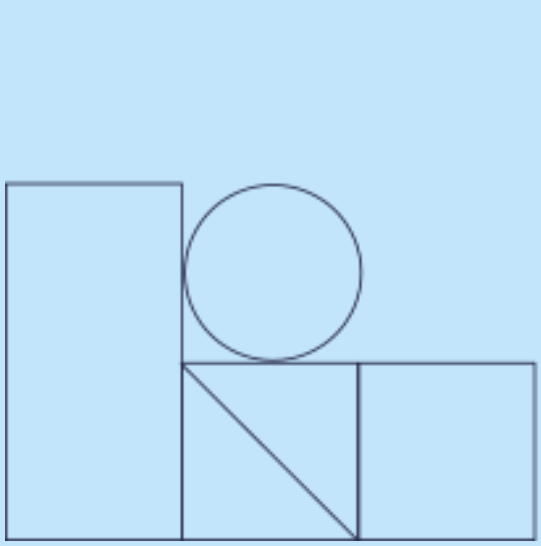
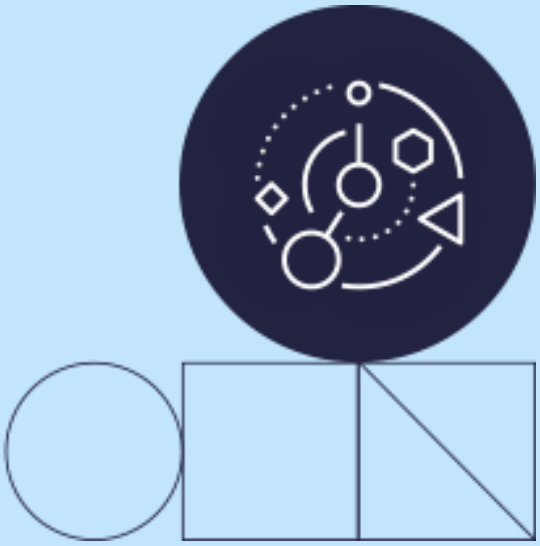
IBM SkillsBuild

IBM

Pruebas con Python

Pruebas unitarias y de integración





Índice

Introducción

De todos los tipos de pruebas que vimos en el tema anterior, nos centraremos en las pruebas unitarias y de integración.

Hoy en día los proyectos de software son más complejos que nunca antes. La industria del software crece a una velocidad altísima y los desarrolladores de software han de estar a la última de novedades y oportunidades en el mundo del software para utilizar la última tecnología de la mejor manera.

Hay muchos tipos de negocio, pero en el desarrollo de software, la calidad es esencial para el éxito. Para garantizar este éxito, la función de testeo es vital.

Ningún buen desarrollador implementa código sin realizar pruebas exhaustivas.

La prueba unitaria de Python es un marco de prueba integrado para probar el código Python. Tiene un corredor de pruebas, lo que nos permite ejecutar las pruebas sin mucho esfuerzo. Entonces, podemos usar el módulo de prueba unitaria para probar sin utilizar los módulos de terceros.

Mientras que las pruebas unitarias corren pruebas sobre partes del código de forma independiente, las pruebas de integración se hacen necesarias cuando queremos probar de una manera sistémica un proyecto como un todo.

Las pruebas de integración ejercitan dos o más partes de una aplicación a la vez, incluidas las interacciones entre las partes, para determinar si funcionan según lo previsto. Este tipo de prueba identifica defectos en las interfaces entre partes dispares de una base de código a medida que se invocan y pasan datos entre sí.

Estos dos enfoques deben usarse juntos, en lugar de hacer solo un enfoque sobre el otro.

Cuando un sistema se prueba de forma exhaustiva por unidad, las pruebas de integración son mucho más fáciles porque muchos de los errores en los componentes individuales ya se habrán encontrado y solucionado.

Es necesario pues que, desde el principio de nuestro desarrollo se realicen tanto pruebas de integración como pruebas unitarias. La combinación de ambas garantiza que el código planteado sea de buena calidad y verdaderamente solucione los problemas para los cuales fue desarrollado.

A medida que aumenta la escala de una base de código, tanto las pruebas unitarias como las de integración permiten a los desarrolladores identificar rápidamente los cambios importantes en su código. Muchas veces, estos cambios importantes no son intencionados y no se conocerán hasta más adelante en el ciclo de desarrollo, posiblemente cuando un usuario final descubre el problema mientras usa el software. Las pruebas automatizadas de unidad e integración aumentan en gran medida la probabilidad de que los errores se encuentren lo antes posible durante el desarrollo para que se puedan solucionar de inmediato.

Pruebas unitarias

Si partimos de la base de que la prueba unitaria es algo así como la "unidad básica" de prueba o testeo en un programa/aplicación, empecemos, pues, por ella para ir poco a poco abordando aspectos un poquito más complejos más adelante.

El módulo unittest de la librería estándar (stdlib) de Python es el encargado de llevar a cabo las pruebas unitarias si no queremos acudir a librerías externas. Se le conoce también como PyUnit, donde el sufijo Unit hace referencia a un conjunto de parámetros construidos exprofeso para ejecutar pruebas unitarias en diferentes lenguajes de programación, y que se conocen como XUnit, donde X es el prefijo que apunta al lenguaje de programación en concreto. En nuestro caso, Py, el prefijo, señala a Python, como fácilmente podemos deducir. Vamos, digo yo.

Esto se relaciona con el proceso de unit testing, que es universal para todos los lenguajes de programación. su recurso es paralelo, o cuando menos, suele serlo, al modelizado de una clase, de cara a probar sobre la marcha la idoneidad de sus métodos antes de pasar a modelizarse la clase siguiente; o bien si esta clase (y, en consecuencia, el módulo que la contiene) experimenta algún tipo de modificación. siempre será conveniente modelizar una clase específica de prueba (unit testing). dentro del propio módulo, al margen o a continuación de la clase o clases que configuran el propio módulo para que puedan trabajar independientemente. dentro del conjunto de estilos de python se considera una buena práctica incluir una clase de prueba por cada módulo del que disponga nuestro proyecto. todo ello genera una suerte de entorno o marco de pruebas, y de ahí que al módulo unittest se le denomine también marco de pruebas.

Este módulo Unittest cuenta con una serie de herramientas que nos permitirán probar nuestro programa o aplicación dentro del mismo código.

Pruebas de integración

Las **pruebas de integración de software** son la herramienta que conjunta cada uno de los módulos de un sistema para comprobar su funcionamiento entre sí. Este tipo de *test* se realizan en las primeras etapas, después de las pruebas unitarias, en las que se analiza un fragmento del código fuente.

Una vez analizadas las unidades por separado, ser debe verificar que los módulos no interfieren con el resto de las funciones.

Imaginemos que una aplicación de correo electrónico está dividida en tres unidades —página de inicio, bandeja de entrada y papelera—. Al ejecutar una prueba de integración el desarrollador debe comprobar que el vínculo entre una unidad y otra es óptimo.

La sesión de inicio debe direccionar correctamente a la bandeja de entrada, así como la eliminación de un email deberá alojarse de inmediato en la papelera de reciclaje. De existir alguna interfaz errónea se suscitarán problemas en el futuro y los usuarios no podrán hacer uso del sistema.

¿Qué errores pueden detectarse con este tipo de pruebas?

Las pruebas unitarias son un primer filtro para la detección de fallos en los sistemas, sin embargo, éstas no identifican su relación con otras interfaces. De ahí la importancia de implementar **pruebas de integración de software**

Entre los problemas más comunes que identifican este tipo de pruebas se encuentran la pérdida de conectividad, el formateo de datos y las respuestas inesperadas.

La detección oportuna de errores puede minimizar el impacto económico y temporal, de manera que los desarrolladores puedan enfocar esfuerzos en otros proyectos y tareas prioritarias.

Niveles de pruebas de integración

Existen dos niveles de pruebas de integración: de componentes y de sistemas. En el primero se evalúan los elementos integrados en un mismo sistema, mientras que en el segundo se verifica la relación entre interfaces de sistemas externos.

Las **pruebas de integración de componentes** suelen ejecutarse primero que las **pruebas de integración de sistemas**, después de verificar las unidades de código de manera individual. Una vez asegurado este paso, es viable realizar los *tests* que involucran interfaces ajenas al sistema original.

Tipos de pruebas de integración

Big Bang

Una prueba de integración *Big Bang* concentra todos los módulos de un sistema para comprobar su funcionamiento en conjunto por lo que, antes de ejecutarse, el desarrollador debe cerciorarse que cada unidad ha sido completada.

Este tipo de test es viable en proyectos pequeños, de lo contrario, se pueden pasar por alto errores significativos.

Ad Hoc

Este término hace referencia al planteamiento de una solución para un problema específico. Para fines de testing software este tipo de prueba de integración puede ejecutarse en cualquier momento, recomendado ampliamente en etapas tempranas, con el objetivo de hallar errores no previstos.

Entre sus ventajas se encuentra la rapidez y poca planificación requerida para su realización. A pesar de ello, las dificultades pueden hallarse en fases posteriores ya que no se requiere documentación para dar cuenta de los errores encontrados.

Top Down

Como su nombre lo indica, las pruebas *top down* (de arriba hacia abajo) inician el análisis de código en los módulos posteriores, en los que se concentra la información central y de manera descendiente conectan con otras interfaces.

En este modelo pueden detectarse errores de mayor relevancia y por la complejidad de su composición las mejoras pueden implicar más tiempo.

Down Top

Contrario al modelo anterior, la prueba de integración *down top* parte de las interfaces inferiores y continúa de manera ascendente. En este caso los problemas son más fáciles de detectar, al igual que las mejoras a realizar.

Su desventaja es que los módulos complejos se sitúan al final de la prueba y la entrega del producto final puede tomar más tiempo del estimado.

Hybrid

Las pruebas de integración híbridas —también conocidas como sándwich— incluyen las prácticas de los dos modelos anteriores —*top down* y *down top*—. El desarrollador puede elegir los módulos posteriores o inferiores simultáneamente con el objetivo de hallar errores en menor tiempo.

Pese a ello, requiere de equipo altamente capacitado para la detección de errores de manera precisa y oportuna. Este tipo de prueba es recomendable en sistemas operativos de mayor complejidad.