Preguntas Teóricas

1. ¿Explique que es git y su relación con github?

- **Git** es un sistema de control de versiones distribuido que permite a los desarrolladores llevar un historial de cambios de su código, trabajar en paralelo y revertir modificaciones si es necesario.
- **GitHub** es una plataforma en la nube que utiliza Git como motor principal para almacenar repositorios, colaborar entre desarrolladores y gestionar proyectos. Git funciona localmente, mientras que GitHub agrega funcionalidades sociales y de colaboración en línea (pull requests, issues, acciones automáticas, etc.).

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

- **Branch (rama)**: Una línea de desarrollo independiente dentro de un repositorio. Permite trabajar en nuevas características o correcciones sin afectar la rama principal (main o master).
- **Fork**: Una copia completa de un repositorio en tu cuenta de GitHub. Sirve para proponer cambios a un proyecto en el que no tienes permisos directos o para crear tu propia versión independiente.

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

Es una solicitud para que los cambios que hiciste en una rama (o en un fork) se revisen y, si se aprueban, se integren en otra rama del repositorio original. Permite discusión, revisión de código y ejecución de pruebas antes de fusionar.

4. ¿Qué es un commit?

Es un registro de un conjunto de cambios en el código, acompañado de un mensaje que describe lo que se hizo. Cada commit tiene un identificador único y forma parte del historial del repositorio.

5. Explique que es un “merge conflict” o “rebase conflict” en el contexto de tratar de hacer merge a un Pull Request o de completar una operación git rebase.

   - **Merge conflict** ocurre cuando Git no puede combinar automáticamente cambios de dos ramas porque se modificaron las mismas líneas en los mismos archivos de forma diferente.
   - Un **rebase conflict** es lo mismo, pero sucede durante una operación de *rebase* (reaplicar commits sobre una nueva base).

     En ambos casos, el desarrollador debe resolver manualmente las diferencias para continuar.

6. ¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?

   Es una prueba automatizada que verifica que una unidad mínima de código (generalmente una función o método) funcione como se espera. Ayuda a detectar errores pronto y asegura que el código siga comportándose correctamente tras cambios.

7. Bajo el contexto de pytest. ¿Cuál es la utilidad de un “assert”?

   En pytest, assert se usa para verificar que una condición sea verdadera. Si la condición falla, la prueba se marca como fallida y pytest muestra un reporte detallado.

   Ejemplo:

   ```
   def test_suma():
     assert 2 + 2 == 4
   ```

8. ¿Explique que son github-actions y su utilidad para el desarrollo continuo de código?

   Son una herramienta de integración continua (CI) y entrega continua (CD) dentro de GitHub.
   Permiten automatizar flujos de trabajo como:

   - Ejecutar pruebas unitarias en cada commit.
   - Revisar formato y estilo del código.
   - Desplegar el proyecto automáticamente cuando se apruebe un PR.

9. ¿Qué es Flake 8?

Es una herramienta para analizar código Python y verificar:

- Errores de sintaxis.
- Estilo de acuerdo a PEP 8.
- Problemas comunes que pueden causar errores.

Sirve para mantener el código limpio, uniforme y legible.

10. Explique la funcionalidad de parametrización de pytest.

Permite ejecutar la misma prueba varias veces con diferentes datos de entrada sin tener que duplicar código.

Ejemplo:

```
import pytest
@pytest.mark.parametrize("a,b,resultado", [
   (2, 3, 5),
   (10, 5, 15),
   (-1, 1, 0)
])
def test_suma(a, b, resultado):
   assert a + b == resultado
```

Aquí, pytest ejecutará test_suma tres veces con valores distintos.