Escuela de Ingeniería Mecatrónica

Microprocesadores y Microcontroladores

Tarea 1: Programming Tools

Profesor:

Rodolfo Jose Piedra Camacho

Estudiantes:

Josué Daniel Morera Ramírez - 2020316765

Alexander Solís Quesada - 2020277449

Grupo 1

II Semestre, 2023

Preguntas Teóricas (20 pts, 2pts c/u)

1) **Explique la principal utilidad de git como herramienta de desarrollo de código.**

Git es un sistema de control de versiones distribuido de código abierto; por ende, su principal utilidad es permitir realizar un seguimiento de los cambios en los archivos correspondientes a un proyecto y de esta manera permitir la colaboración de múltiples personas en el desarrollo de código de manera segura y sin temor a perder información.

2) **¿Qué es un branch?**

Un Branch o rama es una línea de desarrollo creada a partir de una versión específica del repositorio. Trabajar en una rama permite trabajar sobre una versión base e implementar características que no interfieran con la rama principal (también conocida como "master"). Esta funcionalidad da paso a poder experimentar con algún código prototipo sin correr el riesgo de romper el código original y también a que varias personas trabajen de manera paralela sin interferir el uno con el otro.

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

Un Pull Request es una solicitud para proponer que los cambios realizados en una rama secundaria del proyecto pasen a formar parte del master. Al realizar un Pull Request, se le notifica a los colaboradores del proyecto para que puedan revisar el código y considerar la aprobación del mismo para formar parte del master.

4) **¿Qué es un commit?**

Un commit es la manera que tienen los colaboradores de agrupar los cambios realizados en cierto momento. De esta manera se les permite a los usuarios comentar sobre los cambios que han realizado y también le permite al equipo tener un historial de los archivos en forma de versiones para promover la trazabilidad del desarrollo.

5) **Describa lo que sucede al ejecutar las siguientes operaciones: "git fetch" y "git rebase origin/master"**

Git fetch: este es un comando en git que permite obtener los cambios más recientes de un repositorio remoto sin aplicar dichos cambios en el trabajo que se está realizando. Al realizar un git fetch se descargan los commits, branches y cualquier otro elemento que haya sido creado o modificado desde la última vez que se sincronizó el usuario con el repositorio local.

Git rebase origin/master: estes un comando que se utiliza para reorganizar la rama de trabajo actual en relación a la rama master. Esto facilita la incorporación de cambios en la rama master a la rama secundaria que se esté trabajando. Además, este comando facilita que el equipo de trabajo mantenga un historial más limpio y lineal al poder sincronizar de manera más sencilla todas las ramas secundarias en función a los cambios que puedan ocurrir en la rama master.

6) **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.**

Ambos términos corresponden a problemas que ocurren cuando se intenta combinar cambios en Git mediante un "merge" o "rebase" y existen conflictos entre las versiones del código que se intenta combinar. Esto sucede cuando dos o más personas han realizado cambios en el mismo archivo o en las mismas líneas de código y estas modificaciones entran en conflicto.

Cuando se realiza un "merge" en Git, se busca fucionar los cambios de una rama en otra. Un "merge conflict" ocurre cuando hay cambios conflictivos entre la rama de destino y la rama que se trata de fusionar.

En el caso del "rebase", se están moviendo los cambios de una rama a otra y cambiando el punto de inicio de la rama. Un "rebase conflict" ocurre cuando hay cambios conflictivos entre los commits que que se tratan de aplicar durante el rebase y los commits existentes en la rama de destino.

En la mayoría de los casos donde preservar los commit resulta de mucha importancia se recomienda utilizar merge, ya que permite observar las ramificaciones y cronología de desarrollo, cosa que el rebase no lo permite. El rebase unifica las ramas perdiendo el historial de los commit y el merge no.

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

Partiendo de que una unidad se refiere a la parte más pequeña y manejable de un programa que puede ser probada de manera independiente, las pruebas unitarias son un conjunto de casos de prueba diseñados para verificar que cada “unidad” o componente de nuestro código funciona como se espera. Por lo general, una unidad suele ser una función, método o procedimiento.

El objetivo de estas pruebas es asegurarse de que cada unidad de código funcione correctamente y produzca los resultados esperados. Las pruebas unitarias tienen varias ventajas, entre ellas, permite la detección temprana de errores, facilita el mantenimiento, fomenta el diseño modular y mejora la calidad general del código.

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

La palabra assert en Python se refiere a un enunciado que verifica ciertas suposiciones sobre nuestro código. Si la suposición no es cierta, la afirmación falla y se genera una excepción.

El uso de assert y pruebas unitarias en Python es fundamental para garantizar la calidad de nuestro código. Con assert podemos verificar suposiciones importantes y asegurarnos de que se cumplan, mientras que las pruebas unitarias nos permiten verificar que cada componente de nuestro código funciona correctamente.

9) ¿Qué es un Flake 8?

Flake8 es un gran conjunto de herramientas para verificar su código fuente contra el PEP8, errores de programación (como "library imported but unused" y "Undefined name") y para verificar la complejidad ciclomática.

Dentro de las librerías que contiene están:

PyFlakes: Es un verificador de errores que analiza el código fuente de Python en busca de problemas como referencias a variables no definidas, importaciones no utilizadas y otros errores de sintaxis y semántica.

pycodestyle (anteriormente conocido como pep8): Es una herramienta que verifica el cumplimiento de las convenciones de estilo definidas en PEP 8 (Python Enhancement Proposal 8), que es la guía de estilo de código para Python.

McCabe: Calcula la complejidad ciclomática de las funciones y métodos en el código Python. La complejidad ciclomática es una medida de la complejidad estructural de un programa y puede ayudar a identificar funciones que pueden ser difíciles de mantener o comprender.

10) Explique la funcionalidad de parametrización de pytest.

La parametrización de Pytest es una función que le permite ejecutar una sola función de prueba contra múltiples conjuntos de entradas de datos. Pytest luego generará dinámicamente pruebas para cada uno de los valores de entrada.

Los beneficios son:

- Se evita tener que hacer a mano cada prueba.
- Podemos apilar la parametrización para crear una matriz de pruebas
- Proporciona una mejor visibilidad del resultado de probar todas las entradas.

Referencias.

1. S. Chacon y B. Straub, *Pro Git*. 2009. doi: 10.1007/978-1-4302-1834-0.
2. ツ M. C. P., «DIFERENCIA ENTRE GIT REBASE y GIT MERGE , WORKSHOP DE GIT .», *Medium*, 18 de mayo de 2018. [En línea]. Disponible en: https://medium.com/@MiguelCasas/diferencia-entre-git-rebase-y-git-merge-workshop-de-git-8622dedde2d7#:~:text=DIFERENCIAS%20DE%20REBASE%20Y%20MERGE&text=El%20rebase%20unifica%20las%20ramas,el%20gr%C3%A1fico%20de%20las%20ramas.
3. M. Collazos, «Probando una aplicación web en Python -Parte 2: Pruebas unitarias», *Medium*, 6 de diciembre de 2021. [En línea]. Disponible en: https://medium.com/contraslashsas/probando-una-aplicaci%C3%B3n-web-en-python-parte-2-pruebas-unitarias-b005289c0c77
4. Losapuntesde, «Uso de assert y pruebas unitarias en Python: asegura la calidad de tu código», *Apuntes de Programador*, 20 de julio de 2023. [En línea]. Disponible en:

https://apuntes.de/python/uso-de-assert-y-pruebas-unitarias-en-python-asegura-la-calidad-de-tu-codigo/#gsc.tab=0
5. G. A. Diaz, «Código de alta calidad en Python: Linters | Medium», *Medium*, 14 de diciembre de 2021. [En línea]. Disponible en: https://medium.com/@gonzaloandres.diaz/escribiendo-codigo-de-alta-calidad-en-python-parte-2-linters-64ffd8d2df91
6. R. Donato, «Dynamically generating tests with PyTest parametrization», *Packet Coders*, mar. 2023, [En línea]. Disponible en: https://www.packetcoders.io/dynamically-generating-tests-with-pytest-parametrization/#:~:text=Pytest%20parametrization%20is%20a%20feature,have%20to%20handcraft%20each%20test.