GABRIELE ROMANATO

Introduzione al Test-Driven Development (TDD) in Python

Il Test-Driven Development (TDD) è una metodologia di sviluppo software che prevede la scrittura dei test prima dell'implementazione del codice. Questo approccio aiuta a garantire che il codice sia affidabile, manutenibile e ben progettato.

Fasi del TDD

1. **Scrivi un test che fallisce**: il test descrive un comportamento atteso.
2. **Scrivi il codice minimo per farlo passare**: implementazione basilare per superare il test.
3. **Refactoring**: migliora la struttura del codice mantenendo i test verdi.

Strumenti in Python

In Python, il modulo `unittest` è incluso nella libreria standard. Altri strumenti popolari includono `pytest` e `nose2`.

Esempio pratico con `unittest`

Supponiamo di voler creare una funzione che somma due numeri.

1. Scrivere il test

```
import unittest

from calculator import add

class TestCalculator(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(2, 3), 5)
```

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

2. Implementare la funzione

```
# calculator.py
def add(a, b):
    return a + b
```

3. Refactoring (se necessario)

Nel nostro caso, la funzione è già semplice e leggibile, quindi non è necessario un refactoring.

Vantaggi del TDD

- Maggiore affidabilità del codice
- Facilità di refactoring e manutenzione
- Progettazione guidata dal comportamento

Conclusione

Il TDD è una pratica potente che può migliorare significativamente la qualità del software. Anche se richiede disciplina e un cambiamento di mentalità, i benefici nel lungo termine sono evidenti.