¿Cómo importar un .mat en Python(y hacer que se convierta en una variable)?

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2023/06/23/como-importar-un-mat-en-pythony-hacer-que-se-convierta-en-una-variable/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

Si has llegado aquí es porque tienes un fichero .mat de Matlab y quieres abrirlo con Python. Bien, ahora te cuento como se pueden abrir con un ejemplo de apertura y cómo se crean variables en Python con este valor. Repositorio de código

Primer punto: abrir un fichero .mat y generar variable

Para abrir un fichero .mat se tiene que utilizar la librería scipy de Python, y más concretamente abría que utilizar scipy.io

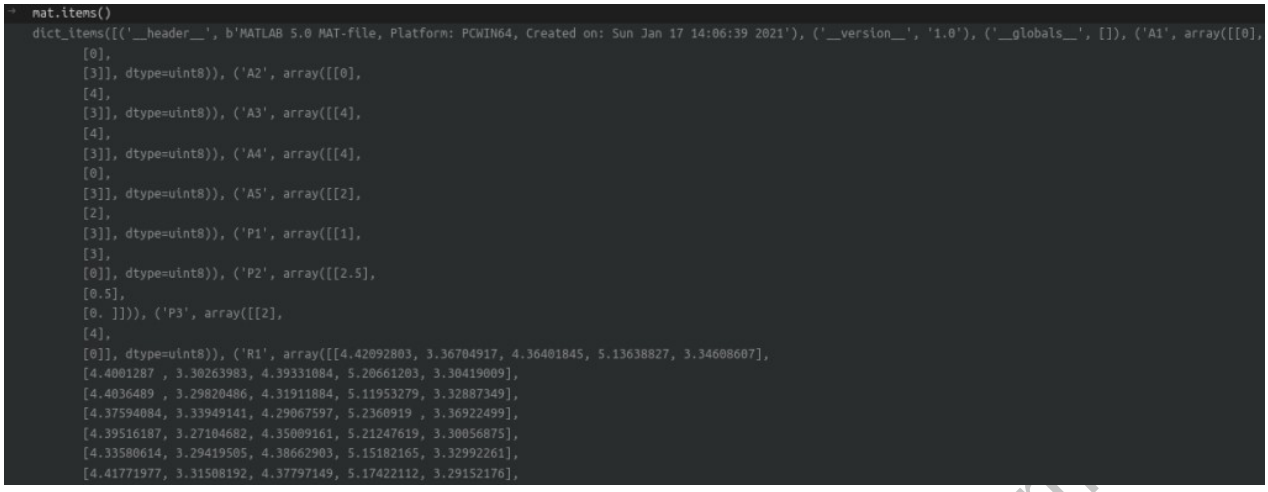
```
import scipy.io
```

Esta librería tiene una función llamada *loadmat*, que es la que se utiliza para leer el fichero.

```
mat = scipy.io.loadmat(file)
```

Esta función devuelve el código interno del .mat. Lo que devuelve es un diccionario, con todos los valores internos.

Para acceder a los campos que tiene se tiene que utilizar el método *items*. Este método desglosa el diccionario.



Ahora para acceder a cada variable se tiene que realizar en un bucle *for* con cada *item*(descartando el segundo valor), y este bucle *for* tiene que hacer uso del método *get*.

```
for k,_ in mat.items():
    print(mat.get(k))
```

Y ahora, una vez se tiene cada valor se tiene que generar la variable. Para ello se utiliza el siguiente código.

```
globals()[f"{k}"] = mat.get(k).tolist()
```

Este código genera una variable global con mismo nombre que la que está guardada en .mat.

Segundo Punto: Hacerlo funcionar

Si has intentado ejecutar cualquier código usando el código anterior habrás visto que no funciona, eso es debido a que los .mat incorporan más información dentro que a Matlab la puede ser útil, pero en Python es información prescindible.

```
('__header__', b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Sun Jan 17
14:06:39 2021')
('__version__', '1.0')
('__globals__', [])
('A1', array([[0],
```

La información prescindible suelen ser las 3 primeras líneas, pero no necesariamente tienen por qué ser las 3 primeras. Sin embargo, si que se sabe que el primer valor de la información prescindible, ésta comienza con unos valores llamados: *__header__*, *__version__* y *__globals__*. Por lo que hay que descartar los *items* que lo contengan.

```
for k,_ in mat.items():
        if k != "__header__" and k != "__globals__" and k != "__version__":
```

Ahora ya casi todo funciona, pero no de forma perfecta, hay que refinar el código.

Para refinar las variables, primero hay que determinar si lo que se tiene en el *item* si estamos ante un valor único de variable, o ante un valor múltiple. Esto se comprueba mirando la forma de los valores obtenidos con *get*.

```
if mat.get(k).shape[0] == 1:    # when list has only a row
```

Si el primer valor del *shape* es 1, significa que estamos ante un valor fila, por lo que se tiene un valor de la forma de Matlab 1xN. Si el *shape* no es 1, estamos ante una variable de la forma MxN, esta forma se puede meter directamente en el variable generada(siempre utilizando un *.tolist*().

```
globals()[f"{k}"] = mat.get(k).tolist()
```

Sin embargo, si estamos en un shape igual a 1, se tiene que saber si el valor de N es 1 o distinto de 1. Esto se debe a que si el valor de N es 1, significa que la variable que se va a crear tiene un solo valor, y a la hora de obtenerlo hay que introducirse en una lista dentro de otra. Pero, si N es distinto de 1, se puede introducir solo la primera lista.

```
if mat.get(k).shape[0] == 1:      # when list has only a row
  if mat.get(k).shape[1] == 1:    # when object has one value
      globals()[f"{k}"] = mat.get(k)[0][0].tolist()
  else:
      globals()[f"{k}"] = mat.get(k)[0].tolist()
```

Y ahora con todos los conceptos explicados se puede leer variables directamente de .mat.

Vuelta de tuerca

Con todo lo que se ha explicado anteriormente se puede generar una función que lea como la función load de Matlab, y además, se le puede añadir la opción de que sólo cargue las variables que el usuario quiera.

```
load("matlab", "A1", "R1")
print("A1: ", A1)
print("R1: ", R1)
```

Por eso te doy aquí mi idea de función para que la puedas utilizar.

```
def load(file, *args):
    '''
    This function loads .mat files to Python like Python variables.
    Input:
        - file : the .mat file you want to load. Doesn't matter if ends in
'.mat' or not.
        - specific_variables : this options load only specific variables
from .mat file.
    '''

    if file[-4:] != ".mat":
        file += ".mat"

    import scipy.io
    mat = scipy.io.loadmat(file)
```

```
    for k,_ in mat.items():
        if k != "__header__" and k != "__globals__" and k != "__version__":

            # specific variable
            if len(args) != 0:
                for i in args:
                    if i == k:
                        if mat.get(k).shape[0] == 1:        # when list has
only a row
                            if mat.get(k).shape[1] == 1:    # when object has
one value
                                globals()[f"{k}"] = mat.get(k)[0][0].tolist()
                            else:
                                globals()[f"{k}"] = mat.get(k)[0].tolist()
                        else:
                            globals()[f"{k}"] = mat.get(k).tolist()

            else:
                # non-specific variable
                if mat.get(k).shape[0] == 1:
                    if mat.get(k).shape[1] == 1:
                        globals()[f"{k}"] = mat.get(k)[0][0].tolist()
                    else:
                        globals()[f"{k}"] = mat.get(k)[0].tolist()
                else:
                    globals()[f"{k}"] = mat.get(k).tolist()
```

Y también, te doy otra función que te permite elegir si quieres pintar por el terminal las variables que se están importando a Python.

```
load_var("matlab.mat", 'on')

def load_var(file, printvar='off', *args):
    '''
    This function loads .mat files to Python like Python variables, and
print the names of
    the import variables.
    Input:
        - file : the .mat file you want to load. Doesn't matter if ends in
'.mat' or not.
        - printvar : this option prints the name of new variable. Two
options:
                    - 'on'  : it prints the name
                    - 'off' : it doesn't print the name
        - specific_variables : this options load only specific variables
from .mat file.
    '''

    if file[-4:] != ".mat":
        file += ".mat"

    import scipy.io
    mat = scipy.io.loadmat(file)
    for k,_ in mat.items():
        if k != "__header__" and k != "__globals__" and k != "__version__":
```

```
            # specific variable
            if len(args) != 0:
                for i in args:
                    if i == k:
                        if mat.get(k).shape[0] == 1:        # when list has
only a row
                            if mat.get(k).shape[1] == 1:    # when object has
one value
                                globals()[f"{k}"] = mat.get(k)[0][0].tolist()
                            else:
                                globals()[f"{k}"] = mat.get(k)[0].tolist()
                        else:
                            globals()[f"{k}"] = mat.get(k).tolist()

            else:
                # non-specific variable
                if mat.get(k).shape[0] == 1:
                    if mat.get(k).shape[1] == 1:
                        globals()[f"{k}"] = mat.get(k)[0][0].tolist()
                    else:
                        globals()[f"{k}"] = mat.get(k)[0].tolist()
                else:
                    globals()[f"{k}"] = mat.get(k).tolist()

            if printvar=='on':
                print(k)
```

Espero haberte sido de ayuda.

Tienes todo el código en el siguiente enlace.