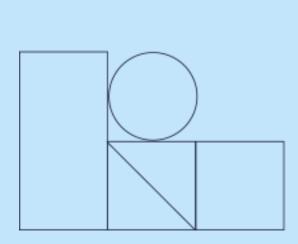
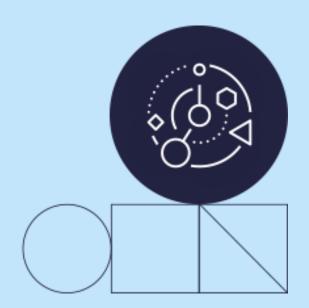
Pruebas con Python

Librerías para testing unitario





Índice	
Introducción	3
Lista de marcos de prueba de Python	4
Comparación de herramientas de prueba	5
Behave	6
lettuce	6
¿Cómo elegir el mejor marco de pruebas de Python?	6
Robot	6
Pytest	6
Unittest	7
Doctest	7
Nose2	7
Testify	7
Behave Framework	7
Lettuce	8

Introducción

Como vimos en capítulos anteriores:

- Las pruebas automatizadas son un contexto bien conocido en el mundo de las pruebas. Es donde se ejecutan los planes de prueba utilizando un script en lugar de un humano.
- Python viene con las herramientas y bibliotecas que admiten pruebas automatizadas para su sistema.
- Los casos de prueba de Python son comparativamente fáciles de escribir. Con el mayor uso de Python, los marcos de automatización de pruebas basados en Python también se están volviendo populares.

Lista de marcos de prueba de Python

A continuación, enumeraremos las características de los principales marcos de prueba basados en Python.

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

Comparación de herramientas de prueba

Resumamos rápidamente estos marcos en una breve tabla comparativa:

	Licencia	Parte de	Categoría	Categoría
				Característica
				especial
pytest.warns ()	Advertencia_esper	Afirmar advertencia		
	ada:	con las funciones		
	Expectativa			
	[coincidencia]			
Robot	Software libre	Bibliotecsa de	Test de aceptación	Enfoque de prueba
	(Licencia ASF)	prueba genéricas		basado en palabras
		de Python		clave
pytest	Software libre	Independiente,	Examen de la	Accesorio de clase
	(licencia MIT)	permite conjuntos	unidad	especial y simple
		de prueba		para facilitar las
		compactos		pruebas
unittest	Software libre	Parte de la	Examen de la	Recopilación de
	(licencia MIT)	biblioteca estándar	unidad	pruebas rápida y
		de Python		ejecución de
				pruebas flexible
DOCtest	Software libre	Parte de la	Examen de la	Python Interactive
	(licencia MIT)	biblioteca estándar	unidad	Shell para el
		de Python		símbolo del
				sistema y la
				aplicación inclusiva
nose	Software libre	Lleva funciones de	extensión unittest	Una gran cantidad
	(Licencia BSD)	unittest con		de complementos
		funciones y		
		complementos		
		adicionales		
Testify	Software libre	Lleva	extensión unittest	Mejora del
	(Licencia ASF)	características		descubrimiento de
		unittest y nose con		pruebas
		características y		
		complementos		
		adicionales		

Estos son, hoy por hoy, los marcos de pruebas de Python más populares, pero podríamos incluir en esta lista algunos que podrían volverse populares en el futuro.

Behave

- Behave usa el lenguaje natural para escribir pruebas y trabaja con cadenas Unicode. Utiliza el llamado BDD (desarrollo impulsado por el comportamiento). Es un marco de prueba que también se utiliza para Pruebas de caja negra.
- El directorio Behave contiene archivos de características que tienen un texto sin formato, parece lenguaje natural e Implementaciones de pasos de Python.

lettuce

- lettuce es útil para Pruebas de desarrollo impulsadas por el comportamiento. Hace que el proceso de prueba sea fácil y escalable.
- **lettuce** incluye pasos como:
 - Describir el comportamiento
 - Definición de pasos en Python.
 - Ejecutando el código
 - Modificando código para pasar la prueba.
 - Ejecutando el código modificado.
- Estos pasos se siguen de 3 a 4 veces para que el software esté libre de errores y, por lo tanto, mejore su calidad.

¿Cómo elegir el mejor marco de pruebas de Python?

Comprender las ventajas y limitaciones de cada marco es una mejor manera de elegir correctamente el mejor marco de pruebas para nuestro proyecto.

Robot

Ventajas

- El enfoque de prueba basado en palabras clave ayuda a crear casos de prueba legibles de una manera más fácil.
- Varias API.
- Sintaxis de datos de prueba sencilla.
- Admite pruebas paralelas a través de Selenium Grid.

Limitaciones

- Crear informes HTML personalizados es bastante complicado con Robot.
- Menos soporte para las pruebas en paralelo.
- Requiere Python 2.7.14 y superior.

Pytest

Ventajas

- Admite un conjunto de pruebas compacto.
- No es necesario el depurador ni ningún registro de prueba explícito.
- Múltiples accesorios
- Complementos extensibles
- Creación de pruebas fácil y sencilla.
- Es posible crear casos de prueba con menos errores.

Limitaciones

• No compatible con otros frameworks.

Unittest

Ventajas

- No es necesario ningún módulo adicional.
- Fácil de aprender para probadores a nivel principiante.
- Ejecución de prueba simple y fácil.
- Generación rápida de informes de prueba.

Limitaciones

- El nombre snake_case de Python y el nombre camelCase de JUnit causan un poco de confusión.
- Intención poco clara del código de prueba.
- Requiere una gran cantidad de código repetitivo.

Doctest

Ventajas

- Una buena opción para realizar pequeñas pruebas.
- La documentación de prueba dentro del método también proporciona información adicional sobre cómo funciona el método.

Limitaciones

 Solo compara la salida impresa. Cualquier variación en la salida provocará un fallo en la prueba.

Nose2

Ventajas

 Nose2 admite más configuraciones de prueba que unittest.

- Incluye un conjunto sustancial de complementos activos.
- API diferente de unittest que proporciona más información sobre el error.

Limitaciones

 Al instalar complementos de terceros, debe instalar la herramienta de configuración / distribuir el paquete, ya que Nose2 admite Python 3 pero no complementos de terceros.

Testify

Ventajas

- Fácil de entender y usar.
- Las pruebas unitarias, de integración y del sistema se pueden crear fácilmente.
- Componentes de prueba manejables y reutilizables.
- Agregar nuevas funciones a Testify es fácil.

Limitaciones

 Inicialmente, Testify se desarrolló para reemplazar unittest y Nose, pero el proceso de transición a pytest está activado, por lo que se recomienda a los usuarios que eviten usar Testify para algunos proyectos próximos.

Behave Framework

Ventajas

- Fácil ejecución de todo tipo de casos de prueba.
- Razonamiento y pensamiento detallados
- Claridad de los resultados de QA / Dev.

Limitaciones

Solo admite pruebas de caja negra.

Lettuce

Ventajas

- Lenguaje simple para crear múltiples escenarios de prueba.
- Útil para casos de prueba basados en el comportamiento para pruebas de caja negra.

Limitaciones

 Necesita una fuerte coordinación entre desarrolladores, probadores y partes interesadas.

Puede elegir el marco de prueba de Python más adecuado considerando las ventajas y limitaciones anteriores que ayudarán a desarrollar los criterios adecuados para sus necesidades comerciales.