In [1]:

```
import requests
import json

# Bucle de 500 llamadas: llamada inicial
url = "https://api.reliefweb.int/v1/reports?appname=apidoc&limit=1"
result = []

#Realizamos try/catch para detectar errores, realizamos catch de la excepción
try:
    for i in range(500):
        #cada 100 registros sacamos print para ver que vamos bien
        if i%100 == 0: print('100 llamadas realizadas...seguimos...')

        #llamamos con la url
        response = requests.get(url)

        #nos guardamos la respuesta
        data = response.json()
        #listamos la url's que utilizamos que obtenemos de la salida y donde obtendremo
s datos detallados del desastre
        #print(data['data'][0]['href'])

        #nos guardamos la url ddonde tenemos que recuperar los datos detallados del des
astre
        url2 = data['data'][0]['href']

        #llamamos a la url de detalles
        response = requests.get(url2)

        #nos guardamos la respuesta de los detalles para luego poder crear el dataset c
on los datos que nos interesan
        result.append(response.json())

        #url next: siguiente llamada
        url = data['links']['next']['href']

except requests.exceptions.HTTPError as errh:
    print ("Http Error:",errh)
except requests.exceptions.ConnectionError as errc:
    print ("Error Connecting:",errc)
except requests.exceptions.Timeout as errt:
    print ("Timeout Error:",errt)
except requests.exceptions.RequestException as err:
    print ("OOps: Something Else",err)

print('Proceso finalizado:  results contiene %s resultados' %len(results))
```

```
100 llamadas realizadas...seguimos...
100 llamadas realizadas...seguimos...
100 llamadas realizadas...seguimos...
100 llamadas realizadas...seguimos...
100 llamadas realizadas...seguimos...

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-1-e6b28dc71ccf> in <module>
     41     print ("OOps: Something Else",err)
     42
---> 43 print('Proceso finalizado:  results contiene %s resultados' %len(r
esults))

NameError: name 'results' is not defined
```

In [2]:

```
print('Proceso finalizado:  results contiene %s resultados' %len(result))
```

```
Proceso finalizado:  results contiene 500 resultados
```

In [3]:

```
##########################################################################################
############
#
#           Definición de funciones que vamos a utilizar
#
##########################################################################################
##########

#Función para buscar un item dentro de una lista enviada: devuelve True en caso de enco
ntrarlo. En caso contrario False
def search_found(list_search, item_search):
    #inicializamos la variable
    found = False

    #Buscamos dentro de la lista enviada
    for col in list_search:
          if col == item_search: found = True

    #devolvemos el valor
    return found


#Crear columnas: nos han de dar los resultados que queremos tratar
#La utilidad de esta función es crear todas las columnas que podemos encontrarnos en la
s diferentes salidas obtenidas
#de esta manera tendremos todas los campos que necesitaremos guardar
def Create_Columns(w_results):

    #inicializamos variable
    principal_fields = []
    max = len(w_results)

    #Nos recorremos todos los resultados
    for out in range(max):

        #Para cada elemento de cada resultado del campo fields
        for i,item in enumerate(w_results[out]['data'][0]['fields']):

            #Comprobamos si es un diccionario, ya que tenemos varios tipos dentro de
 fields
            if isinstance((w_results[out]['data'][0]['fields'][item]),dict):

                #Recorremos los elementos del diccionario
                for j in w_results[out]['data'][0]['fields'][item]:

                    #Si estamos en el primer registro guardamos todos los elementos
como columnas
                    if out == 0:
                        principal_fields.append(str(f'{item}_{j}'))
                    else:
                        #si no estamos en el primero miramos si las items  que enco
ntramos ya las tenemos
                        # como columna. Para ello llamamos a una función que las mi
ra todas.
                        #si no la tenemos la añadimos (cada resultado puede tener v
ariaciones)
                        if not search_found(principal_fields, str(f'{item}_{j}')):p
rincipal_fields.append(str(f'{item}_{j}'))
```

```
            #miramos las listas también (El resto de campos que no son una cosa ni ot
ra son identificadores)
            elif isinstance((w_results[out]['data'][0]['fields'][item]),list):

                    #Recorremos los elementos de la lista
                    for j in w_results[out]['data'][0]['fields'][item]:

                        # Las listas suelen tener diccionarios dentro de datos que
 nos interesan, por tanto
                        #recorremos los elementos encontrados
                        if isinstance(j,dict):
                            for k in j:
                                #En el caso de 'location' tenemos dos elementos que
nos interesan guardar: lat y lon
                                if k == 'location':
                                    #solo lo hacemos la primera vez porque nos qued
amos solo una vez con las columnas
                                    if out == 0:
                                        principal_fields.append(str(f'{item}_{k}_lo
n'))
                                        principal_fields.append(str(f'{item}_{k}_la
t'))
                                else:
                                    #El resto miramos si las tenemos y si no las te
nemos las guardamos como columnas
                                     if not search_found(principal_fields, str(f'{i
tem}_{k}')):
                                         principal_fields.append(str(f'{item}_
{k}'))

    #devolvemos las columnas
    return principal_fields


#Función para crear filas. Es para elementos donde tenemos más de uno, por ejemplo, en
 desastres nos pueden llegar varios
#de la misma zona, para ello tenemos esta función para añadir en el dataframe tantas fi
las como desastres tengamos (el resto
#de variables son iguales, solo cambian los datos de los desastres)
def Create_rows(table, pd_nrows, item, n_nrow):

    #Recorremos los elementos del item
    for i,x in  enumerate(result[out]['data'][0]['fields'][item]):
           #primer registro
            if i == 0:
                for j in x:
                    pd_nrows[table.columns.get_loc(str(f'disaster_type_{j}'))] = x[j]

                #guardamos la fila del primer desastre
                table.loc[n_nrow] = pd_nrows
                n_nrow += 1
            else:
                for j in x:
                    #solo cambiamos los datos del desastre, por el nuevo desastre
                    pd_nrows[table.columns.get_loc(str(f'disaster_type_{j}'))] = x[j]

                #creamos filas según los desastres que tengamos
                table.loc[n_nrow] = pd_nrows
                n_nrow += 1
```

```
    #DEvolvemos la tabla con las filas y el número de fila último que hemos insertado
    return table, n_nrow
```

In [5]:

```
##########################################################################################
#######
#
#                   Proceso
#
##########################################################################################
#######
#importamos las librerias
import pandas as pd
import numpy as np

#inicializamos variables
nrow = 0    #filas que vamos a crear
df=[]

#Creamos las columnas llamando a la función de creación con el número de resultados que
tenemos (creamos todas las columnas que
#necesitemos)
pd_columns = Create_Columns(result)

#Creamos el dataframe y le añadimos las columnas que tendremos
df = pd.DataFrame(columns=pd_columns)

 #Recorremos todos los resultados obtenidos
for out in range(len(result)):

    # creamos una fila inicializada a espacios con la misma cantidad de columnas que te
nemos
    pd_fila = [' ' for x in range(len(df.columns))]

    #Recorremos de cada resultado los elementos que tenemos en 'fields' que es donde te
nemos los datos guardados
    for i in result[out]['data'][0]['fields']:

            #según tengamos una lista o un diccionario actuaremos de forma diferente. C
omprobamos con isinstance que es
            if isinstance((result[out]['data'][0]['fields'][i]),dict):

                #Recorremos todos los elementos del diccionario
                for j in result[out]['data'][0]['fields'][i]:

                    #guardamos en la posición que le toca por la columna que estamos tr
atando el valor
                    pd_fila[df.columns.get_loc(str(f'{i}_{j}'))] = result[out]['data'][
0]['fields'][i][j]

            #En este caso evaluamos si es una lista (El resto de casos no los evaluamos
ya que son datos que no necesitamos)
            elif isinstance((result[out]['data'][0]['fields'][i]),list):

                # disaster lo trataremos de otra manera, ya que en cada registro podemo
s tener más de un desastre y hemos
                #de crear más de una fila
                if (i != 'disaster_type'):

                    #Recorremos los elementos de la lista
                    for j in result[out]['data'][0]['fields'][i]:

                        #Las listas tienen diccionarios dentro por tanto evalúamos el c
```

```
ontenido de dicho diccionario
                        if isinstance(j,dict):
                            for k in j:
                                #cuando tenemos localización tenemos latitud y longitud
                                if k == 'location':
                                     pd_fila[df.columns.get_loc(str(f'{i}_{k}_lon'))] =
j[k]['lon']
                                     pd_fila[df.columns.get_loc(str(f'{i}_{k}_lat'))] =
j[k]['lat']
                                else:
                                    #En el resto de casos no tenemos doble
                                     pd_fila[df.columns.get_loc(str(f'{i}_{k}'))] = j[k
]


    #A veces no tenemos desastres
    if 'disaster_type' in result[out]['data'][0]['fields']:
       df, nrow = Create_rows(df, pd_fila, 'disaster_type', nrow)
    else:
        df.loc[nrow] = pd_fila
        nrow += 1

#mostramos el dataframe creado (solo algunos registros)
df.head()
```

Out[5]:

	date_changed	date_created	date_original	country_hre
0	2017-10-13T06:15:02+00:00	2017-10-13T02:09:29+00:00	2017-10-12T00:00:00+00:00	https://api.reliefweb.int/v1/countries/11
1	2017-10-13T05:35:04+00:00	2017-10-13T02:09:29+00:00	2017-10-12T00:00:00+00:00	https://api.reliefweb.int/v1/countries/16
2	2018-12-21T07:58:02+00:00	2018-12-21T01:59:27+00:00	2018-12-20T00:00:00+00:00	https://api.reliefweb.int/v1/countries/25
3	2018-12-21T00:30:08+00:00	2018-11-17T00:00:00+00:00	2018-11-17T00:00:00+00:00	https://api.reliefweb.int/v1/countries/24
4	2017-10-13T06:14:07+00:00	2017-10-13T01:24:49+00:00	2017-10-12T00:00:00+00:00	https://api.reliefweb.int/v1/countries/18

5 rows × 67 columns

In [8]:

```
#Guardamos el dataframe creadone csv
filename = str(len(result)) + '_registros_desastres.csv'
df.to_csv(filename,  sep='\t', encoding='utf-8')
print('Guardado csv: ', filename)

print('\n\n Ya tienes tus tabla\n\n --\t DISFRUTA  ---')
```

```
Guardado csv:  500_registros_desastres.csv


 Ya tienes tus tabla

 --      DISFRUTA  ---
```