5

src/fizzbuzz.py 1:1  LF  UTF-8  Python  master  Fetch  GitHub
```



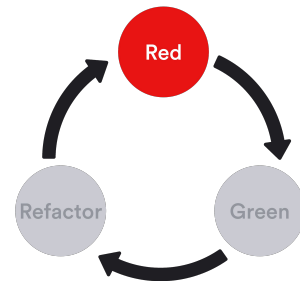


- Retornar "Buzz" se o número é um múltiplo de 5;

```
test_fizzbuzz.py — ~/PycharmProjects/fizzbuzz

test_fizzbuzz.py
1  from src.fizzbuzz import FizzBuzz
2
3
4  def test_deve_retornar_fizz_quando_multiplo_de_3():
5      fizzbuzz = FizzBuzz()
6      assert 'fizz' == fizzbuzz.converter(3)
7
8  def test_deve_retornar_buzz_quando_multiplo_de_5():
9      fizzbuzz = FizzBuzz()
10     assert 'buzz' == fizzbuzz.converter(5)
11
```

test/test\_fizzbuzz.py 1: LF UTF-8 Python master Fetch GitHub

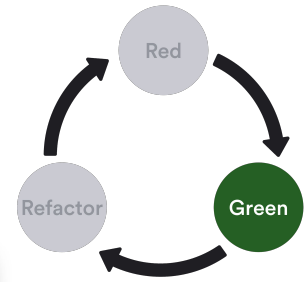


- Retornar "Buzz" se o número é um múltiplo de 5;

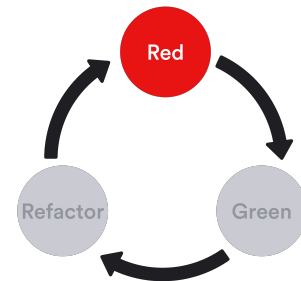
```
fizzbuzz.py — ~/PycharmProjects/fizzbuzz

fizzbuzz.py
1 class FizzBuzz:
2     def converter(self, numero):
3         if numero % 3 == 0:
4             return 'fizz'
5         if numero % 5 == 0:
6             return 'buzz'
7

src/fizzbuzz.py 1:1 LF UTF-8 Python ⓘ master ↻ Fetch GitHub 🔑
```



- Retornar "FizzBuzz" se o número é um múltiplo de 3 e 5 simultaneamente;



```
test_fizzbuzz.py — ~/PycharmProjects/fizzbuzz

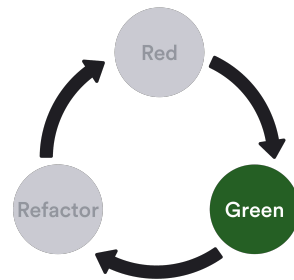
test_fizzbuzz.py
1  from src.fizzbuzz import FizzBuzz
2
3  > def test_deve_retornar_fizz_quando_multiplo_de_3(): ...
6
7  > def test_deve_retornar_buzz_quando_multiplo_de_5(): ...
10
11 def test_deve_retornar_fizzbuzz_quando_multiplo_de_3_e_5():
12     fizzbuzz = FizzBuzz()
13     assert 'fizzbuzz' == fizzbuzz.converter(15)
14

test/test_fizzbuzz.py  7:1  LF  UTF-8  Python  master  Fetch  GitHub  Git (4
```

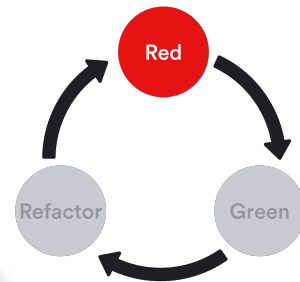
- Retornar "FizzBuzz" se o número é um múltiplo de 3 e 5 simultaneamente;

```
fizzbuzz.py — ~/PycharmProjects/fizzbuzz
fizzbuzz.py
1 class FizzBuzz:
2     def converter(self, numero):
3         saida = ''
4         if numero % 3 == 0:
5             saida += 'fizz'
6         if numero % 5 == 0:
7             saida += 'buzz'
8         return saida
9
```

src/fizzbuzz.py 1: LF UTF-8 Python master Fetch GitHub



- Retornar o próprio número se não for múltiplo de nenhum.

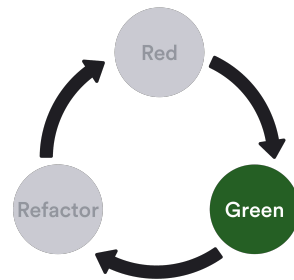


```
test_fizzbuzz.py — ~/PycharmProjects/fizzbuzz

test_fizzbuzz.py
1  from src.fizzbuzz import FizzBuzz
2
3  > def test_deve_retornar_fizz_quando_multiplo_de_3():
6
7  > def test_deve_retornar_buzz_quando_multiplo_de_5():
10
11 > def test_deve_retornar_fizzbuzz_quando_multiplo_de_3_e_5():
14
15 def test_deve_retornar_o_proprio_numero_quando_nao_multiplo_de_3_ou_5():
16     fizzbuzz = FizzBuzz()
17     assert '1' == fizzbuzz.converter(1)
```

test/test\_fizzbuzz.py 1:1    LF   UTF-8   Python   master   Fetch   GitHub   Git (4)

- Retornar o próprio número se não for múltiplo de nenhum.



```
fizzbuzz.py — ~/PycharmProjects/fizzbuzz

fizzbuzz.py
1  class FizzBuzz:
2      def converter(self, numero):
3          saida = ''
4          if numero % 3 == 0:
5              saida += 'fizz'
6          if numero % 5 == 0:
7              saida += 'buzz'
8          if saida == '':
9              saida += str(numero)
10         return saida
11

src/fizzbuzz.py  1:  LF  UTF-8  Python  master  Fetch  GitHub
```

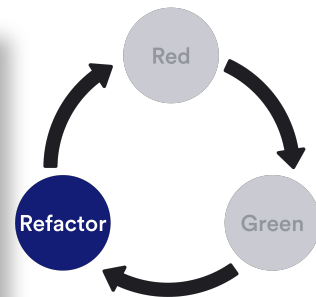
**Vamos refatorar**



fizzbuzz.py — ~/PycharmProjects/fizzbuzz

```
fizzbuzz.py
1 class FizzBuzz:
2     def converter(self, numero):
3         saida = ''
4
5         if self.eh_multiplo_de_3(numero):
6             saida += 'fizz'
7         if self.eh_multiplo_de_5(numero):
8             saida += 'buzz'
9         if saida == '':
10             saida += str(numero)
11         return saida
12
13     def eh_multiplo_de_3(self, numero):
14         return numero % 3 == 0
15
16     def eh_multiplo_de_5(self, numero):
17         return numero % 5 == 0
```

src/fizzbuzz.py 17:31 LF UTF-8 Python master Fetch GitHub Gi



# Python com TDD - Estudo de Caso

— — —

## Novo requisito!



Dado um número, o programa deve:

- Retornar "Fizz" se o número é um múltiplo de 3;
- Retornar "Buzz" se o número é um múltiplo de 5;
- Retornar "FizzBuzz" se o número é um múltiplo de 3 e 5 simultaneamente;
- Retornar o próprio número se não for múltiplo de nenhum;
- Retornar um erro quando um número menor ou igual a zero.

# Python com TDD - Estudo de Caso

— — —

## Números Primos

Dado um número, o programa deve:

- Retornar "Falso" se o valor for 1;
- Retornar "Verdadeiro" se o número é divisível apenas por 1 e por ele mesmo;
- Retornar "Falso" se o número é divisível por 1, por ele mesmo e ainda possui outro (s) divisor (es);

# Próximos passos

— — —

- Praticando:
  - <http://codewars.com>
  - Tente adicionar 1 teste unitário ao seu código :)
- <https://docs.pytest.org/en/stable/>
- <https://dzone.com/articles/7-popular-unit-test-naming>
- Repo [FizzBuzz](#) e [Números Primos](#)

# OBRIGADA!

**JESSYKA LAGE**

DEVELOPER | SÃO PAULO

Twitter: @jessykalage

Instagram: @jessykalage

LinkedIn: Jessyka Lage

Github: @jessyka



<https://bit.ly/2ATDFAA>