uniandes

Laboratorio de Métodos Computacionales
Tarea 5 - Sistemas de ecuaciones lineales
01-2018

uniandes

La solución a este taller debe subirse por SICUA antes de terminada la clase. Los archivos código fuente deben subirse en un único archivo `.zip` con el nombre `NombreApellido_hw5.zip`, por ejemplo yo debería subir el zip `JesusPrada_hw5.zip` (10 puntos). Recuerden que es un trabajo individual y debe ser realizado en un script de python (.py).

Aclaraciones sobre python: El alcance(scope) de las variables

En python y en general en la mayoría de lenguajes de programación hay dos tipos de variables importantes: las variables **LOCALES** y las variables **GLOBALES**.

Las variables **globales** son declaradas por fuera de funciones. **Una buena manera de saber si una variable es global es si no está indentada**. Estas variables son accesibles por todos las funciones, ciclos, etc.

Las variables **locales** son declaradas adentro de una función, o como parámetro. Estas variables son accesibles únicamente adentro de la función en la que fueron declaradas.

Ahora, en python la sintaxis intuitiva y poco exigente, en donde **la asignación de variables automáticamente las declara**, se presta para confusiones con las variables locales y globales. Es decir, en python es posible definir una variable global y luego definir una variable local con el mismo nombre. En este caso, la variable queda local o global? Demostrémoslo!

1. (15 points) **Comentarios**

   Por favor comenten todo su código. Específicamente, a cada variable importante o no evidente deben comentar su significado. A cada función deben comentar su propósito principal y el significado de sus parámetros. A cada iteración deben comentar su objetivo. Se acepta declarar variables con nombre evidente como `matrix` en lugar de comentar su significado.

2. (5 points) **Declarar variables globales**

   Declare una variable global llamada `var1` y otra llamada `var2`. Estas variables deben tener el contenido `'variable 1 global'` y `'variable 2 global'` .

3. (5 points) **Declarar una variable local (1)**

   Como vimos anteriormente, hay dos maneras de declarar una variable local. Una es asignándole un valor dentro de la función.

   Defina una función llamada `fun1()`. Adentro de la función declaren una variable local llamada `var1` con el contenido `'variable 1 local'`. Inmediatamente después esta función debe imprimir la variable `var1`.

Cabe aclarar que definir una función **NO** la ejecuta. El contenido de una función es ejecutado únicamente al **LLAMAR** la función, no al **DEFINIRLA**.

4. (5 points) **Declarar una variable local (2)**

   Otra manera de declarar una variable local es declarándola como parámetro de una función.

   Defina una función llamada `fun2(var2)` que tome como parámetro `var2` y la imprima.

5. (5 points) **Globales o locales?**

   Imprima `'Ejecutando fun1()'`, luego ejecute la función `fun1()`. Imprima `'Imprimiendo var1'`, luego imprima la variable `var1`.

   Imprima `'Ejecutando fun2()'`, luego ejecute la función `fun2('variable 2 local')`. Imprima `'Imprimiendo var2'`, luego imprima la variable `var2`.

   Hemos declarado una variable global y luego hemos declarado una variable local con el mismo nombre. Comente (con un print) si la variable es local, global o ambas.

6. (5 points) **Se puede asignar un valor a una variable global dentro de una función?**

   Como vimos anteriormente, al asignarle el valor a una variable adentro de una función, se está declarando como local. Esto significa que esa asignación sólo es válida dentro de la función. Esta confusión se puede arreglar trabajando con nombres diferentes de variables. Es decir, si se declara una variable (global) por fuera de una función, es mejor NO declarar una variable (local) con el mismo nombre adentro de la función.

   Sin embargo, a veces es necesario asignar o cambiar el valor de una variable global, adentro de una función. Recordando que asignar el valor de una variable adentro de una función la declara como local, esto es un problema que no se puede solucionar cambiando el nombre de las variables. En caso de que se quiera modificar una variable global **adentro** de una función, es necesario, antes de declararla local, recordarle a python que se modificará la variable global. Para esto, se usa la siguiente línea de código:

   ```
   global var
   ```

   Declare una función llamada `fun3()`. Adentro de la función modifique la variable global llamada `var1` con el contenido `'variable 1 global modificada'`. Inmediatamente después esta función debe imprimir la variable `var1`.

   Imprima `'Ejecutando fun3()'`, luego ejecute la función `fun3()`. Imprima `'Imprimiendo var1'`, luego imprima la variable `var1`. Noten la diferencia con el punto anterior.

   Cabe aclarar que establecer una variable `var` como praámetro de una función automáticamente la hace local. Dado que `global var` tiene que ser ejecutado antes de que la variable sea declarada local, modificar una variable global llamada `var` dentro de una función que tiene como parámetro la variable `var`, es imposible.

Ahora, la tarea de verdad!

El filtro sepia

El filtro sepia obtiene nuevas componentes RGB de una imagen como combinación lineal de las anteriores componentes. Los valores recomendados para esta combinación están dados por una operación matricial:

$$\begin{bmatrix} sR \\ sG \\ sB \end{bmatrix} = \begin{bmatrix} .393 & .769 & .189 \\ .349 & .686 & .168 \\ .272 & .534 & .131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

Esto quiere decir que si tenemos una foto en sepia, es posible obtener la foto original resolviendo el sistema lineal de ecuaciones.

1. (30 points) **Funcion de solucion de sistemas lineales**

   Cree una función que dada una matriz `M` y un vector `b`, resuelva el sistema lineal `Mb = v` por eliminación gaussiana. Creen su propio código y comenten absolutamente todo. Esta funcion debe **RETORNAR** el valor del vector `x` solución.

2. (20 points) **Cargar la imagen sepia**

   Cargue la imagen `sepia.npy` a un arreglo de numpy usando `np.load()`. Obtendrá una matriz donde cada elemento es un pixel consistente de un vector `[SR SG SB]`. Grafique esa imagen con `plt.imshow()` y guárdela en la imagen `sepia.png`

3. (20 points) **Obtener la imagen original**

   Para cada pixel `[SR SG SB]` obtenga el correspondiente `[R G B]` resolviendo el sistema lineal `llamando` la función anteriormente definida. La imagen original donde se guardan los valores `[R G B]` debe estar codificada en una matriz llamada `original`.

   Sin embargo, esta combinación puede arrojar valores mayores a 1, por lo cual es necesario renormalizar la imagen por el máximo valor general. En otras palabras, cuando acoplen los nuevos valores de RGB en la imagen `original`, deben realizar la siguiente operación:

   ```
   original = (original/original.max())
   ```

   Con `imshow()` verifiquen que recuperaron el color. Guarde la imagen como `original.png`