Microprocesadores y microcontroladores

Christopher Andrés Jiménez Loría

2019067477

Tarea 1

GitHub, Pytest y Flake 8

Preguntas Teóricas

1) ¿Explique la principal utilidad de git como herramienta de desarrollo de código?

El software de código abierto Git, en que está basado GitHub, permite como característica principal el trabajo colaborativo. Funciona principalmente para almacenar, compartir y trabajar junto con otros usuarios para escribir código.

Permite presentar o compartir trabajo, seguir cambios a largo del tiempo, permitir revisiones y sugerencias a código y colaborar en proyecto compartido sin afectar el trabajo de otros colaboradores antes de que los cambios sean integrados.

2) ¿Qué es un commit?

Commit o revisión es un cambio individual a un archivo o grupo de archivos. Cuando se hace un commit para guardar el trabajo, se crea una identificación hash para guardar la memoria de los cambios específicos. Los commits traen un mensaje con una breve descripción de los cambios.

3) ¿Qué es un branch?

Un branch es una versión alternativa de un repositorio. Está contenido dentro del repositorio, pero no afecta la Branch o rama principal, por lo que permite trabajar libremente sin afectar la versión actual u oficial del trabajo. Cuando se hayan hecho los cambios, la Branch se puede reunir a la Branch principal para publicar los cambios.

4) En el contexto de github. ¿Qué es un Pull Request?

Las solicitudes de cambios son propuestas para combinar los cambios de código en un proyecto. Un "pull request" es la característica de colaboración fundamental de GitHub, que permite discutir y revisar los cambios antes de fusionarlos. Esto ayuda a los equipos a trabajar juntos, detectar problemas al principio y mantener la calidad del código.

5) ¿Si un usuario quiere “Actualizar su repositorio contra el Branch master” que debe de hacer?

6) ¿Bajo qué condiciones una herramienta como Git necesita apoyo de un humano para saber cómo integrar cambios locales con cambios remotos?

Git necesita intervención humana para integrar cambios locales a remotos en diferentes casos:

- Cuando hay conflictos de fusión (merge conflicts) porque los cambios locales y remotos modificaron las mismas líneas de código.

- Cambios locales sin *commit* que serían sobreescritos al integrar. Se necesita hacer *commit* o *stash* antes de integrar.

- El historial remoto se ha reescrito (push forzado) y la rama local divirgió. Se requiere como reconciliarlo con rebase o merge con conflicto.

- Un archivo fue eliminado por un lado y modificado en otro.

7) ¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?

Una prueba unitaria es una prueba automatizada que verifica el comportamiento correcto de la unidad individuales de código de forma aislada del resto del sistema. Su objetivo es asegurar que cada parte individual funcione según lo esperado. En GitHub, se ejecutan frecuentemente (al hacer *commit*) y suelen usar frameworks como JUnit, pytest o NUnit.

8) Bajo el contexto de pytest. ¿Cuál es la utilidad de un "assert"?

Pytest es un *framework* para Python que ofrece la recolección automática de los *tests*, aserciones simples, soporte para *fixtures*, *debugeo* y mucho más… no te preocupes si algunas de estas palabras no te hacen mucho sentido; intentaré aclararlos más adelante a lo largo de este post.

Assert es un comando para verificar condiciones, tomando ventaja de la introspección de aserción avanzada de Pytest que provee mensajes de errores bastante detallados sin necesidad de método especiales de aserciones.

9) ¿Qué es Flake 8?

Flake8 es una herramienta de linting y análisis estático de código para Python. Linting es un análisis automático errores para código fuente y enforzador de estándares de código. Básicamente, es un asistente que revisa tu código en busca de errores de programación, malas prácticas y problemas de estilo y legibilidad sin necesidad de ejecutar el programa.

Incluye Pyflakes, pycodestyle y McCabe.

10) Comente sobre la utilidad de la aleatorización en pruebas de código.

La aleatorización en pruebas de código consiste en introducir datos o condiciones aleatorias durante la ejecución de los tests. Su utilidad principal es descubrir errores imprevistos que no se detectan con pruebas de valores fijos.

Es útil para los siguientes casos:

- Simular entornos reales, donde las entradas son variadas e impredecibles.
- Ampliar la cobertura de casos, explorando combinaciones que el programador no anticipó.
- Detectar fallos de robustez, como crashes ante entradas malformadas (fuzzing) o problemas de concurrencia.
- Validar algoritmos probabilísticos, comprobando que su comportamiento estadístico sea el esperado.