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) ¿Diferencie la herramienta Git de Github?
2) ¿Qué es un branch?
3) ¿Qué es un commit?
4) ¿Qué es la operación cherry-pick?
5) ¿Qué hace el comando git stash?
6) ¿Compare las operaciones git fetch y git pull
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?
8) ¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?
9) Bajo el contexto de pytest. ¿Qué es un "assert"?
10) ¿Qué es Flake 8?

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 del mismo debe nombrarse como: Tarea1PrimerApellidoEstudiante1PrimerApellidoEstudiante2. (**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: 35 pts**):

   Dos métodos distintos **(5 pts c/u)**:
   1. Un método "**check_char**" recibe un único parámetro y posee el siguiente comportamiento.
      a. Si recibe **un único** carácter entre A-Z debe de retorna un 0. No importa si es mayúscula o minúscula, no es necesario contemplar la letra ñ.
      b. Si recibe **más de un carácter** debe de retornar un código de error único.
      c. Si recibe **uno o más caracteres que no sean parte del rango A-Z** debe de retornar un código de error único.

   d. Si el parámetro no es un carácter o string retorna un código de error único. Esto aplica para cualquier otro tipo de dato: int, array, float, clase, etc.
2. Un método "**caps_switch**" que recibe un único parámetro de entrada y posee el siguiente comportamiento.
   a. Usando **check_char** debe de verificar si se le pasó un único carácter entre A-Z. Si esto no es cierto retorna el mismo código de error que **check_char** retorne.
   b. Si **check_char** define que se pasó un único carácter entre A-Z (es decir **check_char** retorna 0). El método debe de pasar el carácter a mayúscula si estaba en minúscula o viceversa.

Este código debe de ser su propio archivo .py. **A parte debe crear un segundo archivo .py** que prueba las funciones creadas mediante pytest y asserts, específicamente debe probar **(5 pts c/u)**:

- Un caso de éxito para **check_char** evaluando todos los caracteres entre A-Z y a-z. Esto quiere decir que se llama a **check_char** y se espera que su resultado siempre sea 0.
- Un caso de éxito para **caps_switch** evaluando todos los caracteres entre A-Z y a-z. Esto quiere decir que; por ejemplo, si se llama el método con parámetro de "a" la respuesta debe de ser "A".
- Un caso negativo donde se verifica el punto b del método **check_char**
- Un caso negativo donde se verifica el punto c del método **check_char**
- Un caso negativo donde se verifica el punto d del método **check_char**

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 testing a su repositorio en github como un commit.
6) Escriba un código en Python que presente al menos tres errores detectables por flake8. (**5 pts**)
7) Suba el código con errores a github como otro commit.
8) Cree un branch del master de su repositorio e introduzca un cuarto commit con los arreglos al archivo .py del punto 6 y 7. (**5 pts**)
9) Suba el branch a su repositorio de github. NOTA: Esto implica que el master de su repositorio va a estar un commit por detrás del Branch.
10) Realice un pull request para poder llevar el commit de su branch al master. (**2 pts**)

Instrucciones Generales

1) La tarea debe ser realizada en parejas
2) Fecha de entrega viernes 5 de marzo.
3) Entregables: Link al repositorio de github con todos los archivos y branches solicitados.

4) El repositorio deberá nombrarse como: Tarea1PrimerApellidoEstudiante1PrimerApellidoEstudiante2, 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. Para los archivos de testeo los comentarios deben explicar los pasos a realizar en el test.
6) Al TEC digital solo entregan un archivo de texto con el link al repositorio
7) La revisión del código será realizada por el profesor de curso descargando el repositorio y ejecutando el código.

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://www-numi.fnal.gov/offline_software/srt_public_context/WebDocs/Errors/unix_system_errors.html. La pareja de trabajo no está obligada a usar estos errores en su solución. Se espera que el grupo de trabajo use códigos de error numéricos.

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