Importando datos en Python

November 5, 2017

1 Introducción y archivos planos

A continuación se va a mostrar como importar datos en Python de ficheros planos(txt,csv), se asume que el lector tiene conocimientos de la librería Numpy y Pandas, en caso de no ser así dispone de información de estas librerías en la sección de Numpy y Pandas.

2 Explorando nuestro directorio

Antes de comenzar a trabajar con ficheros, lo primero que debemos hacer es analizar y ver nuestro espacion de trabajo con el fin de estar seguros de que en el se encuentran los fichero necesarios. Para ello Python cuenta con el comando !ls, que nos muestra todos los ficheros que tenemos en nuestro espacio de trabajo.

```
In [1]: !ls
```

```
airquality.csv
Aprendiendo a Programar en Python (Parte II).ipynb
baseball.csv
cars.csv
Cleaning Data in Python .ipynb
Descargas
DOB_Job_Application_Filings.csv
Documentos
ebola_data_db_format.csv
ebola_melt.csv
Empezando a Programar en Python.ipynb
Empezando a programar en Python (Parte 1).ipynb
Escritorio
examples.desktop
gapminder_agg.csv
gapminder.csv
Imágenes
Importando datos en Python .ipynb
Importando Datos en Python .ipynb
jre1.8.0_144
jre-8u144-linux-x64.tar.gz
```

```
Matplotlib.ipynb
moby_dick.txt
Música
Numpy.ipynb
Pandas.ipynb
pandoc-1.17.0.3
pandoc-1.17.0.3.tar.gz
Plantillas
Público
Python Data Science Toolbox (Part 1).ipynb
spark-2.2.0-bin-hadoop2.7.tgz
state.csv
status_country.csv
survey.csv
tb.csv
Time_Series_Pandas.py
tips.csv
tips_modificado.csv
tips_totaldollar.csv
Tuplas&Diccionarios.ipynb
tweets.csv
Uber1.csv
Uber2.csv
Uber3.csv
Untitled Folder
Untitled.ipynb
Vídeos
visited2.csv
visited.csv
```

3 Importando ficheros de texto

En este ejercicio vamos a trabajar con el fichero de texto moby_dick.txt. Para poder crear una conexión a este fichero y así poder trabajar con este, Python dispone de la función open(), la cual sus dos parámetros principales es el path donde se encuentra el fichero que queremos abrir y luego el parámetro mode, que puede tomar los valores de r(si queremos abrir el fichero en modo lectura) o w(si queremos abrir el fichero en modo escritura). Tras crear la conexión, para ver su contenido Python cuenta con el método read(). Una buena práctica es una vez hemos terminado es la de cerrar la conexión con el fichero, para esto Python dispone del método .close().

```
In [4]: # Nos creamos una conexión al fichero
        file = open('moby_dick.txt', mode = 'r')
        #Vemos el contenido del fichero
        print(file.read())
```

```
CHAPTER 1. Loomings.
```

```
Call me Ishmael. Some years ago--never mind how long precisely--having
little or no money in my purse, and nothing particular to interest me on
shore, I thought I would sail about a little and see the watery part of
the world. It is a way I have of driving off the spleen and regulating
the circulation. Whenever I find myself growing grim about the mouth;
whenever it is a damp, drizzly November in my soul; whenever I find
myself involuntarily pausing before coffin warehouses, and bringing up
the rear of every funeral I meet; and especially whenever my hypos get
such an upper hand of me, that it requires a strong moral principle to
prevent me from deliberately stepping into the street, and methodically
knocking people's hats off--then, I account it high time to get to sea
as soon as I can. This is my substitute for pistol and ball. With a
philosophical flourish Cato throws himself upon his sword; I quietly
take to the ship. There is nothing surprising in this. If they but knew
it, almost all men in their degree, some time or other, cherish very
nearly the same feelings towards the ocean with me.
```

```
In [5]: #Cerramos el fichero
        file.close()
        #Comprobamos que la conexión ha sido cerrada
        print(file.closed)
```

```
True
```

4 Importando un fichero línea a línea

Podemos tener situaciones en las cuales disponemos de un fichero de texto muy extenso y solo estamos interesados en imprimir las primeras líneas, para ello Python cuenta con el método readline(). Cuando ejecutamos por primera vez el comando readline() la primera línea del fichero es mostrada, si lo ejecutamos una segunda vez la segunda línea es mostrado y así sucesivamente.

```
In [7]: #Nos creamos la conexión al fichero y imprimimos las 4 primeras líneas
        with open('moby_dick.txt', mode = 'r') as file:
            print(file.readline())
            print(file.readline())
            print(file.readline())
            print(file.readline())
```

```
CHAPTER 1. Loomings.
```

```
Call me Ishmael. Some years ago--never mind how long precisely--having
```

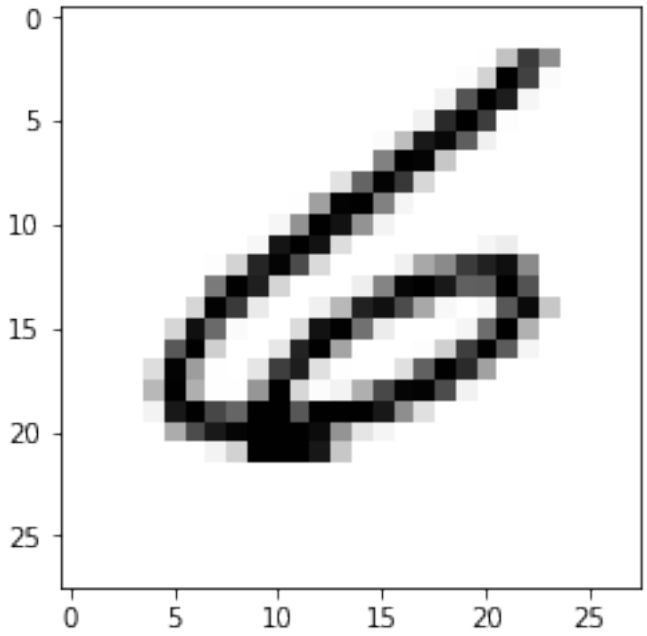
5 Importando datos con Numpy

Cuando estamos trabajando con ficheros que únicamente contienen datos numéricos, la mejor opción para poder trabajar con estos es hacer uso del paquete Numpy, debido a que automáticamente se nos cargarán en formato array y podremos manipularlos de forma rápida y eficiente. Para ello Numpy cuenta con la función loadtxt(), esta función recibe como parámetros principales el path donde se encuentra el fichero que queremos leer y el delimitador de nuestro fichero de datos.

```
In [3]: #Importamos la librería Numpy y matplotlib
        import numpy as np
        import matplotlib.pyplot as plt
        #Cargamos nuestro fichero
        file = np.loadtxt('mnist_kaggle_some_rows.csv', delimiter = ',')
        #Mostramos el tipo de dato que es file
        print(type(file))
```

```
<class 'numpy.ndarray'>
```

```
In [4]: #Seleccionamos y reescalamos ciertas filas y columnas
        im = file[21,1:]
        im_sq = np.reshape(im, (28,28))
        #Mostramos el resultado
        plt.imshow(im_sq, cmap = 'Greys', interpolation = 'nearest')
        plt.show()
```



Además de los parámetros indicados anteriormente la función loadtxt() dispone entre otros del parámetro skiprows en el cual le indicamos el número de filas que no queremos leer, esto puede ser útil en caso de que nuestro fichero contenga cabeceras de tipo string ya que un array por defecto no acepta valores que no sean numéricos. También cuenta con el parámetro usecols en el cual le indicamos mediante una lista las columnas (índice) que queremos cargar.

Es posible que tengamos ficheros los cuales disponen de cabeceras, además pueden estar separados por un separador distinto a la coma, en caso de que tengamos ficheros que deseamos cargar junto con la cabecera, loadtxt() dispone del parámetro dtype mediante el cual le indicamos que tipo de dato queremos cargar.

```
In [6]: #Cargamos nuestro conjunto de datos haciendo uso de dtype
        file = np.loadtxt('seaslug.txt', delimiter = '\t', dtype = 'str')
        #Imprimimos la segunda fila
        print(file[1])
```

```
['99' '0.067']
```

Podemos oservar como los datos han sido almacenados como string y no como numéricos.

```
In [12]: #Cargamos nuestros datos como float
         file = np.loadtxt('seaslug.txt', delimiter = '\t', dtype = 'float', skiprows = 1)
         print(file[1:3])
```

```
[[  9.90000000e+01   1.33000000e-01]
 [  9.90000000e+01   6.70000000e-02]]
```

Cuando queremos importar ficheros que contienen datos numéricos y datos de tipo string, la función loadtxt() no nos sirve, para esto tenemos la función genfromtxt(). Esta función tiene como primer parámetro el path donde se encuentra el fichero que deseamos cargar, el segundo parámetro es el delimitador de los datos, luego dispone del parámetro names mediante el cual le indicamos si nuestros datos contienen cabecera (le asignamos el valor True) o no contienen cabecera (le asignamos el valor False), finalmente dispone del parámetro dtype, que en este caso puede tomar el valor None, con lo que le indcamos que vamos a cargar datos de tipo mixto.

```
In [15]: #Cargamos nuestro fichero mixto
         file = np.genfromtxt('titanic_sub.csv', delimiter = ',', names = True,
                              dtype = None)
         print(file[0:3])
```

```
[(1, 0, 3, b'male',  22., 1, 0, b'A/5 21171',   7.25  , b'', b'S')
 (2, 1, 1, b'female',  38., 1, 0, b'PC 17599',  71.2833, b'C85', b'C')
 (3, 1, 3, b'female',  26., 0, 0, b'STON/O2. 3101282',   7.925 , b'', b'S')]
```

En este caso podemos ver como las variables que son numéricas son cargadas como tal, mientras que aquellas que son de tipo string son cargadas como string.

Finalmente Numpy también dispone de la función recfromcsv() que se trata de una función que tiene el mismo efecto que genfromtxt(), la única diferencia es que se trata de una función que por defecto el parámetro delimiter toma el valor de coma, la variable names toma el valor de True y dtype el valor de None, por lo que en caso de que queramos cargar un fichero con estas características solo le debemos pasar el path del fichero como parámetro.

```
In [17]: #Cargamos nuestro fichero con la función recfromcsv()
         file = np.recfromcsv('titanic_sub.csv')
         print(file[0:3])
```

```
[(1, 0, 3, b'male',  22., 1, 0, b'A/5 21171',   7.25  , b'', b'S')
 (2, 1, 1, b'female',  38., 1, 0, b'PC 17599',  71.2833, b'C85', b'C')
 (3, 1, 3, b'female',  26., 0, 0, b'STON/O2. 3101282',   7.925 , b'', b'S')]
```

6 Importando datos con Pandas

En Python la estructura más adecuada para almacenar datos son los DataFrame, para importar nuestros datos como DataFrame disponemos de la librería pandas. Pandas dispone de la función read_csv() que nos permite importar datos de forma sencilla. El métedo head() de un dataframe nos permite ver las 5 primeras filas de los datos cargados.

```
In [2]: #Importamos Pandas
        import pandas as pd
        #Cargamos nuestro conjunto de datos
        df = pd.read_csv('titanic_sub.csv')
        #Mostramos las 5 primeras filas
        print(df.head())
```

```
   PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch  \
0            1         0       3    male  22.0      1      0
1            2         1       1  female  38.0      1      0
2            3         1       3  female  26.0      0      0
3            4         1       1  female  35.0      1      0
4            5         0       3    male  35.0      0      0

             Ticket     Fare Cabin Embarked
0         A/5 21171   7.2500   NaN        S
1          PC 17599  71.2833   C85        C
2  STON/O2. 3101282   7.9250   NaN        S
3            113803  53.1000  C123        S
4            373450   8.0500   NaN        S
```

Este DataFrame también dispone del método values, este método nos permite extraer los datos, es decir, lo que hacemos es convertir nuestro DataFrame a numpyarray.

```
In [3]: #Importamos como DataFrame haciendo uso de pandas
        df = pd.read_csv('mnist_kaggle_some_rows.csv', header = None, nrows = 5)
        print(df.head())
```

```
   0    1    2    3    4    5    6    7    8    9   ...   775  776  777  778  \
0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 ...   0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 ...   0.0  0.0  0.0  0.0
2  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 ...   0.0  0.0  0.0  0.0
3  4.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 ...   0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 ...   0.0  0.0  0.0  0.0

   779  780  781  782  783  784
0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 785 columns]
```

```
In [4]: #Ahora convertimos a numpyarray
        df_np = df.values
        print(df_np)
```

```
[[ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 1.  0.  0. ...,  0.  0.  0.]
 [ 4.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]]
```

7 Importando datos de otros tipos de ficheros

Como DataScientist debemos saber importar datos de una gran cantidad de tipos de ficheros, a continuación se explicará la forma de exportar datos de ficheros de tipo excel, Matlab, SAS, HDF5 y otros tipos.

8 Introducción a la librería os

Ya hemos visto como el comando !ls nos permite visualizar los ficheros que se encuentra en nuestro directorio de trabajo, sin embargo, Python dispone de la librería os, que se trata de una librería bastante completa. Entre sus métodos más importantes se encuentra el método getcwd() que nos devuelve el directorio de trabajo y el método listdir() que nos retorna el número de ficheros en el directorio que le indiquemos.

```
In [1]: #Importamos la librería os
        import os
        #Accedemos a nuestro espacio trabajo
        wd = os.getcwd()
        #Vemos los ficheros en nuestro espacion de trabajo
        os.listdir(wd)
```

```
Out[1]: ['.bash_logout',
         'spark-2.2.0-bin-hadoop2.7.tgz',
         'Python Data Science Toolbox (Part 1).ipynb',
         'Time_Series_Pandas.py',
         '.gitconfig',
         'Escritorio',
         '.ghc',
         'cars.csv',
         '.macromedia',
         '.vboxclient-clipboard.pid',
         'jre-8u144-linux-x64.tar.gz',
         '.xsession-errors.old',
         'visited2.csv',
         '.xsession-errors',
         '.ssh',
         'pandoc-1.17.0.3.tar.gz',
         '.cache',
         'tips_totaldollar.csv',
         'Público',
         'airquality.csv',
         'Empezando a programar en Python (Parte 1).ipynb',
         'ebola_data_db_format.csv',
         'Plantillas',
         'Aprendiendo a Programar en Python (Parte II).ipynb',
         '.vboxclient-draganddrop.pid',
         '.cabal',
         'Untitled Folder',
```

```
'Música',
'Imágenes',
'.gnupg',
'.thunderbird',
'state.csv',
'Cleaning Data in Python .ipynb',
'.jupyter',
'.sudo_as_admin_successful',
'gapminder_agg.csv',
'Tuplas&Diccionarios.ipynb',
'Untitled.ipynb',
'Uber3.csv',
'pandoc-1.17.0.3',
'.config',
'tips.csv',
'.bashrc',
'titanic_sub.csv',
'Descargas',
'gapminder.csv',
'.ipython',
'Matplotlib.ipynb',
'.bash_history',
'.ipynb_checkpoints',
'seaslug.txt',
'.adobe',
'.oracle_jre_usage',
'.mozilla',
'.profile',
'Documentos',
'Uber2.csv',
'.stack',
'status_country.csv',
'.~Empezando a programar en Python (Parte 1).ipynb',
'baseball.csv',
'Vídeos',
'Numpy.ipynb',
'tb.csv',
'tips_modificado.csv',
'.dmrc',
'visited.csv',
'mnist_kaggle_some_rows.csv',
'tweets.csv',
'Pandas.ipynb',
'.~Cleaning Data in Python .ipynb',
'jre1.8.0_144',
'survey.csv',
'.gconf',
'moby_dick.txt',
```

```
 'DOB_Job_Application_Filings.csv',
 'ebola_melt.csv',
 '.vboxclient-display.pid',
 '.Xauthority',
 '.vboxclient-seamless.pid',
 'Importando datos en Python .ipynb',
 '.idlerc',
 '.ICEauthority',
 '.local',
 'examples.desktop',
 'Importando Datos en Python .ipynb',
 '.nano',
 'Empezando a Programar en Python.ipynb',
 'Uber1.csv']
```

9 Guardando y Cargando Ficheros pickled

Existen datos que no pueden ser almacenados de forma sencilla en ficheros planos, como por ejemplo las listas y los diccionarios. Si queremos almacenar estos datos en ficheros que pueden ser leídos por el usuario, deberíamos guardar estos datos en ficheros de tipo JSON que son los ficheros más apropiados para los diccionarios. Sin embargo, si únicamente queremos importar estos datos en Python, podemos convertirlos en una secuencia de Bytes, a esto se le conoce como serializar los datos. Para ello disponemos del paquete pickle.

```
In [3]: #Importamos la librería pickle
        import pickle
        #Escribimos un diccionario a un archivo pkl
        mydict = {'a': 1, 'b': 2, 'c': 3}
        output = open('myfile.pkl', 'wb')
        pickle.dump(mydict, output)
        output.close()
        #Realizamos ahora la lectura de nuestro archivo pickle
        with open('myfile.pkl', 'rb') as file:
            d = pickle.load(file)

        print(d)
```

```
{'a': 1, 'c': 3, 'b': 2}
```

10 Importando ficheros excel

Como científico de datos es muy probable que alguna vez te tengas que enfrentar a situaciones en las cuales los datos se encuentren almacenados en ficheros de tipo excel. Pandas nos permite esto, antes de comenzar a trabajar con un fichero de excel, debemos ser conscientes del número de hojas que contiene este archivo.

```
In [1]: #Importamos pandas
        import pandas as pd
        #Cargamos nuestro hoja de cálculo
        xl = pd.ExcelFile('battledeath.xlsx')
        #Mostramos las hojas que tiene
        print(xl.sheet_names)
```

```
['2002', '2004']
```

Podemos ver como nuestro fichero cuenta con dos hojas. Ahora ha llegado el momento de cargar estas dos hojas.

```
In [2]: #Cargamos la hoja 2002
        df1 = xl.parse('2002')
        #Mostramos las 5 primeras filas
        print(df1.head())
```

```
  War, age-adjusted mortality due to       2002
0                        Afghanistan  36.083990
1                            Albania   0.128908
2                            Algeria  18.314120
3                            Andorra   0.000000
4                             Angola  18.964560
```

También podemos acceder a la hoja por índice en lugar de por nombre

```
In [4]: #Accedemos a la primera hoja (2002)
        df2 = xl.parse(0)
        #Mostramos las 5 primeras filas
        print(df2.head())
```

```
  War, age-adjusted mortality due to       2002
0                        Afghanistan  36.083990
1                            Albania   0.128908
2                            Algeria  18.314120
3                            Andorra   0.000000
4                             Angola  18.964560
```

La función parse nos permite incorporar una serie de parámetros adicionales que nos mejoran la carga de los datos.

```
In [23]: #Importamos la hoja 2002
         df1 = xl.parse(0,names = ['Country', 'AAM due to war (2002)'], skiprows = 0)
         #Mostramos las 5 primeras filas
         print(df1.head())
```

```
       Country  AAM due to war (2002)
0  Afghanistan              36.083990
1      Albania               0.128908
2      Algeria              18.314120
3      Andorra               0.000000
4       Angola              18.964560
```

También dispone del parámetro parse_cols mediante el cual le indicamos las columnas que deseamos analizar o cargar

```
In [26]: df1 = xl.parse('2002', names = ['Country'], skiprows = 0, parse_cols = [0])
         print(df1.head())
```

```
       Country
0  Afghanistan
1      Albania
2      Algeria
3      Andorra
4       Angola
```

11 Importando ficheros SAS

Los ficheros SAS se tratan de ficheros muy utilizados en economía y en campos como la epidemiología. Para importar este tipo de ficheros Python nos permite hacer uso de la librería sas7bdat.

```
In [4]: #Importamos el paquete
        from sas7bdat import SAS7BDAT
        #Almacenamos el contenido de nuestro fichero en un DataFrame
        with SAS7BDAT('sales.sas7bdat') as file:
            df_sas = file.to_data_frame()
        print(df_sas.head())
```

```
     YEAR     P           S
0  1950.0  12.9  181.899994
1  1951.0  11.9  245.000000
2  1952.0  10.7  250.199997
3  1953.0  11.3  265.899994
4  1954.0  11.2  248.500000
```

Pandas nos permite importar los ficheros de tipo dta, para exportar este tipo de ficheros debemos de hacer uso de la función read_stata().

```
In [2]: #Importamos pandas
        import pandas as pd
        #Importamos los datos
        df_disarea = pd.read_stata('disarea.dta')
        print(df_disarea.head())
```

```
  wbcode               country  disa1  disa2  disa3  disa4  disa5  disa6  \
0    AFG           Afghanistan   0.00   0.00   0.76   0.73    0.0   0.00
1    AGO                Angola   0.32   0.02   0.56   0.00    0.0   0.00
2    ALB               Albania   0.00   0.00   0.02   0.00    0.0   0.00
3    ARE  United Arab Emirates   0.00   0.00   0.00   0.00    0.0   0.00
4    ARG             Argentina   0.00   0.24   0.24   0.00    0.0   0.23

   disa7  disa8   ...    disa16  disa17  disa18  disa19  disa20  disa21  \
0   0.00    0.0   ...       0.0     0.0     0.0    0.00    0.00     0.0
1   0.56    0.0   ...       0.0     0.4     0.0    0.61    0.00     0.0
2   0.00    0.0   ...       0.0     0.0     0.0    0.00    0.00     0.0
3   0.00    0.0   ...       0.0     0.0     0.0    0.00    0.00     0.0
4   0.00    0.0   ...       0.0     0.0     0.0    0.00    0.05     0.0

   disa22  disa23  disa24  disa25
0    0.00    0.02    0.00    0.00
1    0.99    0.98    0.61    0.00
2    0.00    0.00    0.00    0.16
3    0.00    0.00    0.00    0.00
4    0.00    0.01    0.00    0.11

[5 rows x 27 columns]
```

12 Importando ficheros HDF5

Cuando necesitamos almacenar cientos de Gigas de información, los ficheros visto hasta ahora no son los más adecuados. Si el tipo de información que queremos almacenar es de tipo numérico los ficheros HDF5 son la mejor opción. Python dispone de una librería que nos permite tratar este tipo de ficheros.

```
In [7]: #Importamos las librerías necesarias
        import numpy as np
        import h5py
        #Cargamos el fichero en modo lectura
        file = h5py.File('L-L1_LOSC_4_V1-1126259446-32.hdf5', 'r')
        print(type(file))
```

```
<class 'h5py._hl.files.File'>
```

Los ficheros HDF5 son diccionarios a los que a cada clave se le asigna información. A continuación vamos a proceder a imprimir las claves.

```
In [8]: for key in file.keys():
            print(key)
```

```
meta
quality
```

```
strain
```

La descripción de cada una de las claves deben de ser dadas por el profesional. A continuación vamos a proceder a explorar la información en strain.

```
In [9]: #Accedemos a la información de la clave strain
        group = file['strain']
        type(group)
```

```
Out[9]: h5py._hl.group.Group
```

Se trata de información de tipo group, por lo que debemos de acceder a cada una de las claves

```
In [11]: #Imprimimos las claves de group
         for key in group.keys():
             print(key)
```

```
Strain
```

A continuación accedemos a los datos

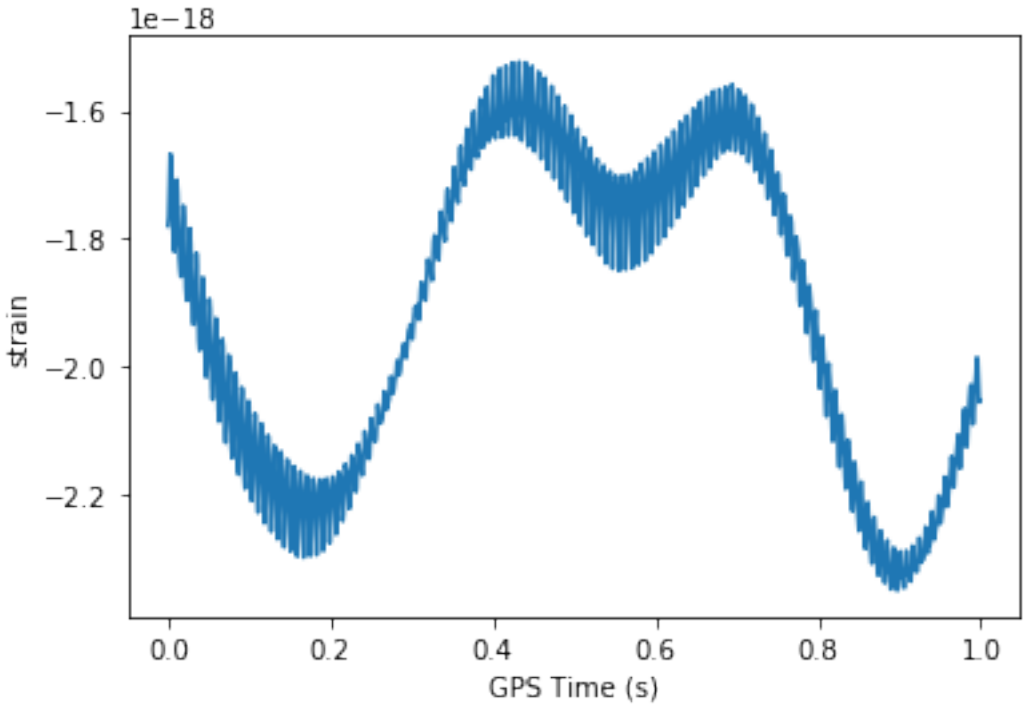
```
In [15]: #Accedemos al valor de los datos
         strain = file['strain']['Strain'].value
         print(type(strain))
         np.shape(strain)
```

```
<class 'numpy.ndarray'>
```

```
Out[15]: (131072,)
```

Podemos ver que estamos tratando con una array con un total de 131072 filas. A continuación vamos a proceder a realizar un simple gráfico con la información de los datos.

```
In [18]: #Importamos las librería matplotlib
         import matplotlib.pyplot as plt
         #Realizamos el gráfico
         num_muestras = 1000
         time = np.arange(0,1,1/num_muestras)
         #Seleccionamos las 1000 primeras muestras de strain
         plt.plot(time, strain[:num_muestras])
         plt.xlabel('GPS Time (s)')
         plt.ylabel('strain')
         plt.show()
```



13 Importando ficheros Matlab

Python dispone de la librería scipy.io que nos permite trabajar con ficheros .mat. Estos ficheros suelen contener información del workspace de Matlab, por lo que pueden contener variables de múltiples tipos (matrices, strings, vectores, etc.). Una vez importamos scipy.io, podemos hacer uso de la funciones .loadmat() y savemat() que nos permiten poder tratar estos ficheros.

```
In [20]: #Importamos el paquete scipy.io
         import scipy.io
         #Leemos el fichero
         file = scipy.io.loadmat('ja_data2.mat')
         print(type(file))
```

```
<class 'dict'>
```

Podemos ver que estamos ante un fichero de tipo diccionario, donde cada clave se trata del nombre de la variable en el workspace y el valor se trata del valor que toma dicha variable.

```
In [21]: #Mostramos las claves del fichero
         print(file.keys())
```

```
dict_keys(['yfpNuc', 'rfpNuc', '__globals__', 'CYratioCyt', 'cfpNuc', 'yfpCyt', '__header__', 'c
```

```
In [25]: #Vemos el tipo de una de las claves
         print(type(file['CYratioCyt']))
         print(np.shape(file['CYratioCyt']))
```

```
<class 'numpy.ndarray'>
(200, 137)
```

Podemos ver que estamos ante array con un total de 200 filas y 237 columnas.