Escalona_Joaquín_8

September 2018

Ejercicio 1

En el archivo de la imagen, localize la cabecera de la imagen. Esta esta compuesta de una lista de key words.

En este ejercicio se requerirán las siguientes librerías de Python:

```
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from matplotlib.colors import LogNorm
```

Escriban un programa que imprima la cabecera, y que identifique el numero de pixels que tiene la imagen (Nx y Ny), la escala de ángulo de cada pixel en arcsec (arcos de segundo), y la coordenada referencial de la imagen.

Abrimos el archivo y luego aplicamos el método `info()` para ver la estructura del FITS y saber cómo está construido.

```
hdulist = fits.open('onc_24.fits')
hdulist.info()
```

out:

```
    Filename: onc_24.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     107   (3442, 5614)   float32
```

Vemos que sólo tiene un HDU, con número 0. Para mostrar el **header** hacemos:

```
header = hdulist[0].header
header
```

out:

```
SIMPLE  =                    T / Fits standard
BITPIX  =                  -32 / Bits per pixel
NAXIS   =                    2 / Number of axes
NAXIS1  =                 3442 / Axis length
NAXIS2  =                 5614 / Axis length
EXTEND  =                    F / File may contain extensions
ORIGIN  = 'NOAO-IRAF FITS Image Kernel July 1999' / FITS file originator
DATE    = '2005-01-24T18:21:35' / Date FITS file was generated
IRAF-TLM= '11:21:30 (24/01/2005)' / Time of last modification
OBJECT  = 'Trapezium'          / Name of the object observed
```

```
COMMENT Begin Mips_enhancer specific processing information
COMMENT  File created by Mips_enhancer, part 3 of MIPS team DAT
ME_VER  = 'v2.80 (23 Jul 2004)' /  Version of Mips_enhancer
ME_ARRAY=                   24 / Mosaic for 24 micron array
ME_BCDNO=                 2475 / Number of BCDs used to create Mosaic
ME_NTILE=                  100 / Number of Tiles
ME_XTILE=                   10 /  Number of tiles in X direction
ME_YTILE=                   10 /  Number of tiles in Y direction
ME_SCALE=                  0.5 /  Pixel Scale of Mosaic pixels
ME_DIST =                    T /  Geometrical Distortion Applied
MEBOOSTF=                    T /  Not using Boost Frames in Mosaic
MEREJECT=                    1 /  Number of DCEs to reject after stim
METHRESH=                   -1. /  Threshold ratio for outlier rejection
MEOUTMED=                    T /  Outlier rejection based on median
MESIGCUT=                   4. /  Sigma clippping for Outliers
MESIGTOL=                  10. /  Sigma tolerance on iterating on Outliers
MESTDEVC=                    T /  Outlier Rejection uses calculated STDEV
MEMINPT =                    3 /  Min Points per pixel for Oultiers
MEMAXIT =                   30 /  Maximum number of iterations in outlier reject
MEW2    =                    T /  Weighting of fluxes based on overlap area
ME_MEANF=                    T /  Subpixel flux determined by weighted average
CRVAL1  =     83.8126875689956 /  RA reference point (deg)
CRVAL2  =    -5.35082057093471 /  DEC reference point (deg)
XPIXSIZE=     1.24500000476837 /  x pixel scale (arc seconds/pixel)
YPIXSIZE=     1.24500000476837 /  y pixel scale (arc seconds/pixel)
BCDXPIXS=      2.4942549020052 /  Original BCD x pixel scale (arc seconds/pixel)
BCDYPIXS=     2.59914897221395 /  Original BCD y pixel scale (arc seconds/pixel)
MECROTA2=                   0. /  Mosaic CROTA2 (degrees)
CD1_1   = -0.000345833334657881 /  CD matrix element 1_1
CD1_2   =                   0. /  CD matrix element 1_2
CD2_1   =                   0. /  CD matrix element 2_1
CD2_2   = 0.000345833334657881 /  CD matrix element 2_2
MEXRANGE=              35.8145 /  x range in arc minutes
MEYRANGE=              58.5565 /  y range in arc minutes
CRPIX1  =               1716.5 /  x reference point
CRPIX2  =               2822.5 /  y reference point
CTYPE1  = 'RA---TAN'           /  CTYPE1
CTYPE2  = 'DEC--TAN'           /  CTYPE2
EFILE0  = 'mips_AOR_4322816_scn_A24_leg10_3s.cal.fits' /  Input files used to cr
EFILE1  = 'mips_AOR_4322816_scn_A24_leg11_3s.cal.fits' /  Input files used to cr
EFILE2  = 'mips_AOR_4322816_scn_A24_leg12_3s.cal.fits' /  Input files used to cr
EFILE3  = 'mips_AOR_4322816_scn_A24_leg13_3s.cal.fits' /  Input files used to cr
EFILE4  = 'mips_AOR_4322816_scn_A24_leg1_3s.cal.fits' /  Input files used to cre
EFILE5  = 'mips_AOR_4322816_scn_A24_leg14_3s.cal.fits' /  Input files used to cr
EFILE6  = 'mips_AOR_4322816_scn_A24_leg15_3s.cal.fits' /  Input files used to cr
EFILE7  = 'mips_AOR_4322816_scn_A24_leg16_3s.cal.fits' /  Input files used to cr
EFILE8  = 'mips_AOR_4322816_scn_A24_leg17_3s.cal.fits' /  Input files used to cr
EFILE9  = 'mips_AOR_4322816_scn_A24_leg18_3s.cal.fits' /  Input files used to cr
EFILE10 = 'mips_AOR_4322816_scn_A24_leg19_3s.cal.fits' /  Input files used to cr
EFILE11 = 'mips_AOR_4322816_scn_A24_leg20_3s.cal.fits' /  Input files used to cr
EFILE12 = 'mips_AOR_4322816_scn_A24_leg21_3s.cal.fits' /  Input files used to cr
EFILE13 = 'mips_AOR_4322816_scn_A24_leg22_3s.cal.fits' /  Input files used to cr
EFILE14 = 'mips_AOR_4322816_scn_A24_leg23_3s.cal.fits' /  Input files used to cr
EFILE15 = 'mips_AOR_4322816_scn_A24_leg2_3s.cal.fits' /  Input files used to cre
EFILE16 = 'mips_AOR_4322816_scn_A24_leg24_3s.cal.fits' /  Input files used to cr
EFILE17 = 'mips_AOR_4322816_scn_A24_leg25_3s.cal.fits' /  Input files used to cr
EFILE18 = 'mips_AOR_4322816_scn_A24_leg3_3s.cal.fits' /  Input files used to cre
EFILE19 = 'mips_AOR_4322816_scn_A24_leg4_3s.cal.fits' /  Input files used to cre
EFILE20 = 'mips_AOR_4322816_scn_A24_leg5_3s.cal.fits' /  Input files used to cre
EFILE21 = 'mips_AOR_4322816_scn_A24_leg6_3s.cal.fits' /  Input files used to cre
```

```
EFILE22 = 'mips_AOR_4322816_scn_A24_leg7_3s.cal.fits' /  Input files used to cre
EFILE23 = 'mips_AOR_4322816_scn_A24_leg8_3s.cal.fits' /  Input files used to cre
EFILE24 = 'mips_AOR_4322816_scn_A24_leg9_3s.cal.fits' /  Input files used to cre
COMMENT ******************************************************************
COMMENT  Plane 1: Signal
COMMENT  Plane 2: Uncertainty in signal
COMMENT  Plane 3: Data flag
COMMENT  Plane 4: Total Overlap
```

Para el resto de lo solicitado, escribimos

```
print 'El número de pixeles Nx es:', header['NAXIS1']
print 'El número de pixeles Ny es:', header['NAXIS2']
print 'La escala de ángulo de cada pixel Nx en arcsec es:', header['XPIXSIZE']
print 'La escala de ángulo de cada pixel Ny en arcsec es:', header['YPIXSIZE']
print 'La coordenada referencial Nx de la imagen es:', header['CRPIX1']
print 'La coordenada referencial Ny de la imagen es:', header['CRPIX2']
```

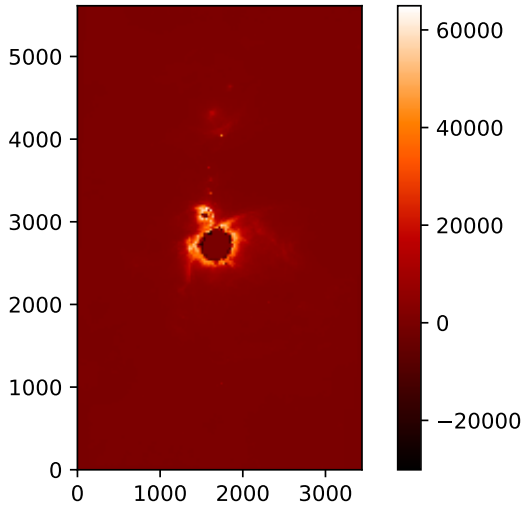
out:

```
El numero de pixeles Nx es: 3442
El numero de pixeles Ny es: 5614
La escala de angulo de cada pixel Nx en arcsec es: 1.24500000477
La escala de angulo de cada pixel Ny en arcsec es: 1.24500000477
La coordenada referencial Nx de la imagen es: 1716.5
La coordenada referencial Ny de la imagen es: 2822.5
```

Hagan una figura en color de la imagen.

Cargamos los datos del archivo FITS con `data = hdulist[0].data` y ploteamos con

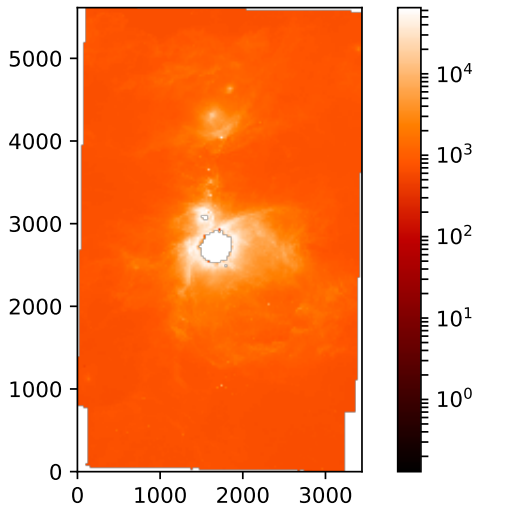
```
plt.imshow(data, origin='lower', cmap= plt.cm.gist_heat)
plt.colorbar()
plt.show()
```



Hagan una figura en color de la imagen, pero a escala logarítmica en los flujos.

Con el paquete `LogNorm` cargado, escribimos:

```
plt.imshow(data, origin='lower', cmap= plt.cm.gist_heat, norm=LogNorm())
plt.colorbar()
plt.show()
```



Ejercicio 2

Modifiquen su programa de las pelotitas para que:

Se inicialize siempre con la misma distribución (aleatoria) de posiciones, pero que las velocidades cambien cada vez que corran el programa.

```
#-*-coding:utf-8

import numpy as np
import matplotlib.pyplot as plt

#Creacion de ejes
plt.axis([-10,10,-10,10,])
#n bolas (10 bolas en este caso)
n     = 10
#La velocidad cambiará cada vez que se ejecuta el programa
vel   = (0.3* np.random.normal(size=n*2)).reshape(n,2)
np.random.seed(0) #se fija el seed del generador
#los numeros siguen siendo aleatorios pero con el seed la distribución es la misma
pos   = (20 * np.random.sample(n*2)-10).reshape(n,2)
#tamaño del radio de las bolas
sizes = 100 * np.random.sample(n)+100
#crea 10 bloques de datos de 1 fila y 4 columnas cada uno
colors = np.random.sample([n,3])
circles = plt.scatter(pos[:,0],pos[:,1],marker = 'o',s = sizes,c=colors)
#se realizarán 100 iteraciones
for i in range(100):
        pos = pos + vel
        bounce = abs(pos) > 10
        vel[bounce] = -vel[bounce]
        circles.set_offsets(pos)
        plt.pause(0.05)
```

Incluyan un "campo de gravedad" en el centro de la caja (con el centro en 0,0) modificando las velocidades directamente en el código.

Para el desarrollo de este ejercicio tuve la ayuda de **Nicolás Sandoval.**

```
#-*-coding: utf-8

import numpy as np
import matplotlib.pyplot as plt

#Configuración de ejes
with plt.xkcd():
    plt.axis([-10,10,-10,10])
    #Se añadirá "punto con gravedad"
        plt.plot(0,0, marker ='*',c='r')
    #Para mayor elegancia (no caos), elegimos 1 bolita
    n = 1
    pos = (20*np.random.sample(n*2) - 10).reshape(n,2)
    vel = (0.1*np.random.normal(size=n*2)).reshape(n,2)
    sizes = 100*np.random.sample(n)+100
    circles = plt.scatter(pos[:,0],pos[:,1],marker = 'o', s=sizes, c='g')
    for i in range(100):
```

```
        #distancia al centro
        dis = np.sqrt(pos[:,0]**2 + pos[:,1]**2)
        #aceleracion (reducido en su decima parte)
        ace = (-1/(15 + dis**2))/0.1
        a_x = (ace)*pos[:,0]
        a_y = (ace)*pos[:,1]
        #velocidades
        vel[:,0] = vel[:,0] + a_x
        vel[:,1] = vel[:,1] + a_y
        pos = pos + vel
        bounce = abs(pos)>10
        circles.set_offsets(pos)
            plt.pause(0.05)
```

pinchando este link podremos ver un video con el resultado del código.

Hagan lo mismo que en (2) pero usando una función para calcular las velocidades nuevas en cada iteración.

```
#-*-coding: utf-8

import numpy as np
import matplotlib.pyplot as plt

#funcion para vel en eje x
def vel_x(x):
        return x + a_x
#funcion para vel en eje y
def vel_y(y):
        return y + a_y

#Configuración de ejes
with plt.xkcd():
        plt.axis([-10,10,-10,10])
#Se añadirá "punto con gravedad"
        plt.plot(0,0, marker ='*',c='r')
#Para mayor elegancia (no caos), elegimos 1 bolita
n = 1
pos = (20*np.random.sample(n*2) - 10).reshape(n,2)
vel = (0.1*np.random.normal(size=n*2)).reshape(n,2)
sizes = 100*np.random.sample(n)+100
circles = plt.scatter(pos[:,0],pos[:,1],marker = 'o', s=sizes, c='g')
for i in range(100):
        x = vel[:,0]
        y = vel[:,1]
        #distancia al centro
        dis = np.sqrt(pos[:,0]**2 + pos[:,1]**2)
        #aceleracion (reducido en su decima parte)
        ace = (-1/(15 + dis**2))/0.1
        a_x = (ace)*pos[:,0]
        a_y = (ace)*pos[:,1]
        #velocidades
        vel[:,0] = vel_x(x)
        vel[:,1] = vel_y(y)
        pos = pos + vel
        bounce = abs(pos)>10
```

```
        circles.set_offsets(pos)
        #imprime cada velocidad
        print '{} , {}'.format(vel_x(x),vel_y(y))
        with plt.xkcd():
                plt.pause(0.05)
```

Finalmente, hagan un gráfico de contornos (contour plot) de su campo de gravedad.

```
#-*-coding: utf-8

import numpy as np
import matplotlib.pyplot as plt

#Configuración de ejes
with plt.xkcd():
        plt.axis([-10,10,-10,10])
#Se añadirá "punto con gravedad"
        plt.plot(0,0, marker ='*',c='r')
xx  = np.linspace(-10,10,100)
yy  = np.linspace(-10,10,100)
X,Y = np.meshgrid(xx,yy)
D = np.sqrt(X**2 + Y**2)
Z = -1./(15 + D**2)
cont = plt.contour(X,Y,Z,10)
#Para mayor elegancia (no caos), elegimos 1 bolita
n = 1
pos = (20*np.random.sample(n*2) - 10).reshape(n,2)
vel = (0.1*np.random.normal(size=n*2)).reshape(n,2)
sizes = 100*np.random.sample(n)+100
circles = plt.scatter(pos[:,0],pos[:,1],marker = 'o', s=sizes, c='g')

for i in range(100):
        #distancia al centro
        dis = np.sqrt(pos[:,0]**2 + pos[:,1]**2)
        #aceleracion (reducido en su decima parte)
        ace = (-1/(15 + dis**2))/0.1
        a_x = (ace)*pos[:,0]
        a_y = (ace)*pos[:,1]
        #velocidades
        vel[:,0] = vel[:,0] + a_x
        vel[:,1] = vel[:,1] + a_y
        pos = pos + vel
        bounce = abs(pos)>10
        circles.set_offsets(pos)
        with plt.xkcd():
                plt.pause(0.05)
```

Ejercicio 3

Usen urllib2 para bajar datos del www. Aquí hay datos en formato FITS: (los archivos FITS terminan en fit.gz, donde gz significa que están comprimidos). Usen la imagen w0ck0101t_c0h.fit.gz. Usen urllib2.urlopen(), urlptr.read(), urlptr.close(), y open, write, write y close para bajar el archivo y escribirlo en formato fits a disco. usen os.listdir para verificar si el archivo se bajo bien. Abran el archivo, y hagan un plot de la imagen (en escala "log" se vera mejor). Que es el objecto en la imagen?

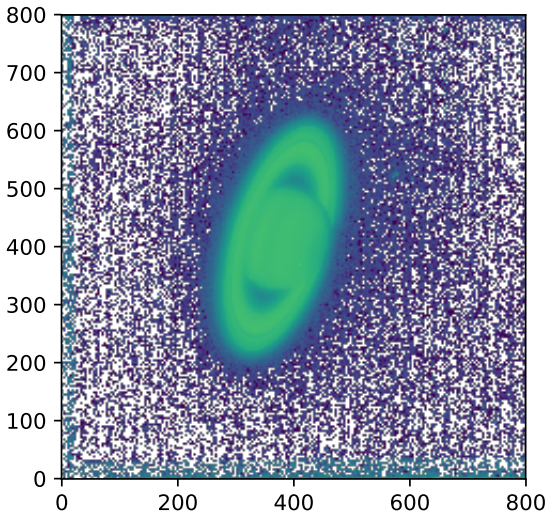
Gracias a **Ibhar** y a **Franco** por ayudarme en el desarrollo de este ejercicio.

```
import urllib2
from astropy.io import fits
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import os


#link hacia el archivo
link = ('https://fits.gsfc.nasa.gov/nrao_data/samples/hst/w0ck0101t_c0h.fit.gz')
#lo renombramos
name = ('archivo.gz')
#lo descargamos
desc = urllib2.urlopen(link)

save = file(name,'w')
#escribimos mientras se descarga el archivo
save.write(desc.read())
save.close()
#verificamos el archivo descargado
print os.listdir('.')

hdu = fits.open('archivo.gz')
data = hdu[0].data[0,:,:]
plt.imshow(data,origin='lower',norm=LogNorm())
```



El objeto es el hermosísimo **Saturno**