INSTITUTO TECNOLÓGICO DE COSTA RICA

Microprocesadores y Microcontroladores ***MT-7003***

TAREA 1: Programming Tools

Estudiantes: Marco Vinicio Rodríguez Barboza **Carné:** 2018092536

Carlos Armando Valverde Díaz **Carné**: 2018165689

Preguntas Teóricas:

1. *¿Qué es git?*

Git es un sistema de control de versiones distribuidas gratuito y de *open source*, tipo Source Code Management (SCM), diseñado para manejar con velocidad y eficiencia todo tipo de proyectos, desde pequeños hasta muy grandes.

La característica que distingue a Git de casi todos los demás SCM es su modelo de ramificación o *branching*. Git permite tener múltiples branches locales que pueden ser completamente independientes entre sí. La creación, fusión y eliminación de esas líneas de desarrollo lleva unos segundos, y permite tener branches para cada nueva característica de un proyecto, en donde se puede alternar sin problemas entre ellas, trabajar propiamente sobre ellas y luego eliminar cada branch cuando esa característica se fusione en su línea principal.

2. *¿Qué es github?*

GitHub es una plataforma de desarrollo inspirada en la forma de trabajo colectiva, propiamente, es un servicio de repositorio de alojamiento (hosting) basado en Git. Aquí se puede alojar y revisar código, provee acceso de control y características de colaboración, como wikis y herramientas de administración básicas, con el objetivo de administrar proyectos y crear software de manera colaborativa. En contraposición con Git, que es una herramienta de líneas de comando, Github tiene una interfaz gráfica basada en la web.

3. *¿Qué es un Branch?*

La gracia de Git está en hacer cambios en un proyecto de forma segura a un lado y fusionarlos nuevamente en el proyecto original una vez que hayan demostrado ser correctos. Git administra la línea de tiempo de versiones de código es mediante el uso de branches (ramas). Una *branch* es un conjunto único de cambios de código con un nombre único, y cada repositorio puede tener una o más ramas. La branch principal es aquella en la que todos los cambios finalmente se fusionan, y se llama *master branch*. Esta es la versión de trabajo oficial del proyecto, y es la que ve cuando visita el repositorio en Github.

4. *¿Qué es un Commit?*

Git guarda cada versión de un proyecto como un snapshot del código en el momento en que se hace un *commit*, de forma que se crea una línea de tiempo de las versiones de un proyecto a medida que se avanza, para que se pueda retroceder a una versión anterior en caso de que ocurra un problema. Por consiguiente, un *commit* hace las veces de una función de guardado de la versión más reciente del proyecto. A cada commit se le asocia un *commit message*, que describe y explica por qué se hicieron cambios.

5. *¿Qué es la operación cherry-pick?*

El comando git cherry-pick commit aplica los cambios de un cierto commit en la Branch actual. Es típicamente usado para introducir commits particulares de un Branch en otro Branch. Es decir, cherry-pick no altera el historial de un repositorio, sino que se añade al historial. Para poder usarlo, se necesita resolver los conflictos para poder aplicar los cambios de un commit dado.

6. *¿Qué hace el comando git checkout?*

Aplicar un *checkout* es el acto de cambiar entre diferentes versiones de una entidad objetivo. El comando git checkout permite cambiar la Branch activa, cuando se quiere cambiar a una Branch diferente para convertirla en la nueva HEAD Branch. A su vez, se puede usar cuando se quiere restaurar una versión histórica de un archivo dado, para resetearlos a versiones más tempranas sin afectar el proyecto.

7. *¿Qué hace el comando git stash?*

El comando *git stash* almacena temporalmente (propiamente, guarda en un *stash*) los cambios efectuado en el código en el se está trabajando para que se pueda trabajar en otra cosa y, más tarde, regresar y volver a aplicar los cambios más tarde. Guardar los cambios en *stashes* resulta práctico si se tiene que cambiar rápidamente de contexto en medio de un cambio en el código, y no se tiene todo listo para confirmar los cambios.

8. *Compare las operaciones git fetch y git pull.*

Cuando se trabaja en un proyecto en conjunto, es probable que un colaborador trabaje sobre un clon o duplicado del repositorio original. Entonces, para mantenerlo actualizado sobre cualquier cambio que se aplique al original, se debe aplicar comandos como git fetch y git pull para traerlos al clonado.

`git fetch`	`git pull`
Le dice al git local que busque la meta-data más reciente del repositorio original, sin hacer ninguna transferencia de archivos. Lo que hace es revisar si hay cambios disponibles.	Hace lo mismo que el git fetch, pero SÍ copia los cambios del repositorio original.

9. *¿Qué hace el comando git reset ~HEAD?*

Cuando se trabaja en un Proyecto compartido, es común que los cambios que se realizan al crear branches, añadir archivos o aplicar commits no sean correctos. Por ello, si en algún momento se modificaron archivos, añadieron o borraron muchas líneas de los archivos y se quiere volver al pasado, se pueden deshacer los cambios y volver a los archivos pasados, por medio de la técnica de “reset to HEAD”.

El propósito del comando git reset es mover la HEAD actual a un cierto commit especificado. Hay de dos tipos, el --hard y el --soft. El git reset --hard se usa para resetear archivos del index y del directorio de trabajo, alterándolos en el proceso; mientras que el git reset --soft no los altera.

10. ¿Qué es Pytest?

Pytest es un framework, o más específicamente, un módulo de python que permite realizar testeos a un código de Python por medio de funciones de prueba, al realizar casos que acierten o fallos controlados; de manera que los métodos retornen los resultados esperados. Debido a la introspección detallada de aserción de Pytest, solo se utilizan declaraciones de *assert* simples.

11. Bajo el contexto de Pytest, ¿qué es un "assert"?

Un *assert* es una comprobación. Funciona llamando al método con una serie de parámetros cuyo resultado sea conocido, luego compara el resultado esperado con lo que efectivamente retorna el método, y posteriormente informa en caso de que el resultado no sea el esperado.

12. ¿Qué es Flake 8?

Flake8 es una herramienta de "linting", que contrasta el código con otros códigos estándar para corroborar si su sintaxis es la correcta, así como su formato. Su fin principal es hacer un código más fácil de leer al hacerlo más ordenado y dotándolo de una sintaxis más eficiente.