Tarea 1

GitHub, Pytest y Flake 8

A continuación, se presentan 2 asignaciones para entender y aplicar herramientas de desarrollo de proyectos de programación.

Preguntas Teóricas (20 pts, 2pts c/u)

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

Git permite el control de versiones para un código, facilitando el seguimiento, organización y gestión de los cambios realizados en un proyecto de software. Esto permite volver a versiones anteriores si hay un error, trabajar en paralelo mediante ramas (branches) sin afectar la versión principal, colaborar con múltiples desarrolladores e integrara cambios locales y remotos de manera controlada.

2) ¿Qué es un commit?
Un commit es un registro permanente de cambios realizados en el repositorio en un momento específico. Cuando se hace un commit se guardan los cambios realizados en los archivos, se crea un punto de historial para el proyecto al que se puede regresar si se necesita y se puede agregar un mensaje que describa los cambios que se realizaron.

3) ¿Qué es un branch?
Es una línea de desarrollo independiente dentro de un mismo repositorio, hacer esto es útil porque permite realizar cambios en el código sin afectar la rama principal. Cada branch tiene su propio historial de commits y una vez que los cambios estén revisados se puede ingresar nuevamente a la rama principal.

4) En el contexto de github. ¿Qué es un Pull Request?
Es una solicitud para integrar los cambios de una branch a otra (generalmente la principal). Permite revisar los cambios realizados (commits y diferencias de código), discutir modificaciones antes de integrarlas y aprobar o rechazar la integración de los cambios.

5) ¿Si un usuario quiere “Actualizar su repositorio contra el Branch master” que debe de hacer?
Hacer esto significa sincronizar la copia local del usuario con la versión más reciente de la rama principal (o branch master), para esto primero se debe posicionar en el branch master y ejecutar un git pull para actualizarlo con la versión más reciente. En código esto sería:

```
git checkout master
git pull origin master
```

Al utilizar pull en lugar de push, el que se actualiza es el local.

6) ¿Bajo que condiciones una herramienta como Git necesita apoyo de un humano para saber como integrar cambios locales con cambios remotos?
Es posible que se genere una situación llamada “merge conflict”, que sucede cuando dos personas modifican la misma sección de un código, ya sea trabajando en ramas distintas o cuando un desarrollador intenta hacer push o pull de cambios que chocan con modificaciones que ya existen en el repositorio remoto. En ese caso, la plataforma necesita una confirmación manual de cuál versión utilizar, ya que no puede tomar la decisión por sí misma al tener dos modificaciones igualmente válidas. Una decisión errónea podría romper la funcionalidad del proyecto.

7) ¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?
Una prueba unitaria permite aislar una sección del código para probarla independientemente y confirmar su correcta funcionalidad. Es habitual utilizarlo para buscar errores, analizando las funciones, métodos o clases para comprobar que se comportan como se espera ante distintas entradas y condiciones.

En Python, se puede utilizar Pytest para realizar unittests de forma flexible.

8) Bajo el contexto de pytest. ¿Cuál es la utilidad de un "assert"?
En pytest, un assert es la función para verificar que el resultado de una función coincide con el valor esperado. Al ejecutar una prueba, si un assert da error, pytest detiene la prueba e indica en cuál función hubo un resultado incorrecto. Su utilidad radica en la facilidad para revisar códigos y determinar el punto exacto en el que se debe hacer una corrección.

9) ¿Qué es Flake 8?
Flake 8 es una combinación de herramientas que se añade a un proyecto de código, que permite hacer revisiones de código en busca de errores, malas prácticas y problemas de formato, sin tener que correr el código. Las herramientas que utiliza son:

PyFlakes, que detecta errores lógicos

PyCodeStyle, que verifica que el código siga las convenciones oficiales de estilo

McCabe, que analiza la complejidad de la estructura lógica

10) Comente sobre la utilidad de la aleatorización en pruebas de código.
La aleatorización es una técnica que implica variar el orden de ejecución de las pruebas o los datos de entrada de forma aleatoria, para detectar dependencias ocultas entre pruebas, donde una prueba depende del estado que dejó otra para funcionar correctamente. Al romper el orden fijo de ejecución, obliga a que cada prueba sea realmente independiente, volviendo los tests más robustos. Además, aleatorizar los datos de entrada permite explorar un mayor número de casos posibles y aumentar la probabilidad de descubrir errores.

Sección Práctica (55 pts, puntos varían según la sección)

La sección práctica busca ejercitar las tres herramientas vistas en la sección teórica para esto realice los siguientes pasos:

1) Cree un nuevo repositorio público en github. El nombre debe de ser: TareasPrimerApellidoEstudiante1PrimerApellidoEstudiante2. Dentro de este repositorio cree una carpeta llamada "Tarea 1", todos los documentos de esta tarea mencionados en los siguientes puntos deben ir en esta carpeta. (**1.5 pts**)
2) Suba todas las respuestas de la parte teórica al repositorio como un primer commit, asegúrese de subir las respuestas como un archivo PDF. Asegure usar una descripción intuitiva al realizar su commit. (**1.5 pts**)
3) Utilice pytest para probar completamente una serie de funciones de python. El código por probar debe de ser el siguiente (**Total: 30 pts**):

Dos métodos distintos **(5 pts c/u)**:

1. Un método "**filtrar_vocales**" recibe dos parámetros: Cadena y Bandera. Y posee el siguiente comportamiento.
    a. Debe verificar que el parámetro cadena sea un string. En caso de no cumplir esta limitación el código retorna un **código de error único.**

b. Debe verificar que el parámetro cadena solo contenga letras del abecedario. En caso de no cumplir esta limitación el código retorna un **código de error único.**
c. Debe verificar que el parámetro cadena no sea un string vacío. En caso de no cumplir esta limitación el código retorna un **código de error único.**
d. Debe verificar que el parámetro cadena no sea mayor a 30 caracteres. En caso de no cumplir esta limitación el código retorna un **código de error único.**
e. Debe verificar que el parámetro "bandera" sea un booleano (True o False). En caso de no cumplir esta limitación el código retorna un **código de error único.**
f. Si la bandera es True, el método retorna un string que contiene solo las vocales (a, e, i, o, u, A, E, I, O, U) de la cadena original en el mismo orden que aparecen.
g. Si la bandera es False, el método retorna un string que contiene solo las consonantes de la cadena original en el mismo orden que aparecen.
h. En caso de ejecutar correctamente debe de retornar **un código de éxito único.**
i. El método siempre devuelve dos valores. El código de error/éxito y el string filtrado en ese orden específico. En los casos con código de error en vez de retornar el string filtrado se retorna None.

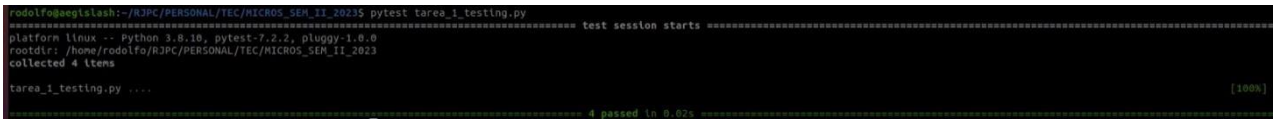
2. Un método "**encontrar_extremos**" que recibe un parámetro de entrada "lista_numeros" y debe de:
    a. Debe verificar que el parámetro de entrada sea una lista. En caso de no cumplirse retorna un **código de error único**.
    b. Debe verificar que todos los elementos de la lista sean números (int o float). En caso de no cumplirse retorna un **código de error único**.
    c. Debe verificar que la lista no esté vacía. En caso de no cumplirse retorna un **código de error único**.
    d. Debe verificar que la lista no tenga más de 15 elementos. En caso de no cumplirse retorna un **código de error único**.
    e. Encuentra el valor mínimo y el valor máximo de la lista. En caso de hacer esto debe de retornar **un código de éxito único.**
    f. El método siempre devuelve tres valores. El código de error/éxito, el valor mínimo y el valor máximo en ese orden específico. En los casos con código de error en vez de retornar los valores mínimo y máximo se retorna None dos veces.

Este código debe de ser su propio archivo .py.

Adjunto a este enunciado usted puede encontrar un archivo de Python llamado **tarea_1_testing.py.** Este archivo contiene las pruebas de pytest que van a ser utilizados

para evaluar el código, por cada prueba que el código pase exitosamente se darán 5pts hasta un total de **20 pts**. El archivo de pruebas debe de poder ejecutar su código sin necesidad de ser modificado, para esto contemple las siguientes indicaciones:

- El archivo de pruebas contiene la forma en que se debe de llamar el archivo con su código.
- El archivo de pruebas explica los códigos de error que se esperan obtener de las funciones
- El archivo de pruebas asume que el archivo con el código que define los métodos se encuentra en la misma carpeta.
- Usted está en libertad de descargar el archivo de pruebas y correrlo localmente para asegurar el funcionamiento de su código.
- Se espera ver lo siguiente al ejecutar el archivo de pruebas:



4) Use flake8 y asegure que su código no presenta errores de formato. (**5 pts**)
5) Suba el código de funciones y el código de pruebas a su repositorio en github como un commit.
6) Cree un branch del master de su repositorio e introduzca un cuarto commit en el cual se agrega un nuevo archivo con 3 errores identificables por flake 8. No puede reutilizar los archivos de los puntos 4 y 5 para esto. (**5 pts**)
7) En el mismo Branch del paso 6 agregue otro commit arreglando los errores de flake 8 del paso 6 (**5pts**)
8) Suba el branch a su repositorio de github. NOTA: Esto implica que el master de su repositorio va a estar dos commit por detrás del Branch.
9) Realice un pull request para poder llevar el commit de su branch al master. El pull request puede estar mergeado o solo creado. (**2 pts**)

Instrucciones Generales

1) La tarea debe ser realizada en parejas
2) Fecha de entrega viernes 27 de febrero.
3) Entregables: Un archivo .txt que posea un link al repositorio de github con todos los archivos y branches solicitados.
4) El repositorio deberá nombrarse como: TareasPrimerApellidoEstudiante1PrimerApellidoEstudiante2, de no seguir esta indicación se le descontarán 10 puntos.
5) Ausencia de comentarios en el código se traducirá a una reducción de 10 puntos de la nota final. Comentarios sustanciales explican los parámetros de entrada y salida de un método, una explicación del método como tal, comentarios dentro del código que orienten al lector para entender la solución.
6) Al TEC digital solo entregan un archivo de texto con el link al repositorio

Uso de Inteligencia Artificial

Para esta evaluación los estudiantes pueden hacer uso libre de herramientas de inteligencia artificial en lo que se refiere a:

1. Generación de código Python.
2. Búsqueda de información para las preguntas teóricas.

En caso de reconocer cualquier otro uso la nota de la tarea será de 0.

Recomendaciones

Usar el tutorial de git conocido como http://gitimmersion.com/ esto provee el conocimiento básico suficiente para realizar la tarea.

Para entender el concepto de “un código de error” se sugiere observar el estándar de errores de sistema de Linux: https://mariadb.com/kb/en/operating-system-error-codes/.

En ocasiones conviene descargar una máquina virtual e instalar Linux. Esto solamente para facilitar el uso de herramientas como pytest desde la terminal.

Hacer uso de la documentación oficial de pytest: https://docs.pytest.org/en/latest/getting-started.html