03.Lectura_archivos_especiales

October 29, 2022

1 Lectura de datos

Abriendo el archivo de la forma mas simple posible

[ ]:
```
import pandas as pd

data_1 = pd.read_csv('../../data/3116406.txt')
data_1.head()
```

[ ]:
```
  STATION           STATION_NAME                                      DATE
PRCP     TAVG     TMAX     TMIN
0  ---------------- ---------------------------…
1  GHCND:PEM00084691                          …
2  GHCND:PEM00084691                          …
3  GHCND:PEM00084691                          …
4  GHCND:PEM00084691                          …
```

[ ]:
```
f = open('../../data/3116406.txt', 'r')
f
```

[ ]:
```
<_io.TextIOWrapper name='../../data/3116406.txt' mode='r' encoding='UTF-8'>
```

Antes de continuar vamos a quitarle el encabezado y guardarlo para despues

[ ]:
```
encabezado = next(f).split()

next(f)  ## comando next????

print(encabezado)
```

```
['STATION', 'STATION_NAME', 'DATE', 'PRCP', 'TAVG', 'TMAX', 'TMIN']
```

[ ]:
```
f
```

[ ]:
```
<_io.TextIOWrapper name='../../data/3116406.txt' mode='r' encoding='UTF-8'>
```

Organizando los datos en una lista

[ ]:
```
#definimos la lista de destino
dato = []

# lectura del archivo linea por linea
for line in f:
    dato.append(line.split())

for i in dato[:10]:
    print(i)
```

```
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630301', '-9999',
'23.3', '27.2', '-9999']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630302', '-9999',
'22.5', '27.8', '17.2']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630303', '-9999',
'21.3', '26.1', '16.1']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630304', '-9999',
'21.1', '26.1', '17.2']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630305', '-9999',
'22.7', '27.2', '17.2']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630306', '-9999',
'21.7', '27.2', '17.2']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630307', '-9999',
'22.4', '26.1', '17.8']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630308', '-9999',
'21.3', '26.1', '16.1']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630309', '-9999',
'21.5', '27.2', '17.8']
['GHCND:PEM00084691', 'PISCO', 'INTERNATIONAL', 'PE', '19630310', '-9999',
'23.4', '-9999', '17.2']
```

Pasando de una lista a un `DataFrame`

[ ]:
```
import pandas as pd
```

[ ]:
```
df = pd.DataFrame(dato, columns=encabezado)

#que paso?
```

```
␣
↪---------------------------------------------------------------------------

AssertionError                            Traceback (most recent call␣
↪last)
```

```
       File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/core/
↪internals/construction.py:969, in _finalize_columns_and_data(content, columns,␣
↪dtype)
       968 try:
   --> 969     columns = _validate_or_indexify_columns(contents, columns)
       970 except AssertionError as err:
       971     # GH#26429 do not raise user-facing AssertionError


       File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/core/
↪internals/construction.py:1017, in _validate_or_indexify_columns(content,␣
↪columns)
      1015 if not is_mi_list and len(columns) != len(content):  # pragma: no␣
↪cover
      1016     # caller's responsibility to check for this…
   -> 1017     raise AssertionError(
      1018         f"{len(columns)} columns passed, passed data had "
      1019         f"{len(content)} columns"
      1020     )
      1021 elif is_mi_list:
      1022
      1023     # check if nested list column, length of each sub-list should be␣
↪equal


       AssertionError: 7 columns passed, passed data had 9 columns


   The above exception was the direct cause of the following exception:


       ValueError                                Traceback (most recent call␣
↪last)

       Cell In [7], line 1
   ----> 1 df = pd.DataFrame(dato, columns=encabezado)


       File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/core/
↪frame.py:744, in DataFrame.__init__(self, data, index, columns, dtype, copy)
       742     if columns is not None:
       743         columns = ensure_index(columns)
   --> 744     arrays, columns, index = nested_data_to_arrays(
       745         # error: Argument 3 to "nested_data_to_arrays" has␣
↪incompatible
       746         # type "Optional[Collection[Any]]"; expected␣
↪"Optional[Index]"
```

```
       747         data,
       748         columns,
       749         index,  # type: ignore[arg-type]
       750         dtype,
       751     )
       752     mgr = arrays_to_mgr(
       753         arrays,
       754         columns,
      (…)
       757         typ=manager,
       758     )
       759 else:


       File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/core/
↪internals/construction.py:510, in nested_data_to_arrays(data, columns, index,␣
↪dtype)
       507 if is_named_tuple(data[0]) and columns is None:
       508     columns = ensure_index(data[0]._fields)
   --> 510 arrays, columns = to_arrays(data, columns, dtype=dtype)
       511 columns = ensure_index(columns)
       513 if index is None:


       File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/core/
↪internals/construction.py:875, in to_arrays(data, columns, dtype)
       872     data = [tuple(x) for x in data]
       873     arr = _list_to_arrays(data)
   --> 875 content, columns = _finalize_columns_and_data(arr, columns, dtype)
       876 return content, columns


       File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/core/
↪internals/construction.py:972, in _finalize_columns_and_data(content, columns,␣
↪dtype)
       969     columns = _validate_or_indexify_columns(contents, columns)
       970 except AssertionError as err:
       971     # GH#26429 do not raise user-facing AssertionError
   --> 972     raise ValueError(err) from err
       974 if len(contents) and contents[0].dtype == np.object_:
       975     contents = _convert_object_array(contents, dtype=dtype)


       ValueError: 7 columns passed, passed data had 9 columns
```

Nuestro problema es el número de colmnas, entonces tenemos que verificar la cantidad de elementos en cada lista que esta dentro de nuestra variable `dato`.

[ ]:
```
dato[0]
```

[ ]:
```
['GHCND:PEM00084691',
 'PISCO',
 'INTERNATIONAL',
 'PE',
 '19630301',
 '-9999',
 '23.3',
 '27.2',
 '-9999']
```

Veamos que es lo que tenemos en nuestro encabezado

[ ]:
```
encabezado
```

[ ]:
```
['STATION', 'STATION_NAME', 'DATE', 'PRCP', 'TAVG', 'TMAX', 'TMIN']
```

[ ]:
```
encabezado.insert(1, 'Station name')
encabezado.insert(1, 'Station name')
```

[ ]:
```
# verificando lo que tenemos ahora
print(encabezado)
```

```
['STATION', 'Station name', 'Station name', 'STATION_NAME', 'DATE', 'PRCP',
'TAVG', 'TMAX', 'TMIN']
```

Ahora que ya tenemos la cantidad de columnas completas, le decimos cual es la forma correcta de dejarlo como un `DataFrame`

[ ]:
```
df = pd.DataFrame(dato, columns=encabezado)
```

Ahora veamos que pasa con nuestro `DataFrame`

[ ]:
```
print(df.head())
# hay algun problema?
```

```
             STATION Station name   Station name STATION_NAME      DATE  \
0  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630301
1  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630302
2  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630303
3  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630304
4  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630305

    PRCP  TAVG  TMAX   TMIN
0  -9999  23.3  27.2  -9999
1  -9999  22.5  27.8   17.2
2  -9999  21.3  26.1   16.1
```

```
3  -9999  21.1  26.1   17.2
4  -9999  22.7  27.2   17.2
```

[ ]:
```
import numpy as np
df = df.replace('-9999', np.nan)

df
```

[ ]:
```
                 STATION Station name   Station name STATION_NAME      DATE  \
0      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630301
1      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630302
2      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630303
3      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630304
4      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630305
…                    …           …            …           …        …
16682  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211027
16683  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211028
16684  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211029
16685  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211030
16686  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211031

       PRCP  TAVG  TMAX  TMIN
0       NaN  23.3  27.2   NaN
1       NaN  22.5  27.8  17.2
2       NaN  21.3  26.1  16.1
3       NaN  21.1  26.1  17.2
4       NaN  22.7  27.2  17.2
…     …   …   …   …
16682   NaN  17.1   NaN   NaN
16683   NaN  16.7   NaN  12.9
16684   NaN  17.3   NaN  13.8
16685   NaN  16.9   NaN   NaN
16686   NaN  18.0   NaN  16.9

[16687 rows x 9 columns]
```

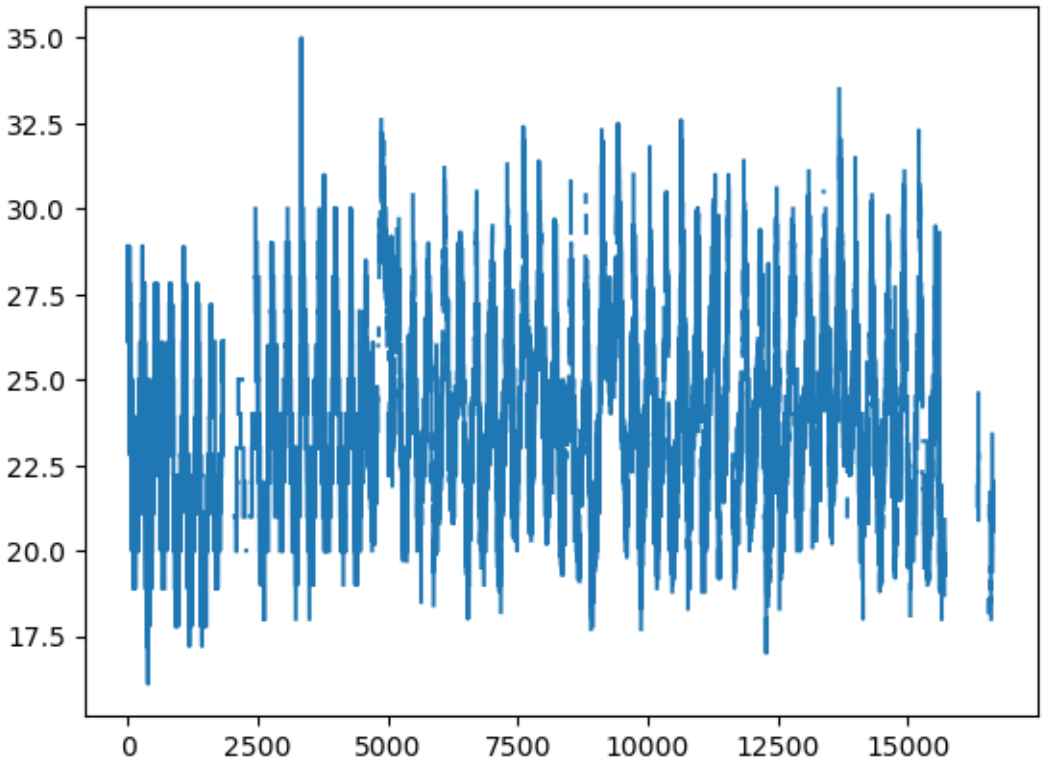
[ ]:
```
type(df.TMAX[0])
df.TMAX.plot()
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

Cell In [15], line 2
  1 type(df.TMAX[0])
```

```
----> 2 df.TMAX.plot()


File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/
↪plotting/_core.py:1000, in PlotAccessor.__call__(self, *args, **kwargs)
    997             label_name = label_kw or data.columns
    998             data.columns = label_name
-> 1000 return plot_backend.plot(data, kind=kind, **kwargs)


File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/
↪plotting/_matplotlib/__init__.py:71, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result


File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/
↪plotting/_matplotlib/core.py:450, in MPLPlot.generate(self)
    448 def generate(self) -> None:
    449     self._args_adjust()
--> 450     self._compute_plot_data()
    451     self._setup_subplots()
    452     self._make_plot()


File ~/miniconda3/envs/geocat/lib/python3.9/site-packages/pandas/
↪plotting/_matplotlib/core.py:635, in MPLPlot._compute_plot_data(self)
    633 # no non-numeric frames or series allowed
    634 if is_empty:
--> 635     raise TypeError("no numeric data to plot")
    637 self.data = numeric_data.apply(self._convert_to_ndarray)


TypeError: no numeric data to plot
```

[ ]:
```
df.TMAX.astype('float').plot()
```

[ ]:
```
<AxesSubplot: >
```



Recuerda cerrar el archivo siempre, porque sino permanece en la memoria.

[ ]:
```
f.close()
```

Ya que vimos como se hace entonces ahora hagamoslo de forma mas simple

[ ]:
```
f = open('../../data/3116406.txt', 'r')

encabezado = next(f).split()
next(f)  ## comando next????
dato = []
for line in f:
    d_line = line.split()
    if '-9999' in d_line:
        # reemplazar los -999 con la funciona lambda
        d_line = list(map(lambda x: x.replace('-9999', 'NaN'), d_line))
    dato.append(d_line)

f.close() # no se olviden de cerrar el archivo

encabezado.insert(1, 'Station name')
encabezado.insert(1, 'Station name')
```

```
df = pd.DataFrame(dato, columns=encabezado)
```

[ ]:
```
df
```

[ ]:
```
                 STATION Station name   Station name STATION_NAME      DATE  \
0      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630301
1      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630302
2      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630303
3      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630304
4      GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  19630305
…                    …           …              …           …        …
16682  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211027
16683  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211028
16684  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211029
16685  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211030
16686  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE  20211031

       PRCP  TAVG  TMAX  TMIN
0       NaN  23.3  27.2   NaN
1       NaN  22.5  27.8  17.2
2       NaN  21.3  26.1  16.1
3       NaN  21.1  26.1  17.2
4       NaN  22.7  27.2  17.2
…     …   …   …   …
16682   NaN  17.1   NaN   NaN
16683   NaN  16.7   NaN  12.9
16684   NaN  17.3   NaN  13.8
16685   NaN  16.9   NaN   NaN
16686   NaN  18.0   NaN  16.9

[16687 rows x 9 columns]
```

[ ]:
```
df.index = df.DATE.apply(pd.to_datetime)
tmax = df.TMAX.astype('float')
```

[ ]:
```
df.head()
```

[ ]:
```
                     STATION Station name   Station name STATION_NAME  \
DATE
1963-03-01  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE
1963-03-02  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE
1963-03-03  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE
1963-03-04  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE
1963-03-05  GHCND:PEM00084691        PISCO  INTERNATIONAL           PE

                 DATE PRCP  TAVG  TMAX  TMIN
DATE
```

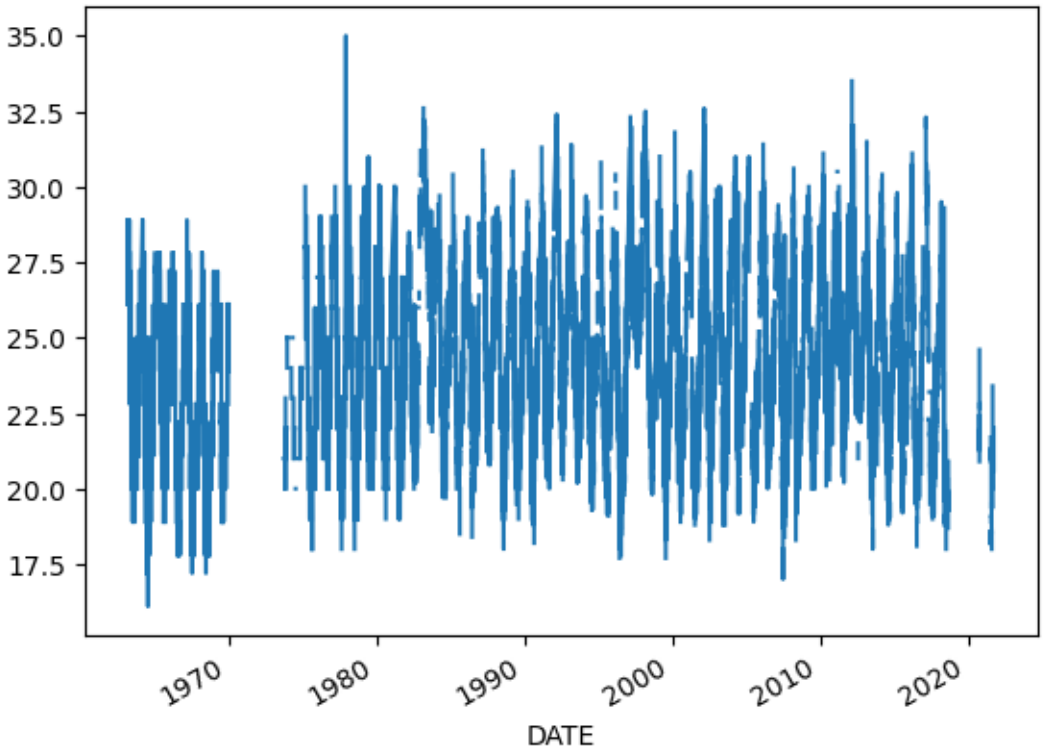
```
1963-03-01  19630301  NaN  23.3  27.2   NaN
1963-03-02  19630302  NaN  22.5  27.8  17.2
1963-03-03  19630303  NaN  21.3  26.1  16.1
1963-03-04  19630304  NaN  21.1  26.1  17.2
1963-03-05  19630305  NaN  22.7  27.2  17.2
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 16687 entries, 1963-03-01 to 2021-10-31
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   STATION       16687 non-null  object
 1   Station name  16687 non-null  object
 2   Station name  16687 non-null  object
 3   STATION_NAME  16687 non-null  object
 4   DATE          16687 non-null  object
 5   PRCP          16687 non-null  object
 6   TAVG          16687 non-null  object
 7   TMAX          16687 non-null  object
 8   TMIN          16687 non-null  object
dtypes: object(9)
memory usage: 1.3+ MB
```

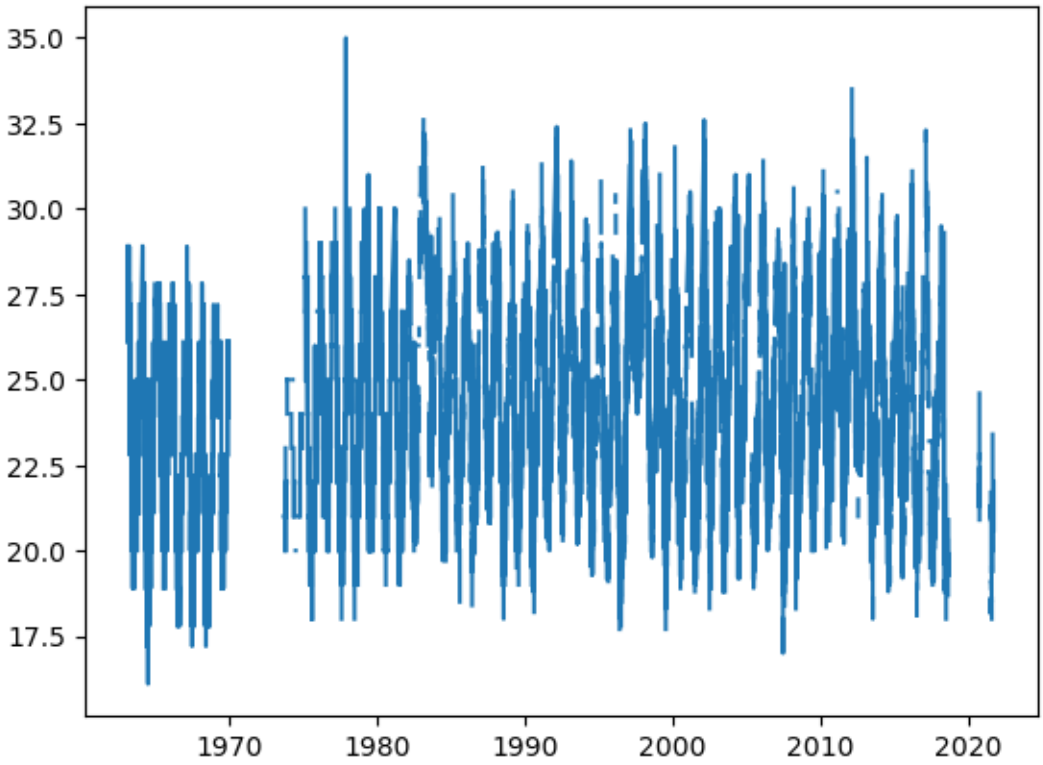
[ ]:
```
tmax.plot()
```

[ ]:
```
<AxesSubplot: xlabel='DATE'>
```



[ ]:
```
from matplotlib import pyplot as plt

plt.plot(tmax)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f484cfc1fa0>]
```



1.1 Pandas llegando a la ayuda!

[ ]:
```
df_0 = pd.read_fwf('../../data/3116406.txt')
df_0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16688 entries, 0 to 16687
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   STATION       16688 non-null  object
 1   STATION_NAME  16688 non-null  object
 2   DATE          16688 non-null  object
 3   PRCP          16688 non-null  object
 4   TAVG          16688 non-null  object
 5   TMAX          16688 non-null  object
 6   TMIN          16688 non-null  object
dtypes: object(7)
memory usage: 912.8+ KB
```

[ ]:
```
df  = pd.read_fwf('../../data/3116406.txt', skiprows=2,
                  header=None, na_values='-9999')
```

```
df.head()
```

[ ]:

	0	1	2	3	4	5	6	7	8
0	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630301	NaN	23.3	27.2	NaN
1	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630302	NaN	22.5	27.8	17.2
2	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630303	NaN	21.3	26.1	16.1
3	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630304	NaN	21.1	26.1	17.2
4	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630305	NaN	22.7	27.2	17.2

Debido a que le pedimos al **pandas** a abrir olvidando el encabezado tenemos que reasignar los nombres de columnas.

[ ]:
```
df.columns =['STATION', 'CITY', 'INTER', 'COUNTRY', 'DATE', 'PRCP',
             'TAVG', 'TMAX', 'TMIN']
df.index = pd.to_datetime(df['DATE'], format="%Y%m%d")
df.head()
```

[ ]:

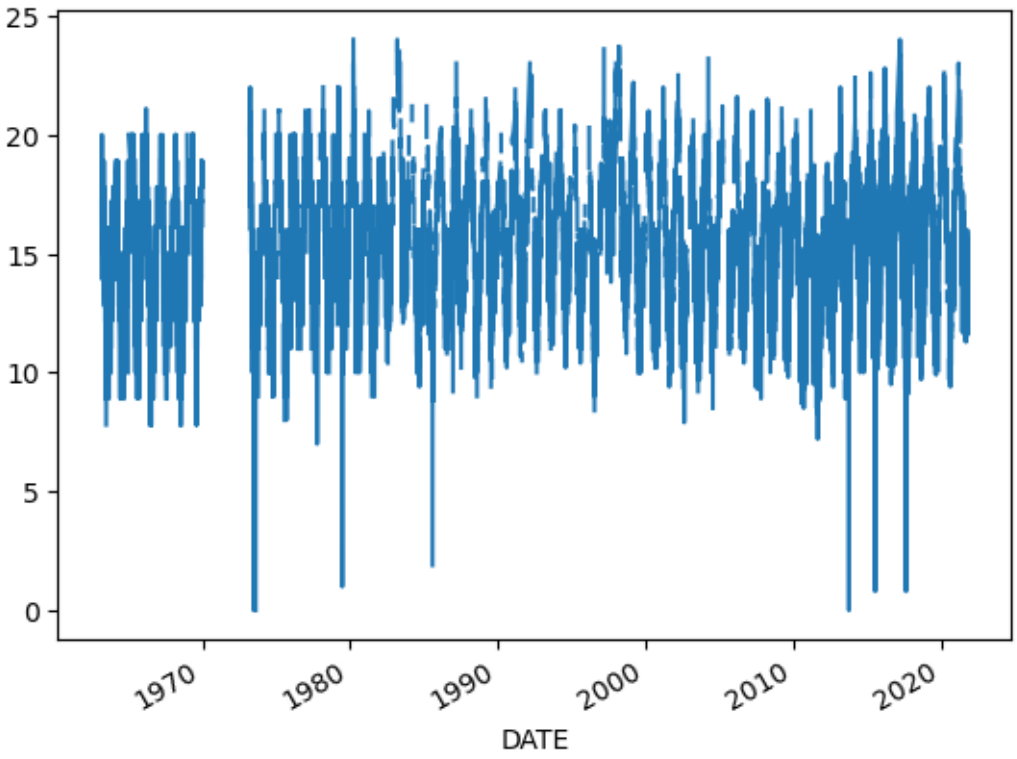
DATE	STATION	CITY	INTER	COUNTRY	DATE	PRCP	TAVG	TMAX	TMIN
1963-03-01	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630301	NaN	23.3	27.2	NaN
1963-03-02	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630302	NaN	22.5	27.8	17.2
1963-03-03	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630303	NaN	21.3	26.1	16.1
1963-03-04	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630304	NaN	21.1	26.1	17.2
1963-03-05	GHCND:PEM00084691	PISCO	INTERNATIONAL	PE	19630305	NaN	22.7	27.2	17.2

[ ]:
```
df.TMIN[1]
```

[ ]: 17.2

[ ]:
```
df.TMIN.plot()
```

[ ]: <AxesSubplot: xlabel='DATE'>



[ ]:

1.2 Retornar al indice