class5b

August 6, 2018

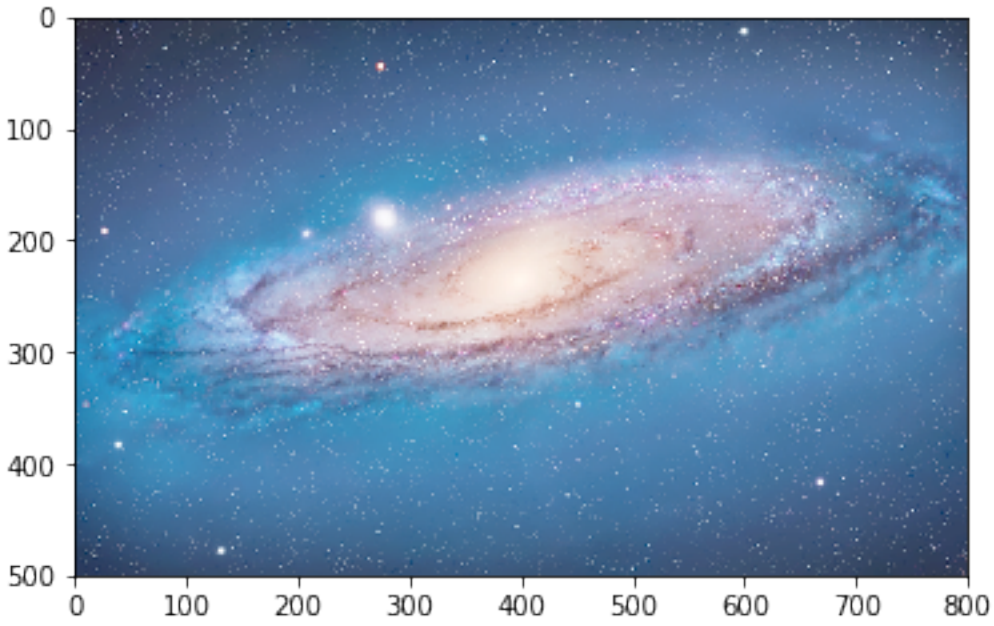
```
In [1]: %pylab inline
        import imageio as im
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: imagen = im.imread("galaxy_image2.png")
```

```
In [3]: imshow(imagen)
```

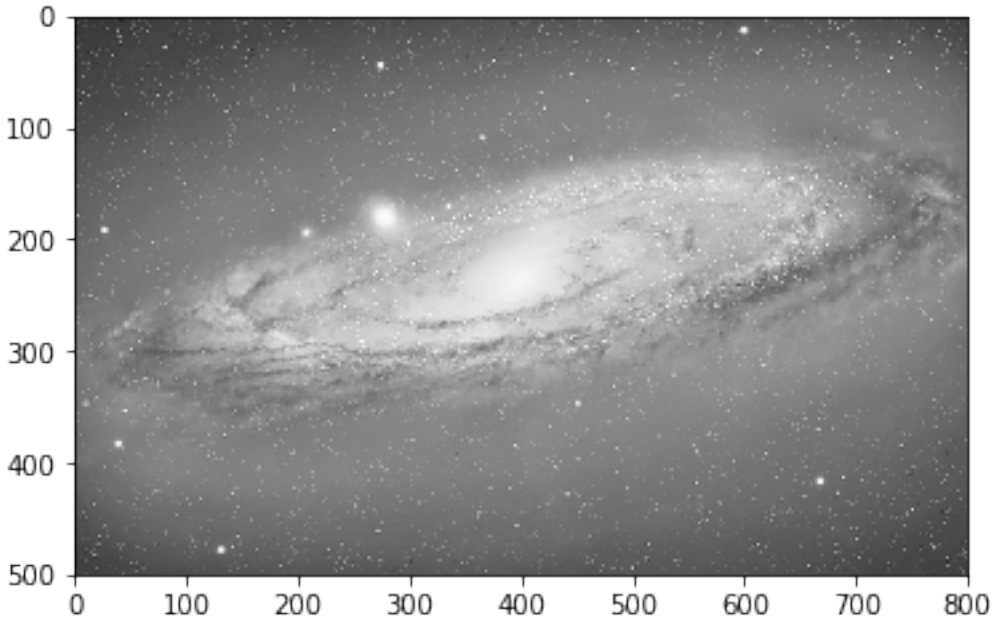
```
Out[3]: <matplotlib.image.AxesImage at 0x7f4582c72358>
```



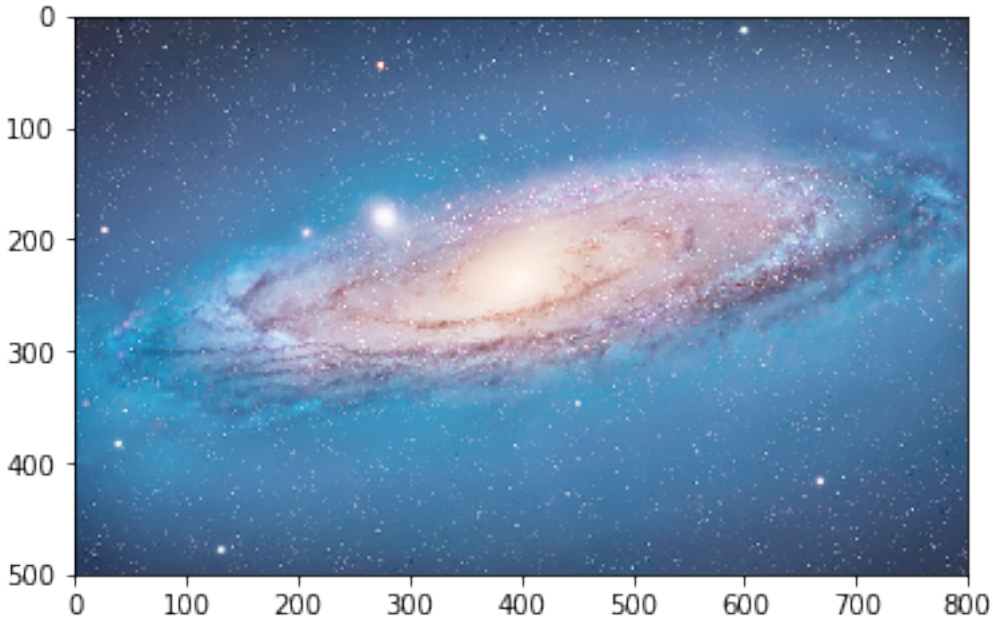

```
In [4]: imagen.shape
```

```
Out[4]: (500, 800, 3)
```

```
In [5]: for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1]):
                rgb = imagen[i,j,:]
                imagen[i,j,:] = mean(rgb)

In [6]: imshow(imagen)

Out[6]: <matplotlib.image.AxesImage at 0x7f4582beb6a0>
```

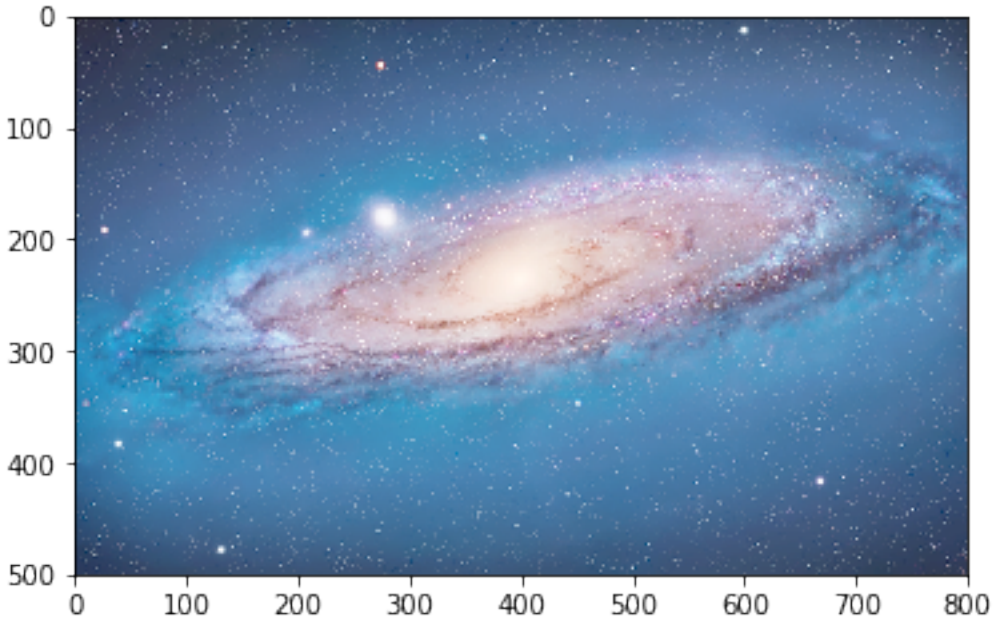



```
In [7]: imagen_original = im.imread("galaxy_image2.png")
        imagen = imagen_original.copy()

In [8]: imshow(imagen_original)

Out[8]: <matplotlib.image.AxesImage at 0x7f4582bac6d8>
```

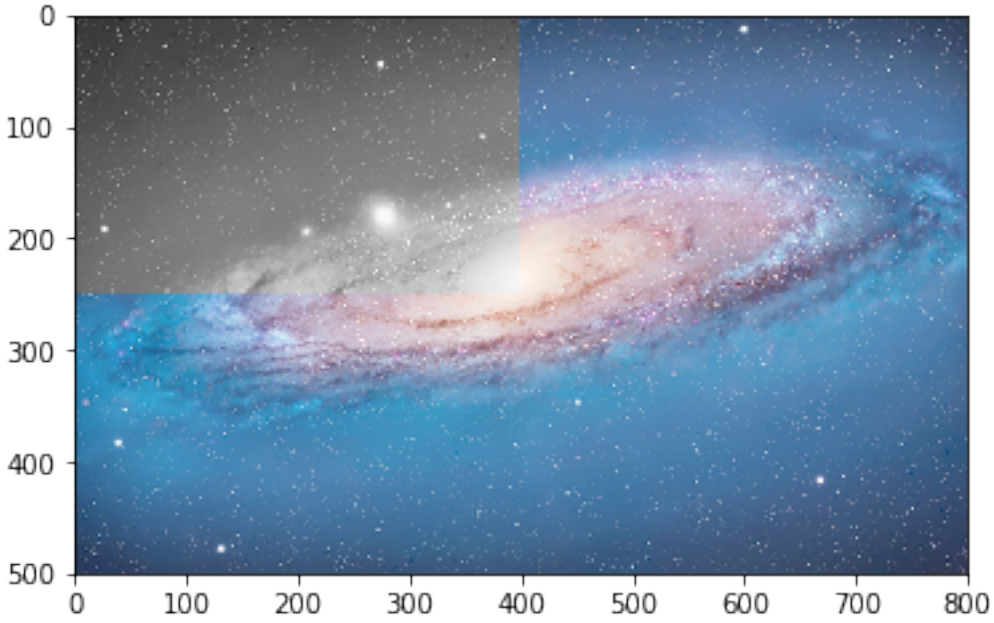


In [9]: imshow(imagen)

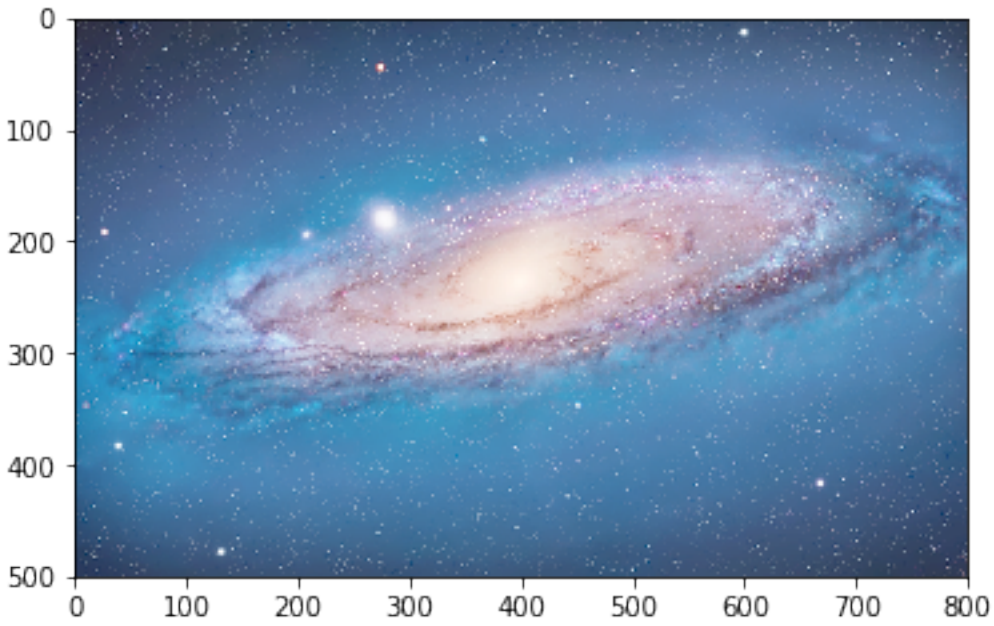
Out[9]: <matplotlib.image.AxesImage at 0x7f4582be2e48>



```
In [10]: def procesar_imagen(i1,i2,j1,j2):
             for i in range(i1,i2):
                 for j in range(j1,j2):
                     rgb = imagen[i,j,:]
                     imagen[i,j,:] = mean(rgb)

In [11]: imagen.shape

Out[11]: (500, 800, 3)

In [12]: procesar_imagen(0,250,0,400)

In [13]: imshow(imagen)

Out[13]: <matplotlib.image.AxesImage at 0x7f4582ba0828>
```




```
In [14]: imagen = imagen_original.copy()

In [15]: imshow(imagen)

Out[15]: <matplotlib.image.AxesImage at 0x7f4548e1bfd0>
```

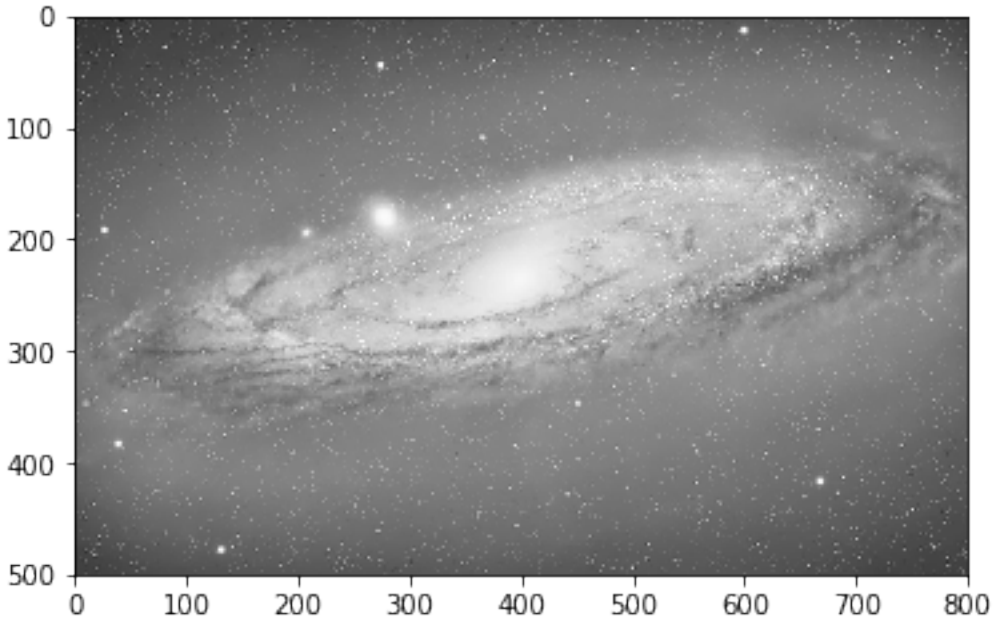


In [16]:
```
import threading
```

In [17]:
```
## Creamos los hilos o "threads", cada uno trabaja en una parte distinta de la imágen.
t1 = threading.Thread(target=procesar_imagen, args=(0,250,0,400))
t2 = threading.Thread(target=procesar_imagen, args=(0,250,400,800))
t3 = threading.Thread(target=procesar_imagen, args=(250,500,0,400))
t4 = threading.Thread(target=procesar_imagen, args=(250,500,400,800))

## Los hilos comienzan su trabajo...
t1.start()
t2.start()
t3.start()
t4.start()

## Esperamos hasta que todos los hilos hayan completado su trabajo...
t1.join()
t2.join()
t3.join()
t4.join()
```

In [18]:
```
imshow(imagen)
```

Out[18]:
```
<matplotlib.image.AxesImage at 0x7f4548dd9be0>
```



1 Multithreading

- Esta tarea es muy apto para *multithreading*: los threads en un proceso no comunican. Son "trabajadores" independientes.
- Pero, todos los threads ven el mismo bloque de memoria, donde tenemos los datos de la imágen.
- En esta tarea, el trabajo de cada thread es totalmente indepdendiente de los otros.

No todos los algoritmos son tan fácil de paralelizar para procesamiento multithread.

1.0.1 Multithreading en Python

- El ejemplo que tenemos aquí, de hecho, es algo que Python hace muy mal!
- Por razones que tiene que ver con el diseño del lenguaje, Python no hace *multithreading* de verdad!
- Su módulo "threading" es útil en un programa que busca datos del Internet, o que lee/escribe mucho en el disco duro. Pero no funciona bien para la paralelización de cálculos númericos...