```
import pandas as pd
from functools import partial
import numpy as np
from multiprocessing import Pool
import os
import matplotlib.pyplot as plt


class miDataFrame():

    def __init__(self,url=[],n_cores=1,emisiones=pd.DataFrame):
        self.url = url
        self.n_cores = n_cores
        self.emisiones = emisiones
        self.contadorFiltro = 0
        self.contadorReestructurar = 0
        self.tryfechas = 0
        self.dropvalidador=0
        # Códigos de las magnitudes contaminantes medidas
        self.magnitudes = {
            '1':'Dióxido de Azufre',
            '6':'Monóxido de Carbono',
            '7':'Monóxido de Nitrógeno',
            '8':'Dióxido de Nitrógeno',
            '9':'Partículas < 2.5 µm',
            '10':'Partículas < 10 µm',
            '12':'Óxidos de Nitrógeno',
            '14':'Ozono',
            '20':'Tolueno',
            '30':'Benceno',
            '35':'Etilbenceno',
            '37':'Metaxileno',
            '38':'Paraxileno',
            '39':'Ortoxileno',
            '42':'Hidrocarburos totales(hexano)',
            '43':'Metano',
            '44':'Hidrocarburosno metánicos (hexano)'
        }

        # Códigos de las estaciones de medición.
        self.estaciones = {
            '1':'Pº. Recoletos',
            '2':'Glta. de Carlos V',
            '35':'Pza. del Carmen',
            '4':'Pza. de España',
            '39':'Barrio del Pilar',
            '6':'Pza. Dr. Marañón',
            '7':'Pza. M. de Salamanca',
            '8':'Escuelas Aguirre',
            '9':'Pza. Luca de Tena',
            '38':'Cuatro Caminos',
            '11':'Av. Ramón y Cajal',
            '12':'Pza. Manuel Becerra',
            '40':'Vallecas',
            '14':'Pza. Fdez. Ladreda',
            '15':'Pza. Castilla',
```

```
            '16':'Arturo Soria',
            '17':'Villaverde Alto',
            '18':'Calle Farolillo',
            '19':'Huerta Castañeda',
            '36':'Moratalaz',
            '21':'Pza. Cristo Rey',
            '22':'Pº. Pontones',
            '23':'Final C/ Alcalá',
            '24':'Casa de Campo',
            '25':'Santa Eugenia',
            '26':'Urb. Embajada (Barajas)',
            '27':'Barajas',
            '47':'Méndez Álvaro',
            '48':'Pº. Castellana',
            '49':'Retiro',
            '50':'Pza. Castilla',
            '54':'Ensanche Vallecas',
            '55':'Urb. Embajada (Barajas)',
            '56':'Plaza Elíptica',
            '57':'Sanchinarro',
            '58':'El Pardo',
            '59':'Parque Juan Carlos I',
            '60':'Tres Olivos'
        }
    def mapcontaminantes(self,x):
        if(isinstance(x,int)):
            return self.magnitudes.get(str(x),"Error de entrada")
        elif(isinstance(x,str)):
            return self.magnitudes.get(x,"Error de entrada")

    def generarDF(self):
        pool = Pool(self.n_cores)
        self.emisiones = pd.concat(pool.map(partial(pd.read_csv, sep=';'),self.url))

    def filtrarDataFrame(self):
        if(self.contadorFiltro ==0):
            def genListaFilt():
                r=range(1,32)
                return list(map(lambda x: f'V0{x}' if x<10 else f'V{x}',r))+['PUNTO_MUESTREO']
            self.emisiones = self.emisiones.iloc[:,2:]
            self.emisiones.drop(genListaFilt(), axis='columns', inplace=True)
            self.contadorFiltro +=1
        else:
            print("No puedes filtrar nuevamente.")

    def reestructurarDF(self):
        if(self.contadorReestructurar==0):
            self.emisiones = self.emisiones.melt(id_vars=['ESTACION', 'MAGNITUD', 'ANO', 'MES'
                                                 var_name='DIA', value_name='VALOR')
            self.contadorReestructurar+=1
        else:
             print("Ya esta en el formato correcto.")

    def agregarFecha(self):
        if(self.tryfechas==0):
            self.emisiones['DIA'] = self.emisiones.DIA.str.strip('D')
            self.emisiones['FECHA'] = self.emisiones.ANO.apply(str) + '/' + self.emisiones.MES
            self.emisiones['FECHA'] = pd.to_datetime(self.emisiones.FECHA, format='%Y/%m/%d',
```

```
            self.tryfechas+=1
        else:
             print("Ya tienes el formato de fechas creado.")


    def delNotValid(self):
        if(self.dropvalidador==0):
            print("Estoy Eliminando valores no validos.\n")
            self.emisiones = self.emisiones.drop(self.emisiones[np.isnat(self.emisiones.FECHA)
            self.emisiones.sort_values(['ESTACION', 'MAGNITUD', 'FECHA'])
            self.dropvalidador+=1
        else:
            print("Ya se eliminaron los valores no validos.")


    def mostrarEstMag(self):
        print('Estaciones:', self.emisiones.ESTACION.unique())
        print('Contaminantes:', self.emisiones.MAGNITUD.unique())

    def mostrarDescriptivoContaminante(self):
        print(self.emisiones.groupby('MAGNITUD').VALOR.describe())

    def mostrarDescritivoContaminanteDistrito(self):
        #df_temp = self.emisiones.groupby(['ESTACION', 'MAGNITUD']).VALOR.describe()
        print(self.emisiones.groupby(['ESTACION', 'MAGNITUD']).VALOR.describe())

    def evolucion_mensual(self,cont, año):
        print(self.emisiones[(self.emisiones.MAGNITUD == cont) &
                             (self.emisiones.ANO == año)]
              .groupby(['ESTACION', 'MES']).VALOR.mean().unstack('MES'))

    def evolucion_diaria(self,estacion, mes):
        print(self.emisiones[(self.emisiones.ESTACION == estacion) &
                             (self.emisiones.MES == mes)].groupby(['MAGNITUD', 'DIA'])
                             .VALOR.mean().unstack('DIA'))

    #------Crear una función que reciba un rango de fechas y una magnitud y genere un gráfico
    def evolucionMagnitudRange(self,magnitud, desde, hasta):
        #self.emisiones['NOMBRE ESTACION'] = self.emisiones['ESTACION'].apply(lambda x: self.e
        df1 = self.emisiones[(self.emisiones['MAGNITUD'] == magnitud) &
                             (self.emisiones.FECHA >= desde) &
                             (self.emisiones.FECHA <= hasta)]
        df1.reset_index(inplace=True)
        df1.set_index('FECHA', inplace = True)
        fig, ax = plt.subplots(figsize=(22,6))
        df1.groupby('ESTACION')['VALOR'].plot(legend = True)
        plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
        plt.show();

    #-----
    def evolucionMagnitud(self,magnitud):
        if(isinstance(magnitud, str)):
            magnitud = int(magnitud)
        #cont = [ 1  6  7  8 12  9 10 14 20 30 35 42 43 44]
        #[i for i in cont if i ]
        df1 = self.emisiones[(self.emisiones['MAGNITUD'] == magnitud)]

        df1.set_index('FECHA', inplace = True)
```

```
        fig, ax = plt.subplots(figsize=(22,6))
        df1.groupby('ESTACION')['VALOR'].plot(legend = True)
        plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
        plt.show();
```

Crear Objeto

```
URL = ["/content/drive/MyDrive/CursoCapacitacion/proyectoIntegrador/emisiones-2016.csv",
       "/content/drive/MyDrive/CursoCapacitacion/proyectoIntegrador/emisiones-2017.csv",
       "/content/drive/MyDrive/CursoCapacitacion/proyectoIntegrador/emisiones-2018.csv",
       "/content/drive/MyDrive/CursoCapacitacion/proyectoIntegrador/emisiones-2019.csv"]
n_cores = os.cpu_count() - 1
miClase = miDataFrame(URL,n_cores)
```

Generar un DataFrame con los datos de los cuatro ficheros.

```
miClase.generarDF()
miClase.emisiones.head()
```

	PROVINCIA	MUNICIPIO	ESTACION	MAGNITUD	PUNTO_MUESTREO	ANO	MES	D01	V01	D(
0	28	79	4	1	28079004_1_38	2016	1	8.0	V	7
1	28	79	4	1	28079004_1_38	2016	2	12.0	V	13
2	28	79	4	1	28079004_1_38	2016	3	11.0	V	10
3	28	79	4	1	28079004_1_38	2016	4	8.0	V	9
4	28	79	4	1	28079004_1_38	2016	5	7.0	V	8

5 rows × 69 columns

Filtrar las columnas del DataFrame para quedarse con las columnas ESTACION, MAGNITUD, AÑO, MES y las correspondientes a los días D01, D02, etc.

```
miClase.filtrarDataFrame()
miClase.emisiones.head()
```

	ESTACION	MAGNITUD	ANO	MES	D01	D02	D03	D04	D05	D06	...	D22	D23	D24
0	4	1	2016	1	8.0	7.0	6.0	6.0	7.0	6.0	...	10.0	11.0	11.0
1	4	1	2016	2	12.0	13.0	9.0	9.0	11.0	9.0	...	11.0	10.0	9.0
2	4	1	2016	3	11.0	10.0	9.0	9.0	7.0	8.0	...	8.0	8.0	9.0
3	4	1	2016	4	8.0	9.0	9.0	8.0	8.0	9.0	...	8.0	8.0	8.0

Reestructurar el DataFrame para que los valores de los contaminantes de las columnas de los días aparezcan en una única columna.

```
1 miClase.reestructurarDF()
2 #miClase.emisiones
```

Añadir una columna con la fecha a partir de la concatenación del año, el mes y el día (usar el módulo datetime).

```
1 miClase.agregarFecha()
2 miClase.emisiones
```

	ESTACION	MAGNITUD	ANO	MES	DIA	VALOR	FECHA
0	4	1	2016	1	01	8.0	2016-01-01
1	4	1	2016	2	01	12.0	2016-02-01
2	4	1	2016	3	01	11.0	2016-03-01
3	4	1	2016	4	01	8.0	2016-04-01
4	4	1	2016	5	01	7.0	2016-05-01
...	...	...	...	...	...	...	...
225241	60	14	2019	8	31	98.0	2019-08-31
225242	60	14	2019	9	31	0.0	NaT
225243	60	14	2019	10	31	47.0	2019-10-31
225244	60	14	2019	11	31	0.0	NaT
225245	60	14	2019	12	31	4.0	2019-12-31

225246 rows × 7 columns

Eliminar las filas con fechas no válidas (utilizar la función isnat del módulo numpy) y ordenar el DataFrame por estaciones, contaminantes y fecha.

```
1 miClase.delNotValid()
2 miClase.emisiones
```

```
Estoy Eliminando valores no validos.
```

	ESTACION	MAGNITUD	ANO	MES	DIA	VALOR	FECHA
0	4	1	2016	1	01	8.0	2016-01-01
1	4	1	2016	2	01	12.0	2016-02-01
2	4	1	2016	3	01	11.0	2016-03-01
3	4	1	2016	4	01	8.0	2016-04-01
4	4	1	2016	5	01	7.0	2016-05-01
...	...	...	...	...	...	...	...
225238	60	14	2019	5	31	85.0	2019-05-31
225240	60	14	2019	7	31	92.0	2019-07-31
225241	60	14	2019	8	31	98.0	2019-08-31
225243	60	14	2019	10	31	47.0	2019-10-31
225245	60	14	2019	12	31	4.0	2019-12-31

221158 rows × 7 columns

Mostrar por pantalla las estaciones y los contaminantes disponibles en el DataFrame

```
1 miClase.mostrarEstMag()
```

```
Estaciones: [ 4  8 11 16 17 18 24 27 35 36 38 39 40 47 48 49 50 54 55 56 57 58 59 60]
Contaminantes: [ 1  6  7  8 12  9 10 14 20 30 35 42 43 44]
```

Mostrar un resumen descriptivo (mímino, máximo, media, etc) para cada contaminante.

```
1 miClase.mostrarDescriptivoContaminante()
```

MAGNITUD	count	mean	std	min	25%	50%	75%	max
1	14610.0	7.428953	7.012504	0.00	4.00	7.00	10.00	610.00
6	14610.0	0.350233	0.215935	0.00	0.20	0.30	0.40	14.90
7	35064.0	20.446412	135.123509	0.00	4.00	9.00	23.00	24742.00
8	35064.0	37.677618	20.118050	0.00	22.00	35.00	50.00	148.00
9	8948.0	10.087729	10.643591	0.00	6.00	9.00	13.00	850.00
10	17897.0	18.772923	35.723619	0.00	10.00	16.00	24.00	4481.00
12	35064.0	67.959417	61.443940	0.00	29.00	48.00	84.00	1005.00
14	20454.0	49.941772	24.753120	0.00	31.00	52.00	69.00	336.00
20	8766.0	2.364944	4.236706	0.00	0.80	1.60	2.80	195.00
30	8766.0	0.531371	0.538180	0.00	0.20	0.40	0.70	15.10
35	8766.0	0.479751	1.183618	0.00	0.10	0.20	0.50	35.70
42	4383.0	1.400897	0.251836	-0.01	1.25	1.38	1.54	3.09
43	4383.0	1.292923	0.230898	-0.14	1.17	1.28	1.43	2.77
44	4383.0	0.108941	0.068776	0.00	0.06	0.10	0.14	1.31

Mostrar un resumen descriptivo para cada contaminente por distritos.

```
1 miClase.mostrarDescritivoContaminanteDistrito()
```

ESTACION	MAGNITUD	count	mean	std	min	25%	50%	75%	max
4	1	1461.0	7.329911	16.379050	1.0	4.0	7.0	9.0	610.0
	6	1461.0	0.411499	0.172902	0.1	0.3	0.4	0.5	1.3
	7	1461.0	31.939767	37.667968	0.0	8.0	16.0	42.0	239.0
	8	1461.0	44.398357	17.766063	0.0	31.0	43.0	55.0	105.0
	12	1461.0	93.341547	72.436531	0.0	44.0	69.0	119.0	467.0
...	...	...	...	...	...	...	...	...	...
60	7	1461.0	12.326489	19.593109	1.0	2.0	4.0	12.0	151.0
	8	1461.0	31.125941	18.101896	3.0	18.0	27.0	41.0	101.0
	10	1461.0	17.033539	12.205022	1.0	9.0	14.0	21.0	215.0
	12	1461.0	50.023956	45.933843	6.0	22.0	33.0	60.0	328.0
	14	1461.0	60.718001	26.309952	4.0	42.0	65.0	81.0	119.0

[153 rows x 8 columns]

Crear una función que devuelva las emisiones medias mensuales de un contaminante y un año dados para todas las estaciones.

```
1 miClase.evolucion_mensual(1, 2016)
```

MES	1	2	3	4	5	6
ESTACION						
4	8.354839	8.551724	8.612903	8.066667	7.354839	7.266667
8	16.387097	17.827586	16.838710	15.400000	14.967742	17.333333
17	8.387097	8.551724	9.290323	16.800000	15.225806	15.333333
18	3.516129	3.793103	3.838710	3.266667	3.580645	4.033333
24	2.387097	2.172414	2.129032	2.066667	2.000000	2.100000
35	8.161290	10.517241	10.677419	12.433333	13.000000	16.466667
36	10.161290	7.310345	7.322581	6.133333	5.032258	5.933333
38	8.677419	4.068966	3.838710	2.766667	2.193548	2.133333
40	6.096774	5.793103	6.645161	5.633333	4.806452	5.300000
57	15.935484	16.137931	16.290323	4.766667	4.387097	4.533333

MES	7	8	9	10	11	12
ESTACION						
4	7.032258	7.000000	8.100000	7.387097	4.033333	8.129032
8	8.548387	5.806452	8.866667	11.064516	8.966667	10.032258
17	8.354839	7.870968	9.066667	19.064516	11.900000	14.548387
18	3.548387	9.129032	7.333333	6.741935	5.166667	6.806452
24	2.516129	2.580645	2.500000	2.774194	2.966667	3.709677
35	12.741935	11.612903	13.733333	10.709677	8.566667	4.032258
36	6.193548	7.096774	7.966667	7.322581	12.066667	4.548387
38	1.774194	1.709677	2.166667	3.032258	4.766667	7.516129
40	3.645161	3.419355	4.533333	6.419355	8.000000	11.161290
57	3.741935	3.129032	4.700000	7.451613	9.633333	15.709677

Crear una función que reciba un mes y una estación de medición y devuelva un diccionario con las medias de las magnitudes medidas por la estación durante ese mes

```
1 miClase.evolucion_diaria(4, 2)
```

DIA	01	02	03	04	05	06	07	08
MAGNITUD								
1	9.500	8.250	7.00	7.750	8.50	8.50	8.50	7.75
6	0.575	0.475	0.35	0.425	0.50	0.45	0.45	0.40
7	71.500	44.000	21.00	31.000	54.25	40.00	40.75	31.50
8	58.250	46.000	36.25	42.000	49.00	43.75	46.75	46.75
12	167.250	113.500	68.75	89.000	131.75	105.25	108.75	95.00

DIA	09	10	...	20	21	22	23	24
MAGNITUD			...					
1	8.25	8.00	...	10.000	9.50	11.75	11.000	9.250
6	0.45	0.35	...	0.525	0.55	0.65	0.625	0.525
7	40.25	20.75	...	54.250	57.75	83.75	79.250	57.000
8	53.00	43.25	...	57.500	57.75	67.50	68.750	55.500
12	114.75	75.25	...	140.750	146.25	196.25	190.000	142.500

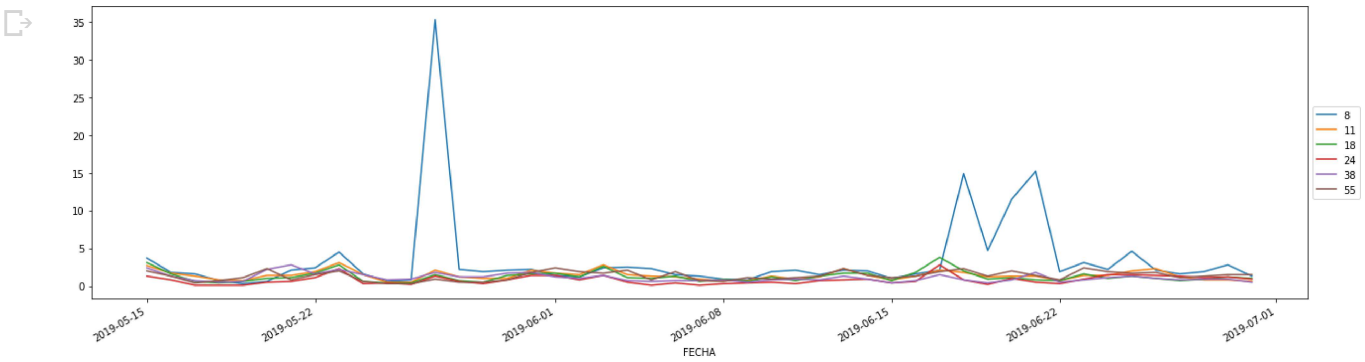
DIA	25	26	27	28	29
MAGNITUD					
1	9.50	9.000	8.250	8.50	9.0
6	0.55	0.525	0.425	0.45	0.4
7	49.25	52.250	29.500	33.00	18.0

```
8          59.00   54.500  42.500  47.00  42.0
12        134.50  134.250  88.250  97.25  70.0

[5 rows x 29 columns]
```

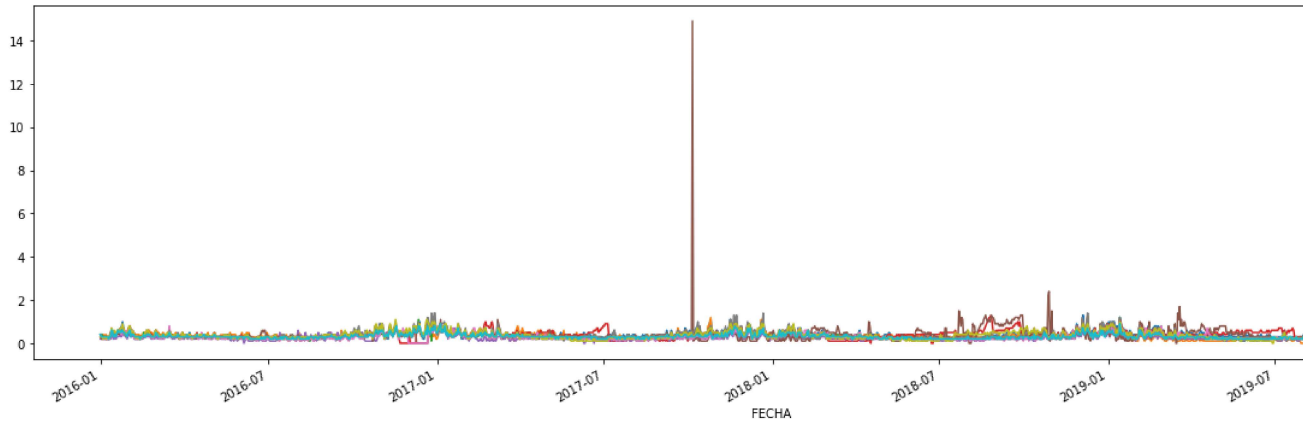
Crear una función que reciba un rango de fechas y una magnitud y genere un gráfico con la evolución diaria de la magnitud para cada estación de medición en las fechas indicadas.

```
#magnitudes : 1  6  7  8 12  9 10 14 20 30 35 42 43 44
miClase.evolucionMagnitudRange(20, '2019-05-15', '2019-06-30')
```



Crear una función que reciba una magnitud y genere un gráfico con las medias mensuales dentro de Madrid Central y fuera de ella.

```
miClase.evolucionMagnitud(6)
```




```
1
```

```
1
```

✓ 0 s se ejecutó 09:27