Profa. Berô
Engenharia de Software

Testes unitários com Python

A biblioteca **unittest** é o módulo de testes unitários padrão do Python.

A biblioteca fornece um conjunto de ferramentas para escrever e executar testes automatizados em Python, permitindo que você verifique se o comportamento **de uma parte específica do código** está correto ou não.

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Profa. Berô
Engenharia de Software

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y

class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Profa. Berô
Engenharia de Software

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y

class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

Importe a biblioteca

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Profa. Berô
Engenharia de Software

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y


class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

crie uma classe teste que herda de unittest.TestCase

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Profa. Berô
Engenharia de Software

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y

class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

Os nomes dos métodos de teste devem começar com a palavra "test" para que a biblioteca unittest os reconheça como métodos de teste

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Profa. Berô
Engenharia de Software

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y

class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

O self é uma referência ao objeto atual que está sendo manipulado dentro da classe.

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y

class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

o método assertEqual verifica se o resultado da chamada da função é o que esperamos.

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Profa. Berô
Engenharia de Software

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y

class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Profa. Berô
Engenharia de Software

Testes unitários com Python

```python
import unittest

def add(x, y):
    return x + y

class Teste_adicao(unittest.TestCase):

    def testando_add_se_positivo(self):
        self.assertEqual(add(2, 3), 5)

    def testando_add_se_negativo(self):
        self.assertEqual(add(-2, -3), -5)

    def testando_add_com_zero(self):
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

bloco será executado somente se o programa estiver sendo executado diretamente como o programa principal e não se estiver sendo importado como um módulo.