Respuestas de la tarea 1:

Pregunta 1

Utilidad de Git como herramienta de desarrollo de código: Git es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software.

Sus principales utilidades son:

- Seguimiento de cambios: Permite rastrear modificaciones en archivos a lo largo del tiempo.
- Colaboración: Facilita el trabajo en equipo al permitir que varios desarrolladores trabajen en el mismo proyecto.
- Versionado: Guarda versiones históricas del código, lo que ayuda a revertir cambios o comparar versiones.
- Branches: Permite trabajar en paralelo en diferentes características sin afectar la rama principal (master/main).

Pregunta 2

¿Qué es un branch?: Un Branch o rama en Git es una línea de desarrollo independiente que se deriva de la rama principal (generalmente llamada master o main). Los branches permiten trabajar en nuevas características o correcciones sin afectar el código base. Una vez que se completa el trabajo en un branch, se puede fusionar con la rama principal.

Pregunta 3

Pull Request (PR) en GitHub: Un Pull Request es una solicitud para fusionar cambios de una rama (generalmente una rama de características) en otra (generalmente la rama principal). Los PRs son comunes en plataformas como GitHub. Los revisores pueden comentar, aprobar o solicitar cambios antes de que los cambios se fusionen oficialmente.

Pregunta 4

¿Qué es un commit?: Un commit es una confirmación de cambios en Git. Representa un conjunto de modificaciones en archivos y se asocia con un mensaje descriptivo. Los commits son la unidad básica de seguimiento en Git.

Pregunta 5

git fetch: Esta operación descarga los cambios desde el repositorio remoto al repositorio local, pero no los integra en tu rama actual. Es útil para obtener actualizaciones del repositorio remoto sin alterar tu trabajo actual.

Los cambios se almacenan en las ramas remotas (por ejemplo, origin/master), permitiéndote revisar los cambios antes de fusionarlos.

git rebase origin/master: Esta operación toma los cambios de la rama origin/master y los aplica sobre tu rama actual.

Primero, Git aplica los commits de origin/master en tu rama actual, y luego aplica tus commits locales encima de esos cambios. El resultado es un historial de commits más limpio y lineal, ya que rebase reescribe el historial de commits.

Pregunta 6

Merge Conflict (conflicto de fusión): Un merge conflict ocurre cuando Git no puede fusionar automáticamente dos ramas debido a cambios conflictivos en el mismo archivo o líneas de código. Para resolverlo, debes editar manualmente los archivos afectados, resolver los conflictos y luego confirmar los cambios.

Un rebase conflict es similar, pero ocurre durante una operación de rebase. Cuando haces git rebase, Git intenta aplicar tus commits locales sobre la nueva base (por ejemplo, origin/master). Si hay cambios incompatibles entre tus commits y los de la nueva base, se producirá un conflicto.

En ambos casos, Git te mostrará los archivos en conflicto y te permitirá editarlos para resolver las diferencias. Una vez resueltos, puedes continuar con la operación de merge o rebase.

Pregunta 7

Prueba Unitaria o Unittest: Una prueba unitaria es una técnica de prueba en la que se verifica el funcionamiento correcto de una parte específica del código (como una función o método) de manera aislada. Ayuda a detectar errores temprano y garantiza que las partes individuales del software funcionen correctamente.

Pregunta 8

Utilidad del “assert” en pytest: En el contexto de pytest, el uso de assert permite verificar que una condición sea verdadera. Si la condición es falsa, la prueba falla. Es una forma efectiva de verificar resultados esperados en las pruebas unitarias.

Pregunta 9

Errores de formato detectables con Flake8: Flake8 es una herramienta de análisis estático para Python. Detecta errores de formato, como:

Indentación incorrecta:

- Espacios o tabulaciones mal colocados.
- Líneas demasiado largas: Exceder el límite de caracteres por línea.
- Espacios en blanco innecesarios: Espacios al final de las líneas.

Pregunta 10

Funcionalidad de parametrización en pytest: La parametrización en pytest permite ejecutar una misma prueba con diferentes conjuntos de datos de entrada. Es útil para probar una función o método con múltiples casos de prueba sin duplicar código.

Referencias:

Atlassian, (2024*) Usa Git correctamente*, extraído de https://www.atlassian.com/es/git/glossary#commands

Documentación oficial de Github, extraído de https://docs.github.com/es

Documentación oficial de Pytest, extraído de https://docs.pytest.org/en/latest/contents.html

Documentación oficial de Flake 8. extraído de https://flake8.pycqa.org/en/latest/index.html