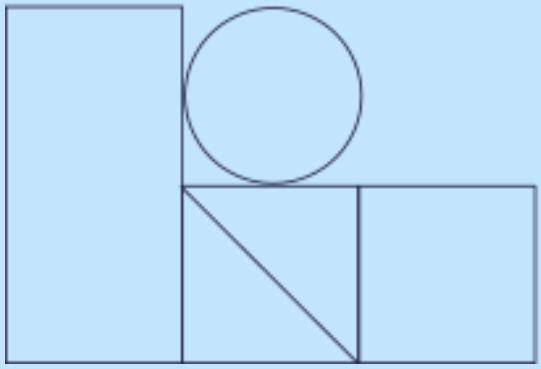
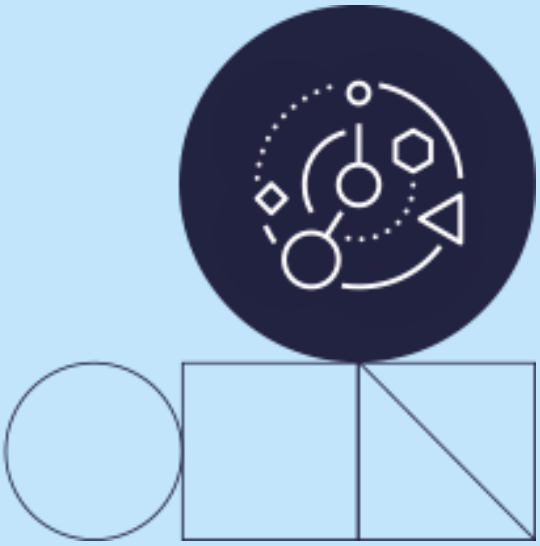
IBM SkillsBuild

IBM

# Testes com Python

## Testes de unidade e integração





# Índice

# Introdução

De todos os tipos de testes que vimos no tópico anterior, vamos nos concentrar em testes unitários e de integração.

Os projetos de software de hoje são mais complexos do que nunca. A indústria de software está crescendo a uma velocidade muito alta e os desenvolvedores de software têm que acompanhar os últimos desenvolvimentos e oportunidades no mundo do software a fim de usar a tecnologia mais recente da melhor maneira possível.

Existem muitos tipos de negócios, mas no desenvolvimento de software, a qualidade é essencial para o sucesso. Para garantir este sucesso, a função de teste é vital.

Nenhum bom desenvolvedor implementa código sem testes completos.

O teste da unidade Python é uma estrutura de teste integrada para testar o código Python. Possui um test runner, o que nos permite realizar os testes sem muito esforço. Assim, podemos usar o módulo de teste da unidade para testar sem usar módulos de terceiros.

Enquanto os testes unitários fazem testes em partes do código independentemente, testes de integração se tornam necessários quando queremos testar sistematicamente um projeto como um todo.

Os testes de integração exercitam duas ou mais partes de uma aplicação de cada vez, incluindo as interações entre as partes, para determinar se elas funcionam como pretendido. Este tipo de teste identifica defeitos nas interfaces entre partes distintas de uma base de códigos à medida que invocam e passam dados uns aos outros.

Estas duas abordagens devem ser usadas em conjunto, em vez de apenas uma abordagem sobre a outra. Quando um sistema é testado exaustivamente por unidade, o teste de integração é muito mais fácil porque muitos dos erros nos componentes individuais já terão sido encontrados e corrigidos.

Portanto, é necessário realizar tanto testes de integração quanto testes unitários desde o início de nosso desenvolvimento. A combinação dos dois assegura que o código proposto é de boa qualidade e resolve verdadeiramente os problemas para os quais ele foi desenvolvido.

medida que a escala de uma base de código aumenta, tanto os testes de unidade quanto os de integração permitem aos desenvolvedores identificar rapidamente mudanças importantes em seu código. Muitas vezes, estas importantes mudanças são involuntárias e só serão conhecidas mais tarde no ciclo de desenvolvimento, possivelmente quando um usuário final descobre o problema enquanto utiliza o software. A unidade automatizada e os testes de integração aumentam muito a probabilidade de que os bugs sejam encontrados o mais cedo possível durante o desenvolvimento, para que possam ser corrigidos imediatamente.

# Testes unitarios

Se partirmos do pressuposto de que o teste unitário é algo como a "unidade básica" do teste em um programa/aplicação, então vamos começar com ele e passar gradualmente para aspectos mais complexos mais tarde.

O módulo unittest da biblioteca padrão Python (stdlib) está encarregado de realizar testes unitários se não quisermos usar bibliotecas externas. Também é conhecido como PyUnit, onde o sufixo Unit se refere a um conjunto de parâmetros propositalmente construídos para executar testes unitários em diferentes linguagens de programação, conhecido como XUnit, onde X é o prefixo que aponta para a linguagem de programação específica. Em nosso caso, Py, o prefixo, aponta para Python, como podemos facilmente deduzir. Vamos lá, eu digo.

Isto está relacionado ao processo de teste unitário, que é universal para todas as linguagens de programação. seu uso é paralelo, ou pelo menos geralmente é, à modelagem de uma classe, a fim de testar de imediato a adequação de seus métodos antes de modelar a próxima classe; ou se esta classe (e, consequentemente, o módulo que a contém) sofrer algum tipo de modificação. será sempre conveniente modelar uma classe de teste específica (unit testing). Dentro do conjunto estilo Python, é considerado boa prática incluir uma classe de teste para cada módulo em nosso projeto. tudo isso cria uma espécie de ambiente ou estrutura de teste, e é por isso que o módulo de teste unitário também é conhecido como estrutura de teste.

Este módulo Unittest tem uma série de ferramentas que nos permitirá testar nosso programa ou aplicação dentro do mesmo código.

# Testes de integração

As **testes de integração de software** são a ferramenta que reúne cada um dos módulos de um sistema para testar como eles funcionam uns com os outros. Este tipo de teste é realizado nas primeiras etapas, após os testes unitários, nos quais um fragmento do código fonte é analisado.

Uma vez analisadas as unidades individuais, é necessário verificar se os módulos não interferem com o resto das funções.

Imaginemos que um aplicativo de e-mail seja dividido em três unidades - página inicial, caixa de entrada e lixo. Ao executar um teste de integração, o desenvolvedor deve verificar se a ligação entre uma unidade e a outra é ótima.

A sessão inicial deve ser encaminhada corretamente para a caixa de entrada, e a exclusão de um e-mail deve ser imediatamente colocada na lixeira de reciclagem. Se houver uma interface errada, isto causará problemas no futuro e os usuários não poderão usar o sistema.

## Que erros podem ser detectados por este tipo de teste?

Os testes unitários são um primeiro filtro para detectar bugs em sistemas, entretanto, eles não identificam sua relação com outras interfaces. Daí a importância de implementar **testes de integração de software**.

Entre os problemas mais comuns identificados por este tipo de teste estão a perda de conectividade, formatação de dados e respostas inesperadas.

A detecção oportuna de erros pode minimizar o impacto econômico e temporal, para que os desenvolvedores possam concentrar seus esforços em outros projetos e tarefas prioritárias.

## Níveis de testes de integração

Há dois níveis de testes de integração: componente e sistema. O primeiro avalia os elementos integrados em um único sistema, enquanto o segundo testa a relação entre as interfaces externas do sistema.

As **testes de integração de componentes** são geralmente executados primeiro antes do **testes de integração de sistemas**, após a verificação das unidades de código individuais. Uma vez assegurada esta etapa, é possível realizar os *tests* envolvendo interfaces fora do sistema original.

## Tipos de testes de integração

### Big Bang

Um teste de integração do *Big Bang* concentra todos os módulos de um sistema para verificar se eles funcionam em conjunto, portanto, antes de executar o teste, o desenvolvedor deve certificar-se de que cada unidade foi concluída.

Este tipo de teste é viável para pequenos projetos, caso contrário, erros significativos podem ser negligenciados.

### Ad Hoc

Este termo se refere ao desenvolvimento de uma solução para um problema específico. Para fins de testing de software, este tipo de teste de integração pode ser executado a qualquer momento, amplamente recomendado em estágios iniciais, com o objetivo de encontrar erros imprevistos.

Entre suas vantagens está o fato de ser rápido e exigir pouco planejamento. Entretanto, dificuldades podem ser encontradas em fases posteriores, pois não é necessária nenhuma documentação para contabilizar os erros encontrados.

### Top Down

Como o nome sugere, os testes *top down* (de cima para baixo) iniciam a análise do código em módulos downstream, onde as informações centrais são concentradas e o downstream se conecta a outras interfaces.

Neste modelo, erros de maior relevância podem ser detectados e, devido à complexidade de sua composição, as melhorias podem levar mais tempo.

### Down Top

Ao contrário do modelo anterior, o teste de integração *down top* continua para cima. Neste caso, os problemas são mais fáceis de detectar, assim como as melhorias a serem feitas.

A desvantagem é que módulos complexos são colocados no final do teste e a entrega do produto final pode levar mais tempo do que o estimado.

### Hybrid

O teste de integração híbrida - também conhecido como teste em sanduíche - inclui as práticas dos dois modelos anteriores - *top down* e *down top* -. O desenvolvedor pode escolher os módulos downstream ou downstream simultaneamente com o objetivo de encontrar bugs em menos tempo.

Entretanto, requer equipamento altamente treinado para a detecção precisa e oportuna de erros. Este tipo de teste é recomendado para sistemas operacionais mais complexos.