

## Caso Valida Método de Desconto

### Estória:

Maria trabalha em uma empresa do ramo de **E-Commerce** e seu gestor pediu para ela validar um método chamado "calcula\_preco\_com\_desconto". O objetivo desse método é **calcular o preço final** de um produto dado um **valor base** e uma **porcentagem de desconto**. Ex.: Se o **valor base for 100**, e a **porcentagem for 20**, a método deverá retornar:

$$100 - (100 * (20/100)) = 80$$

Contudo, ao ser passado um valor *X* e sua porcentagem *Y%* para esse método, o mesmo está **retornando um valor maior do que esperado**. Os clientes da loja virtual estão muito confusos, e alguns deles desistiram de realizar a compra.

O principal motivo dessa desistência é por, além de não receberem o desconto, o valor gerado no pedido estar sendo **maior** do que o **valor inicial**.

Maria, ao estudar o código do método, notou que por se tratar de um **parte isolada do sistema** uma boa alternativa seria criar um **teste de unidade** para garantir que essa função esteja **retornando os valores corretos**(Imagens logo abaixo).

Função a ser validada:

```
#Classe que contém as operações 'desconto' e 'acrécimo'
class Operacoes:
    def calcula_preco_com_desconto(self, valor, porcentagem):
        desconto = (valor * porcentagem) / 100
        return valor + desconto

    def calcula_preco_com_acrescimo(self, valor, porcentagem):
        acrescimo = (valor * porcentagem) / 100
        return valor + acrescimo
```

## Criação da Classe de Teste e Saída com Erro:

```
teste_unidade.py  teste_desconto.py X
teste_desconto.py > Operacoes > calcula_preco_com_desconto
1  import unittest
2
3  #Classe que contém as operações 'desconto' e 'acrécimo'
4  class Operacoes:
5      def calcula_preco_com_desconto(self, valor, porcentagem):
6          desconto = (valor * porcentagem) / 100
7          return valor + desconto
8
9      def calcula_preco_com_acrescimo(self, valor, porcentagem):
10         acrescimo = (valor * porcentagem) / 100
11         return valor + acrescimo
12
13 # Classe de teste que herda de unittest.TestCase
14 class TestCalculaDesconto(unittest.TestCase):
15     def test_calcula_preco_com_desconto(self):
16         valor = 100
17         porcentagem = 20
18         operacoes = Operacoes()
19         resultado = operacoes.calcula_preco_com_desconto(valor, porcentagem)
20         self.assertEqual(resultado, 80)# valida se para o argumento 100 o resultado é 80
21
22 if __name__ == '__main__':
23     unittest.main()
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
AssertionError: 120.0 != 80
-----
Ran 1 test in 0.000s
FAILED (failures=1)
```

## Função corrigida após realização do teste:

```
teste_unidade.py  teste_desconto.py X
teste_desconto.py > ...
1  import unittest
2
3  #Classe que contém as operações 'desconto' e 'acrécimo'
4  class Operacoes:
5      def calcula_preco_com_desconto(self, valor, porcentagem):
6          desconto = (valor * porcentagem) / 100
7          return valor - desconto
8
9      def calcula_preco_com_acrescimo(self, valor, porcentagem):
10         acrescimo = (valor * porcentagem) / 100
11         return valor + acrescimo
12
13 # Classe de teste que herda de unittest.TestCase
14 class TestCalculaDesconto(unittest.TestCase):
15     def test_calcula_preco_com_desconto(self):
16         valor = 100
17         porcentagem = 20
18         operacoes = Operacoes()
19         resultado = operacoes.calcula_preco_com_desconto(valor, porcentagem)
20         self.assertEqual(resultado, 80)# valida se para o argumento 100 o resultado é 80
21
22 if __name__ == '__main__':
23     unittest.main()
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
-----
Ran 1 test in 0.000s
OK
```