

Tarea 1

GitHub, Pytest y Flake 8

Link repositorio: <https://github.com/wendygomezr/Tarea1BeiruteGomez>

Preguntas Teóricas

1. ¿Diferencie la herramienta Git de Github?

Git es un sistema de control de versiones abierto y gratuito diseñado para gestionar el historial de código fuente de un proyecto de desarrollo. Donde un control de versiones es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

Por otro lado, Github es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores. En general, permite trabajar en colaboración con otras personas de todo el mundo, planificar proyectos y realizar un seguimiento del trabajo. [2]

En resumen, Git es una herramienta de control de versiones que puede mientras que GitHub es una plataforma basada en la nube construida alrededor de la herramienta Git.

2. ¿Qué es un branch?

Un branch son ramificaciones, significa que se ha tomado la rama principal de desarrollo (master) de un proyecto y a partir de ahí se ha continuado trabajando sin seguir la rama principal de desarrollo[1]. Es decir, son una copia temporal con la que se trabaja en paralelo al proyecto a desarrollar para implementar o probar cambios.

3. ¿Qué es un commit?

Un "commit" es la acción de guardar o subir archivos a un repositorio remoto o local.

Cuando se realizan algunos cambios y confirmar instantáneas de esos cambios en el repositorio, se está realizando un commit. Cada archivo de un repositorio puede tener dos estados: rastreados y sin rastrear. Los archivos rastreados son todos aquellos archivos que estaban en la última instantánea del proyecto; pueden ser archivos sin modificar, modificados o preparados. Los

archivos sin rastrear son todos los demás - cualquier otro archivo en el directorio de trabajo que no estaba en tu última instantánea (commit) y que no está en el área de preparación. [1]

4. ¿Qué es la operación cherry-pick?

Cherry-pick es un comando de git que permite que las confirmaciones arbitrarias de Git se elijan por referencia y se añaden al actual HEAD de trabajo. La ejecución de cherry-pick es el acto de elegir una confirmación de una rama y aplicarla a otra.

5. ¿Qué hace el comando git stash?

El comando git stash permite guardar cambios realizados en el código y retomar la última versión o commit almacenado, estos cambios también pueden recuperarse en cierto momento de ser requerido, o en caso de que se hayan deshecho por error del usuario.

6. Compare las operaciones git fetch y git pull

El comando git fetch guarda el URL de donde se encuentra el repositorio de un git. El comando git pull actualiza todos los archivos a través del URL del repositorio guardado por el comando git fetch. Es decir, el comando git fetch dice de donde bajar los datos o a donde subirlos, mientras que git pull los descarga.

7. Asumiendo que usted está en un Branch llamado “secundario” y su Branch principal se llama “master” ¿Qué resultado espera de hacer git rebase master? ¿Qué resultado espera de hacer git rebase origin/master?

Como rebase utiliza los datos del commit anterior en el master, esto le permite crear un “pseudo commit” para evitar afectar al master actual para el caso de git rebase master. Para el caso de git rebase origin/master sucede igual, pero no se altera el commit inicial.

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

Unittest es el marco de pruebas que viene con Python para hacer pruebas unitarias [4]. Esto consiste en probar con diferentes métodos secciones del código, a diferencia del código completo. Su utilidad se debe a que permite encontrar errores en partes específicas del código y señala posibles “puntos débiles” en que el código podría fallar a futuro.

9. Bajo el contexto de pytest. ¿Qué es un “assert”?

Un “assert” es una función empleada en pytest para comprobar que se cumple una condición [4]. En el código hay ciertos valores que se esperan tener. Para comprobar eso se utiliza la función assert para que se verifique que se cumple la condición dada por los valores que se esperan tener. Para ello, es necesario estipular primero cuáles son los valores que se esperan tener de modo que la función assert pueda hacer la comparación de la condición .

10. ¿Qué es Flake 8?

Flake 8 es una librería para Python que revisa el código y lo compara con normas establecidas de modo que se tenga un mayor orden y eficacia con el código. Ejemplos de algunos errores que puede detectar son librerías no usadas, comentarios que exceden una cantidad de caracteres específicos, variables locales no usadas, espacios en blanco después de comentarios entre otros.

Referencias

- [1] git (S.F). [Online]. Recuperado de: <https://git-scm.com/>
[2]hostinger (2019, mayo 13). [Online]. Recuperado de: <https://www.hostinger.es/tutoriales/que-es-github>
[3]A. Sharma, "Unit Testing in Python", *Datacamp*, 2020. [Online]. Recuperado de: <https://www.datacamp.com/community/tutorials/unit-testing-python>.
[4] "Assert statements", *Docs.pytest.org*, 2015. [Online]. Recuperado de: https://docs.pytest.org/en/reorganize-docs/new-docs/user/assert_statements.html.

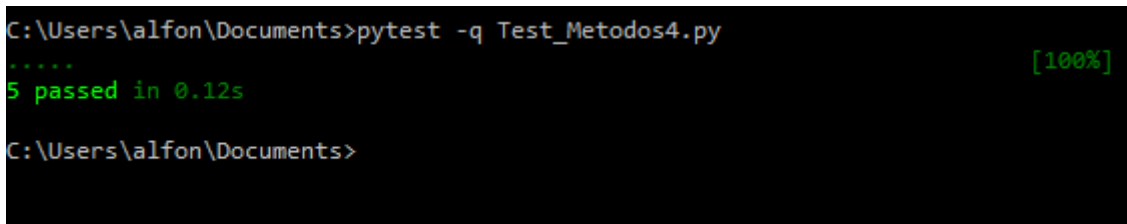
Pruebas Realizadas

- **Pytest**

Para la realización de la prueba de pytest primero, en la ventana de comandos, se instaló el programa utilizando el comando: `pip install -U pytest`

El archivo a probar es el que contiene los test de los dos métodos realizados. Este archivo de python se encuentra en el repositorio de github con el nombre `test_Metodos4`. Para realizar la prueba, en la ventana de comandos, se especificó la dirección en la que se ubica el archivo. Luego, se utilizó el comando: `pytest -q Test_Metodos4.py`

Con esto se obtuvieron los resultados de la figura 1. En este se observa que se obtuvieron cinco pruebas correctas, de las cuales, dos corresponden a la pruebas de caso de éxito para `check_char` evaluando todos los caracteres entre A-Z y a-z, y la prueba de caso de éxito para `caps_switch` evaluando todos los caracteres entre A-Z y a-z. Las otras tres, corresponden a las pruebas del caso negativo de `check_char` para los puntos b, c y d; en estas se espera que pasen el programa de pytest pero el código ya tiene un código de error único para que el programa identifique cada caso negativo.



```
C:\Users\alfon\Documents>pytest -q Test_Metodos4.py
..... [100%]
5 passed in 0.12s
C:\Users\alfon\Documents>
```

Figura 1. Prueba de pytest archivo `Test_Metodos4.py`.

- **Flake 8**

Para la realización de la prueba de pytest primero, en la ventana de comandos, se instaló el programa utilizando el comando: `pip install flake8`

En este caso se probaron tres archivos; el archivo que contiene los dos métodos realizados (`check_char` y `caps_switch`), el archivo que contiene los test de los dos métodos realizados y el archivo creado para que contuviera al menos 3

errores que flake8 pudiera detectar. Estos archivos de python se encuentran en el repositorio de github con los nombres Metodos3, test_Metodos4 y ErroresDetectables, respectivamente.

Para realizar la prueba, en la ventana de comandos, se especificó la dirección en la que se ubica el archivo. Luego, se utilizó el comando: `flake8 "NOMBRE".py`

Donde en el comando anterior en "Nombre" se colocó el nombre correspondiente de cada archivo al realizar la prueba. Para la prueba del archivo Metodos.py se obtuvo lo observado en la figura 2. Para la prueba del archivo Test_Metodos.py se obtuvo lo observado en la figura 3. Para la prueba del archivo ErroresDetectables.py se obtuvo lo observado en la figura 4.

Se puede observar que los errores obtenidos en las figuras 2 y 3, son todos del tipo E***/W***. Estos errores corresponden a errores o advertencias del estilo del código. Por mencionar algunos de los errores obtenidos:

- E501: Este error se presentaba cuando se escribían líneas de código muy extensas debido a los comentarios. Para corregirlo, se escribieron comentarios más concisos.
- E712: Este error se presentaba debido a que en una condición if, se tenía una variable booleana y se estaba verificando si era TRUE utilizando un doble igual. Para corregirlo, se utilizó la palabra "is" en lugar del doble igual.
- E225: Este error se presentaba debido a que se debía dejar un espacio en blanco antes y después de usar un operador como "=", "+" o "-".
- E302: Este error se presentaba debido a que se tiene que dejar dos espacios en blanco antes de poner un comentario en una línea de código.
- W293: Este error se presentó porque se tenían líneas en blanco al final del código

En la figura 4 se puede observar los resultados obtenidos para el código escrito que debía poseer al menos tres errores detectables por flake8. Algunos de los errores de este código son:

- Se exportó una librería que no se iba a utilizar.
- Se definió una variable que no se utilizó.
- No se dejaron los espacios necesarios antes de poner un comentario.

- Se debía dejar dos líneas en blanco antes de la definición de la función y no se hizo.
- No se dejó un espacio en blanco antes y después de usar el operador “=”

Luego, se corrigieron los errores correspondientes en cada uno de los archivos y se volvió a realizar la prueba, en este caso ya no se obtuvieron errores detectables por flake8.

```
:\Users\alfon\Documents>flake8 Metodos3.py
etodos3.py:2:80: E501 line too long (87 > 79 characters)
etodos3.py:3:80: E501 line too long (83 > 79 characters)
etodos3.py:4:80: E501 line too long (87 > 79 characters)
etodos3.py:5:31: E712 comparison to True should be 'if cond is True:' or 'if cond:'
etodos3.py:5:80: E501 line too long (110 > 79 characters)
etodos3.py:9:80: E501 line too long (140 > 79 characters)
etodos3.py:9:141: W291 trailing whitespace
etodos3.py:10:80: E501 line too long (101 > 79 characters)
etodos3.py:11:80: E501 line too long (182 > 79 characters)
etodos3.py:11:183: W291 trailing whitespace
etodos3.py:13:80: E501 line too long (132 > 79 characters)
etodos3.py:17:80: E501 line too long (88 > 79 characters)
etodos3.py:18:80: E501 line too long (140 > 79 characters)
etodos3.py:19:80: E501 line too long (110 > 79 characters)
etodos3.py:20:31: E712 comparison to True should be 'if cond is True:' or 'if cond:'
etodos3.py:20:80: E501 line too long (83 > 79 characters)
etodos3.py:21:80: E501 line too long (80 > 79 characters)
etodos3.py:23:80: E501 line too long (80 > 79 characters)
```

Figura 2. Prueba de pytest archivo Metodos3.py.

```

:\Users\alfon\Documents>flake8 Test_Metodos3.py
est_Metodos3.py:7:9: E225 missing whitespace around operator
est_Metodos3.py:7:80: E501 line too long (105 > 79 characters)
est_Metodos3.py:8:10: E225 missing whitespace around operator
est_Metodos3.py:8:80: E501 line too long (93 > 79 characters)
est_Metodos3.py:9:10: E225 missing whitespace around operator
est_Metodos3.py:9:80: E501 line too long (96 > 79 characters)
est_Metodos3.py:11:1: E302 expected 2 blank lines, found 1
est_Metodos3.py:11:80: E501 line too long (104 > 79 characters)
est_Metodos3.py:15:1: E302 expected 2 blank lines, found 1
est_Metodos3.py:21:1: W293 blank line contains whitespace
est_Metodos3.py:22:1: W293 blank line contains whitespace
est_Metodos3.py:23:5: E303 too many blank lines (2)
est_Metodos3.py:27:1: E302 expected 2 blank lines, found 1
est_Metodos3.py:29:17: E231 missing whitespace after ','
est_Metodos3.py:31:1: E302 expected 2 blank lines, found 1
est_Metodos3.py:33:17: E231 missing whitespace after ','
est_Metodos3.py:35:1: E302 expected 2 blank lines, found 1
est_Metodos3.py:37:15: E231 missing whitespace after ','
est_Metodos3.py:44:1: W293 blank line contains whitespace
est_Metodos3.py:45:1: W391 blank line at end of file
est_Metodos3.py:45:1: W293 blank line contains whitespace

```

Figura 3. Prueba de pytest archivo Test_Metodos4.py.

```

C:\Users\alfon\Documents>flake8 ErroresDetectables.py
ErroresDetectables.py:1:1: E265 block comment should start with '#'
ErroresDetectables.py:4:1: F401 'matplotlib.pyplot as plt' imported but unused
ErroresDetectables.py:6:9: E225 missing whitespace around operator
ErroresDetectables.py:8:1: E302 expected 2 blank lines, found 1
ErroresDetectables.py:10:5: F841 local variable 'VariableLocalNoUsada' is assigned to but never used
ErroresDetectables.py:10:29: E261 at least two spaces before inline comment
ErroresDetectables.py:10:29: E262 inline comment should start with '#'
ErroresDetectables.py:10:80: E501 line too long (175 > 79 characters)
ErroresDetectables.py:13:1: W391 blank line at end of file
ErroresDetectables.py:13:1: W293 blank line contains whitespace

```

Figura 4. Prueba de pytest archivo ErroresDetectables.py.