ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Testes automatizados

Testes de software

É muito importante realizar testes para determinar a qualidade do software.

Testes exploratórios têm sido realizados ao longo deste curso, uma vez que são feitos sem planejamento.

Testes de software

Um processo completo e manual de verificação de software envolve listar todas as suas funcionalidades, tipos diferentes de entrada e respectivos resultados esperados.

A cada modificação do software é necessário executar todo processo de verificação novamente, o que torna os testes manuais caros e propensos a erros.

Testes automatizados

É a execução automática de um plano de testes.

Existem dois tipos de testes automatizados:

- Testes unitários

- Testes integrados

Nos testes integrados, é um desafio identificar um problema uma vez que o resultado produzido é diferente do esperado.

Testes automatizados

Exemplo simples:

```
>>> assert sum([1, 2, 3]) == 6, "Deve ser 6"

>>> assert sum([1, 1, 1]) == 6, "Deve ser 6"
Traceback (most recent call last):
  File "<input>", line 1, in <module>
AssertionError: Deve ser 6
```

Testes automatizados

Exemplo simples: Vamos testar duas funções, soma e mult, sendo que a primeira está incorreta.

```
def soma(arg):
    total = 1
    for i in arg:
        total += i
    return total
```

```
def mult(arg):
    total = 1
    for i in arg:
        total *= i
    return total
```

Suponha que ambas funções estão definidas em func.py.

Testes automatizados

Utilização de um arquivo de teste (teste.py):

```
from func import soma, mult


def test_soma():
    assert soma([1, 2, 3]) == 6, "Deve ser 6"

def test_mult():
    assert mult((2, 3, 4)) == 24, "Deve ser 24"

if __name__ == "__main__":
    test_soma()
    test_mult()
    print('Tudo ok!')
```

```
MacBook-Pro-2:Exemplo01 manzato$ python3 teste.py
Traceback (most recent call last):
  File "teste.py", line 10, in <module>
    test_soma()
  File "teste.py", line 4, in test_soma
    assert soma([1, 2, 3]) == 6, "Deve ser 6"
AssertionError: Deve ser 6
MacBook-Pro-2:Exemplo01 manzato$
```

Problema:

Como executar todos os casos de teste, mesmo que algum falhe?

Test Runners

Aplicação especialmente projetada para executar testes, checar a saída e fornecer ferramentas para depurar e diagnosticar testes e programas.

unittest é um exemplo disponível na biblioteca padrão Python.

Outros test runners: nose/nose2, pytest, etc.

unittest

Para usar o unittest, é necessário:

- **agrupar os testes em métodos de uma ou mais classes que herdam de unittest.TestCase**
- **usar métodos específicos da classe unittest.TestCase (e.g. assertEqual) para cada caso de teste**

unittest

Exemplo (arquivo test_func.py):

```
MacBook-Pro-2:Exemplo01 manzato$ python3 test_func.py
F.
======================================================================
FAIL: test_sum1 (__main__.TestSum)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_func.py", line 6, in test_sum1
    self.assertEqual(soma([1, 2, 3]), 6, "Deve ser 6")
AssertionError: 7 != 6 : Deve ser 6

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)
MacBook-Pro-2:Exemplo01 manzato$
```

```
import unittest
from func import soma, mult

class TestSum(unittest.TestCase):
    def test_sum1(self):
        self.assertEqual(soma([1, 2, 3]), 6, "Deve ser 6")

    def test_sum2(self):
        self.assertEqual(mult((2, 3, 4)), 24, "Deve ser 24")

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

unittest

A validação da saída em relação a um resultado esperado é chamada de assertion.

Boas práticas:

- Certificar que o teste pode ser repetido e que, ao executá-lo, múltiplas vezes, o mesmo resultado será gerado.
- Criar assertions que de fato relacionam a saída gerada com a entrada fornecida.

unittest

Tipos de assertions:

Method	Equivalent to
`.assertEqual(a, b)`	`a == b`
`.assertTrue(x)`	`bool(x) is True`
`.assertFalse(x)`	`bool(x) is False`
`.assertIs(a, b)`	`a is b`
`.assertIsNone(x)`	`x is None`
`.assertIn(a, b)`	`a in b`
`.assertIsInstance(a, b)`	`isinstance(a, b)`

Fonte: https://realpython.com/python-testing/

unittest

Se houver vários arquivos de teste, pode-se executar todos automaticamente, por meio da instrução:

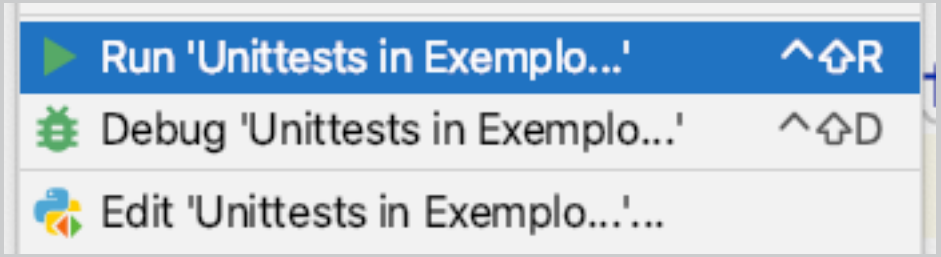
python –m unittest discover

Essa instrução irá procurar na pasta atual todos os arquivos no formato test*.py, executando-os um a um.

unittest

Também é possível executar os testes via IDE.

No PyCharm, basta clicar com o botão direito no projeto, e selecionar Run 'Unittests in...'



ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Testes automatizados