1) ¿Qué es git y su relación con GitHub?

Git es un sistema de control de versiones distribuido. Permite llevar un historial de cambios en el código, trabajar de forma colaborativa, crear ramas, revertir cambios y fusionar trabajo de distintos colaboradores sin sobrescribir el trabajo de otros. **GitHub** es una plataforma en la nube que aloja repositorios Git y añade funciones adicionales como control de acceso, seguimiento de issues, Pull Requests, integración continua, automatización y gestión de proyectos.

2) ¿Qué es un branch? ¿Qué es un fork?

- ***Branch (rama)***: Línea independiente de desarrollo dentro de un mismo repositorio. Permite aislar cambios (por ejemplo, para desarrollar una nueva funcionalidad) sin afectar directamente la rama principal (main o master).
- ***Fork***: Copia de un repositorio alojado en GitHub hacia tu cuenta. Es independiente del original, aunque se puede sincronizar con él. Generalmente se usa para contribuir a proyectos donde no se tiene acceso directo de escritura.

3) En el contexto de GitHub, ¿Qué es un Pull Request?

Es una solicitud para que los cambios realizados en una rama (puede estar en el mismo repositorio o en un fork) sean revisados y fusionados en otra rama (normalmente main). Permite discusión, revisión de código, ejecución de pruebas automáticas y control de calidad antes de integrar cambios.

4) ¿Qué es un commit?

Es una confirmación de cambios en Git. Un commit guarda el estado de los archivos modificados en un punto específico del historial, junto con un mensaje descriptivo y metadatos (autor, fecha, hash único). Los commits son la unidad básica del historial de versiones.

5) ¿Qué es un "merge conflict" o "rebase conflict"?

- **Merge conflict**: Ocurre cuando Git no puede fusionar automáticamente cambios entre dos ramas porque existen modificaciones incompatibles en las mismas líneas de código o en la misma estructura. El usuario debe resolver manualmente las diferencias.
- **Rebase conflict**: Similar al merge conflict, pero ocurre durante un **rebase** (reestructuración del historial), donde los commits se "reaplican" sobre una nueva base y entran en conflicto con cambios ya existentes.

En ambos casos, Git pausa la operación hasta que el desarrollador resuelva los conflictos y confirme la resolución.

6) ¿Qué es una Prueba Unitaria o Unittest?

Es una prueba automatizada que valida el comportamiento de una pequeña unidad de código (por ejemplo, una función o método) de forma aislada. Su objetivo es asegurar que cada unidad funcione correctamente de manera independiente y detectar errores en etapas tempranas del desarrollo.

7) Bajo el contexto de pytest, ¿Cuál es la utilidad de un "assert"?

En ***pytest***, ***assert*** se usa para verificar que una condición es verdadera durante una prueba. Si la condición es falsa, la prueba falla y pytest reporta el fallo con información del valor real y esperado.

8) ¿Qué son GitHub Actions y su utilidad para el desarrollo continuo de código?

GitHub Actions es una herramienta de automatización e integración continua (CI/CD) integrada en GitHub. Permite ejecutar flujos de trabajo (*workflows*) automáticamente en respuesta a eventos (push, pull request, creación de release, etc.). Se usa para: ejecutar pruebas automáticas. desplegar aplicaciones, ejecutar análisis de calidad de código, publicar paquetes.

9) ¿Qué es Flake8?

Es una herramienta de *linter* para Python que analiza el código para verificar el cumplimiento de las normas de estilo (PEP8), detectar errores sintácticos y advertencias. Ayuda a mantener un código limpio y consistente.

10) Explique la funcionalidad de parametrización de pytest.

La parametrización en *pytest* permite ejecutar la misma prueba con diferentes conjuntos de datos de forma automática, sin duplicar código. Esto mejora la cobertura de pruebas y reduce redundancia.