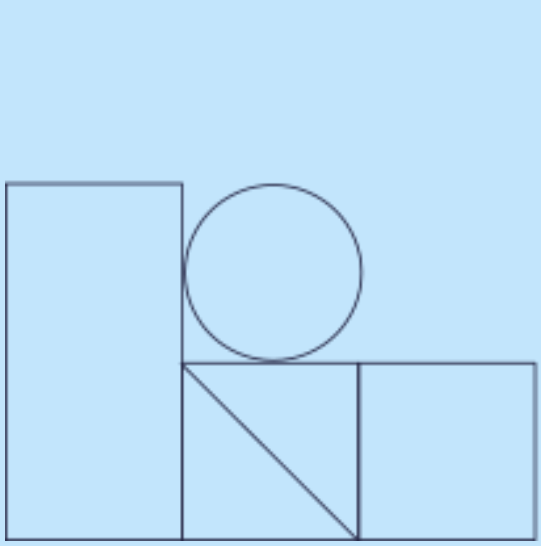
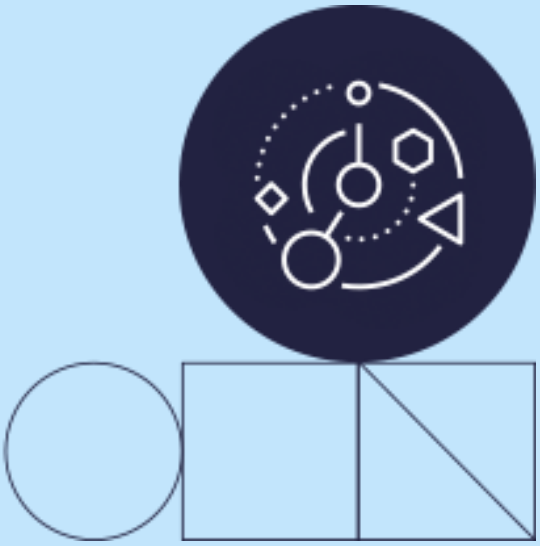
IBM SkillsBuild

IBM

Testes com Python

Bibliotecas para testes unitários





Índice

Introdução

Como vimos nos capítulos anteriores:

- Os testes automatizados são um contexto bem conhecido no mundo dos testes. É onde os planos de teste são executados usando um roteiro, em vez de um humano.
- Python vem com as ferramentas e bibliotecas que suportam testes automatizados para seu sistema.
- Os casos de teste Python são comparativamente fáceis de escrever. Com o aumento do uso do Python, as estruturas de automação de testes baseadas em Python também estão se tornando populares.

Lista de estruturas de teste Python

A seguir, listaremos as características das principais estruturas de teste baseadas em Python.

- Robot
- PyTest
- Prueba de unidad
- DocTest
- Nariz2
- Testify
- Unittest

Comparação das ferramentas de teste

Vamos resumir rapidamente estas estruturas em uma breve tabela comparativa:

	Licença	Parte de	Categoria	Categoria Característica especial
pytest.warns ()	Aviso_esperado: Expectativa [coincidência]	Assinalar aviso com as funções		
Robot	Software livre (Licença ASF)	Biblioteca de testes genéricos Python	Teste de aceitação	Abordagem de teste com base em palavras-chave
pytest	Software livre (Licença MIT)	Autônomo, permite conjuntos de teste compactos	Revisão da unidade	Acessório de classe especial e simples para facilitar os testes
unittest	Software livre (Licença MIT)	Parte da biblioteca padrão Python	Revisão da unidade	Coleta rápida de testes e execução flexível de testes
DOCtest	Software livre (Licença MIT)	Parte da biblioteca padrão Python	Revisão da unidade	Python Interative Shell para aplicação nclusiva e ronta para comando
nose	Software livre (Licença BSD)	Ele carrega funções de unittest com funções adicionais e complementos.	extensão unittest	Um grande número de complementos
Testify	Software livre (Licença ASF)	Ele traz características de unittest e nose com características adicionais e complementos.	extensão unittest	Melhorando a descoberta de provas

Estas são as estruturas de teste Python mais populares atualmente, mas poderíamos incluir nesta lista algumas que poderiam se tornar populares no futuro.

Behave

- Behave usa linguagem natural para escrever testes e trabalha com cordas Unicode. Ele utiliza o chamado **BDD (desenvolvimento orientado pelo comportamento).** É uma estrutura de teste que também é utilizada para **Testes da caixa preta.**
- O diretório Behave contém **arquivos de características** que têm texto simples, parece uma linguagem natural e **Implementações da etapa Python**.

lettuce

- lettuce é útil para **Testes de desenvolvimento orientados pelo comportamento**. Torna o processo de teste fácil e escalável.
- **lettuce** inclui etapas tais como:
- Descrevendo o comportamento
- Definindo etapas em Python.
- Executando o código
- Modificando o código para passar no teste.
- Executando o código modificado.
- Estes passos são seguidos 3 a 4 vezes para que o software esteja livre de erros e assim melhorar sua qualidade.

Como escolher a melhor estrutura de testes Python?

Compreender as vantagens e limitações de cada estrutura é a melhor maneira de escolher corretamente a melhor estrutura de testes para seu projeto.

Robot

Vantagens

- A abordagem de teste baseada em palavras-chave ajuda a criar casos de teste legíveis de uma maneira mais fácil.
- Múltiplos APIs
- Sintaxe de dados de teste simples
- Suporta testes paralelos via Selenium Grid.

Limitações

- Criar relatórios HTML personalizados é bastante complicado com o Robô.
- Menos suporte para testes paralelos.
- Requer Python 2.7.14 e superior.

Pytest

Vantagens

- Suporta um conjunto compacto de testes.
- Não é necessário um depurador ou registro explícito de teste.
- Acessórios múltiplos
- Complementos extensíveis
- Criação de teste fácil e simples.
- Os casos de teste podem ser criados com menos erros.

Limitações

- Não compatível com outros frameworks.

Unittest

Ventagens

- Não são necessários módulos adicionais.
- Fácil de aprender para os testadores de nível iniciante.
- Execução de testes simples e fácil.
- Geração rápida de relatórios de teste.

Limitações

- O nome snake_case Python e o nome camelCase JUnit causam alguma confusão.
- Intenção pouco clara do código de teste.
- Requer muito código repetitivo.

Doctest

Vantagens

- Uma boa opção para pequenos testes.
- A documentação de teste dentro do método também fornece informações adicionais sobre como o método funciona.

Limitações

- Comparar apenas a produção impressa. Qualquer variação na saída fará com que o teste falhe.

Nose2

Vantagens

- Nose2 suporta mais configurações de teste do que unittest.
- Inclui um conjunto substancial de suplementos ativos.
- API diferente do unittest que fornece mais informações sobre o erro.

Limitações

- Ao instalar plugins de terceiros, você deve instalar a ferramenta de configuração / distribuir o pacote, já que o Nose2 suporta Python 3, mas não plugins de terceiros.

Testify

Vantagens

- Fácil de entender e usar.
- Os testes de unidade, integração e sistema podem ser facilmente criados.
- Componentes de teste gerenciáveis e reutilizáveis.
- Adicionar novas características para Testify é fácil.

Limitações

- - Inicialmente, Testify foi desenvolvido para substituir o Unittest e o Nose, mas o processo de transição para o pytest está em andamento, por isso os usuários são aconselhados a evitar o uso de Testify em alguns projetos futuros.

Behave Framework

Vantagens

- Fácil execução de todos os tipos de casos de teste.
- Raciocínio e pensamento detalhados
- Clareza dos resultados de QA / Dev.

Limitações

- Suporta apenas testes de caixa preta.

Lettuce

Vantagens

- Linguagem simples para a criação de múltiplos cenários de teste.
- Útil para casos de teste baseados em comportamento para teste de caixa preta.

Limitações

- Ela precisa de forte coordenação entre desenvolvedores, testadores e partes interessadas.

Você pode escolher a estrutura de teste Python mais adequada, considerando as vantagens e limitações acima que o ajudarão a desenvolver os critérios certos para suas necessidades comerciais.