TAREA 6

Fecha de entrega: 10/11/2018 23:59 hrs

Problema 1

Software Carpentry es una fundación sin fines de lucro que tiene como objetivo enseñar a científicos e ingenieros de diversas ramas las habilidades necesarias *"to get more done in less time, and with less pain"*.

El siguiente tutorial desarrollado por Software Carpentry les ayudará a mejorar el diseño del software que desarrollen para sus tareas y en el futuro. Estudien el tutorial y respondan las preguntas que se encuentran a continuación. Incluya sus respuestas en el informe. No se complique haciéndolas calzar con el formato del informe, simplemente responda las preguntas.

El tutorial en la página de Software Carpentry (link) incluye las diapositivas y una transcripción del video. También está disponible como youtube playlist (link). Les recomiendo reproducir el video a velocidad $1.5\times - 2\times$.

Preguntas:

- Describa la idea de escribir el `main driver` primero y llenar los huecos luego. ¿Por qué es buena idea?
- ¿Cuál es la idea detrás de la función `mark_filled`? ¿Por qué es buena idea crearla en vez del código original al que reemplaza?
- ¿Qué es refactoring?
- ¿Por qué es importante implmentar tests que sean sencillos de escribir? ¿Cuál es la estrategia usada en el tutorial?
- El tutorial habla de dos grandes ideas para optimizar programas, ¿cuáles son esas ideas? Descríbalas.
- ¿Qué es *lazy evaluation*?
- Describa la *"other moral"* del tutorial (es una de las más importantes a la hora de escribir buen código).

Problema 2

Para planetas que orbitan **cerca del Sol**, el potencial gravitacional se puede escribir como:

$$U(r) = -\frac{GMm}{r} + \alpha\frac{GMm}{r^2}$$

donde α es un número pequeño. Esta corrección a la ley de gravitación de Newton es una buena aproximación derivada de la teoría de la relatividad general de Einstein.

Bajo este potencial, las órbitas siguen siendo planas pero ya no son cerradas, sino que precesan, es decir, el afelio (punto más alejado del Sol en la órbita) gira alrededor del Sol.

1. El archivo llamado `planeta.py` contiene el esqueleto de la clase `Planeta`. Ud. debe implementar los métodos de esa clase. Los docstrings explican en qué debe consistir cada método. Ud. tiene libertad de mejorar los docstrings, y agregar atributos y métodos a la clase según le parezca conveniente para resolver el problema descrito a continuación.

El archivo llamado `solucion_usando_rk4.py` muestra cómo incluir la clase `Planeta` en un script separado. Ud. también puede resolver todo dentro del mismo archivo, en cuyo caso puede descartar `solucion_usando_rk4.py`.

2. Parta por estudiar el caso $\alpha = 0$ y considere las siguientes condiciones iniciales:

$$x_0 = 10$$
$$y_0 = 0$$
$$v_x = 0$$

   Además, utilice unidades tales que $GMm = 1$ y escoja v_y según le parezca (pero asegúrese de que la energía total sea negativa).

   Integre la ecuación de movimiento por aproximadamente 5 órbitas usando los métodos de RK4 y Verlet. Plotee las órbitas y la energía total del sistema como función del tiempo en los 2 casos. Comente los resultados.

3. Ahora considere el caso $\alpha = 10^{-2.XXX}$ (donde XXX son los 3 últimos digitos de su RUT, antes del dígito verificador). Integre la ecuación de movimiento usando el método de Verlet por al menos 30 órbitas. Determine la velocidad angular de precesión. ¿Cómo lo hizo? En particular, ¿cómo determinó la posición del afelio? Grafique la órbita y la energía como función del tiempo.

> **Comentario.** Esta tarea pide explicitamente que utilice OOP (Object Oriented Programming) para su desarrollo. Es un ejercicio útil para aprender esta técnica.

Instrucciones Importantes.

- El algoritmo RK4 está implementado en muchas librerías y en la tarea pasada Ud. ya lo implementó. Si quiere, puede (recomendado) utilizar su propia implementación pero también puede utilizar otra de uso libre. El algoritmo de Verlet, sin embargo, lo debe implementar Ud. para esta tarea (puede usar cualquiera de sus variantes).
- Evaluaremos su uso correcto de `python`. Si define una función relativametne larga o con muchos parámetros, recuerde escribir el *docstring* que describa los parámetros que recibe la función, el output, y el detalle de qué es lo que hace la función. Recuerde que generalmente es mejor usar varias funciones cortas (que hagan una sola cosa bien) que una muy larga (que lo haga todo). Utilice nombres explicativos tanto para las funciones como para las variables de su código. El mejor nombre es aquel que permite entender qué hace la función sin tener que leer su implementación ni su *docstring*.
- Su código debe aprobar la guía sintáctica de estilo (`PEP8`). En esta página puede chequear si su código aprueba `PEP8`.

- Utilice `git` durante el desarrollo de la tarea para mantener un historial de los cambios realizados. La siguiente cheat sheet le puede ser útil. **Revisaremos el uso apropiado de la herramienta y asignaremos una fracción del puntaje a este ítem.** Realice cambios pequeños y guarde su progreso (a través de *commits*) regularmente. No guarde código que no corre o compila (si lo hace por algún motivo deje un mensaje claro que lo indique). Escriba mensajes claros que permitan hacerse una idea de lo que se agregó y/o cambió de un `commit` al siguiente.

- Para hacer un informe completo Ud. debe decidir qué es interesante y agregar las figuras correspondientes. No olvide anotar los ejes e incluir una *caption* o título que describa el contenido de cada figura. Tampoco olvide las unidades asociadas a las cantidades mostradas en los diferentes plots.

- La tarea se entrega subiendo su trabajo a github. Clone este repositorio (el que está en su propia cuenta privada), trabaje en el código y en el informe y cuando haya terminado asegúrese de hacer un último `commit` y luego un `push` para subir todo su trabajo a github.

- El informe debe ser entregado en formato `pdf`, este debe ser claro sin información de más ni de menos. **Esto es muy importante, no escriba de más, esto no mejorará su nota sino que al contrario**. La presente tarea probablemente no requiere informes de más de 4 o 5 páginas en total (dependiendo de cuántas figuras incluya; esto no es una regla estricta, sólo una referencia útil). Asegúrese de utilizar figuras efectivas y tablas para resumir sus resultados. Revise su ortografía.

- No olvide indicar su RUT en el informe.

- Repartición de puntaje: P1: 30 %; P2: 70 %. El P2 además, se subdivide de la siguiente forma: 40 % implementación y resolución del problema (independiente de la calidad de su código); 45 % calidad del reporte entregado: demuestra comprensión del problema y su solución, claridad del lenguaje, calidad de las figuras utilizadas; 5 % aprueba o no `PEP8`; 10 % diseño del código: modularidad, uso efectivo de nombres de variables y funciones, docstrings, uso de git, etc.