Instituto Tecnológico de Costa Rica

Área de Ingeniería Mecatrónica

Curso: MT-7003 Microprocesadores y microcontroladores

Profesor: Dr. -Ing. Carlos Adrián Salazar García

Tarea 1

GitHub, Pytest y Flake 8

Preguntas Teóricas

Integrantes:

Aarón Moises Granados Fonseca	2016004670
Katharina María Monge Calvo	2016254698
Francisco Elías Pérez Agüero	2021117284

Verano 2025 – 2026

Preguntas Teóricas

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

La principal utilidad de Git es el control de versiones del código fuente. Permite registrar y administrar el historial de cambios de un proyecto, facilitando la colaboración entre múltiples desarrolladores, la recuperación de versiones anteriores, la experimentación segura mediante ramas y la trazabilidad de quién realizó cada cambio y por qué. Git optimiza el trabajo distribuido y reduce el riesgo de pérdida o sobrescritura de código.

2) Explique la diferencia entre git y github

Git es un sistema de control de versiones distribuido que se ejecuta localmente y gestiona el historial del código. GitHub, en cambio, es una plataforma en la nube que aloja repositorios Git y añade servicios adicionales como gestión de issues, pull requests, revisión de código, control de acceso y automatización (CI/CD). Git funciona sin GitHub; GitHub utiliza Git como tecnología base.

3) ¿Qué es un branch?

Un branch (rama) es una línea independiente de desarrollo dentro de un repositorio Git. Permite trabajar en nuevas funcionalidades, correcciones o experimentos sin afectar la rama principal del proyecto. Las ramas facilitan el desarrollo paralelo y luego pueden integrarse nuevamente al código principal mediante merge o rebase.

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

Un Pull Request es una solicitud para integrar cambios de una rama a otra dentro de un repositorio en GitHub. Su función principal es permitir la revisión del código, la discusión técnica y la validación de cambios antes de que estos sean incorporados a ramas importantes como main o develop.

5) ¿Qué es un commit?

Un commit es una instantánea del estado del proyecto en un momento específico. Representa un conjunto de cambios confirmados en el repositorio, acompañado de un mensaje descriptivo. Cada commit tiene un identificador único (hash) que permite rastrear su contenido y posición en el historial.

6) Describa lo que sucede al ejectuar la siguiente operación: “git rebase main”.

Al ejecutar git rebase main, Git toma los commits de la rama actual y los reaplica, uno por uno, sobre la punta de la rama main. Esto reescribe el historial de la rama actual, generando una secuencia de commits más lineal y limpia, como si los cambios se hubieran desarrollado directamente a partir de la versión más reciente de main.

7) Explique que es un “merge conflict” y como lo resolvería.

Un merge conflict ocurre cuando Git no puede resolver automáticamente diferencias entre dos ramas, generalmente porque ambas modificaron las mismas líneas de un archivo. Para resolverlo, se debe editar manualmente el archivo afectado, decidir qué cambios conservar o combinar, eliminar los marcadores de conflicto y luego confirmar la resolución mediante un nuevo commit.

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

Una Prueba Unitaria (Unit Test) es una verificación automatizada que evalúa el comportamiento correcto de una unidad mínima de código, como una función o método, de forma aislada. Su objetivo es detectar errores tempranamente, asegurar que los cambios no rompan funcionalidades existentes y facilitar el mantenimiento del software.

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

En pytest, un assert se utiliza para verificar que una condición sea verdadera durante la ejecución de una prueba. Si la condición falla, la prueba se marca como fallida. Los assert permiten validar resultados esperados, estados del sistema o excepciones, siendo el mecanismo central de validación en las pruebas.

10) Mencione y explique tres errores de formato detectables con

Flake8 Tres errores de formato detectables con Flake8 son:

- Líneas que exceden la longitud máxima permitida (por defecto 79 caracteres), lo cual afecta la legibilidad del código.
- Uso incorrecto de la indentación, como espacios o niveles inconsistentes, que puede generar confusión o errores lógicos.
- Importaciones no utilizadas, que ensucian el código y pueden indicar errores de diseño o refactorizaciones incompletas.