USM Numérica

Librarían Pandas

Objetivos

1. Conocer los principales comandos de la librería pandas
2. Utilizar pandas para limpieza y manipulación de datos.

Sobre el autor

Sebastián Flores

ICM UTFSM

sebastian.flores@usm.cl

Sobre la presentación

Contenido creada en ipython notebook (jupyter)

Versión en Slides gracias a RISE de Damián Avila

Software:

- python 2.7 o python 3.1
- pandas 0.16.1

Opcional:

- numpy 1.9.2
- matplotlib 1.3.1

Aprender haciendo

Consideraremos el siguiente archivo `data.csv` que contiene datos incompletos:

In [132]:
```
%%bash
cat data/data.csv
```

```
diametro;altura;volumen;tipo_de_arbol
11.2;75;19.9;Cherrie Tree
11.3;79;24.2;Cherry Tree
11.4;76;21.0;Cherry Tree
11.4;76;21.4;Apple Tree
13.7;71;25.7;Cherry Tree
13.8;64;24.9;Cherry Tree
14.0;78;34.5;Cherrie Tree
14.2;80;31.7;Cherry Tree
;74;36.3;Apple Tree
16.0;72;38.3;Cherry Tree
16.3;77;42.6;Cherry Tree
17.3;81;55.4;Apple Tree
17.5;;55.7;Cherry Tree
17.9;80;58.3;Cherry Tree
18.0;80;51.5;Cherry Tree
18.0;;51.0;
20.6;;;Cherry Tree
```

1.- ¿Porqué utilizar pandas?

Razón oficial: Porque en numpy no es posible mezclar tipos de datos, lo cual complica cargar, usar, limpiar y guardar datos mixtos.

Razón personal: Porque habían cosas que en R eran más fáciles pero no pythonísticas. La librería pandas es un excelente compromiso.

In [133]:

```
import numpy as np
df = np.loadtxt("data/data.csv", delimiter=";", dtype=str)
print( df )
```

```
[['diametro' 'altura' 'volumen' 'tipo_de_arbol']
 ['11.2' '75' '19.9' 'Cherrie Tree']
 ['11.3' '79' '24.2' 'Cherry Tree']
 ['11.4' '76' '21.0' 'Cherry Tree']
 ['11.4' '76' '21.4' 'Apple Tree']
 ['13.7' '71' '25.7' 'Cherry Tree']
 ['13.8' '64' '24.9' 'Cherry Tree']
 ['14.0' '78' '34.5' 'Cherrie Tree']
 ['14.2' '80' '31.7' 'Cherry Tree']
 ['' '74' '36.3' 'Apple Tree']
 ['16.0' '72' '38.3' 'Cherry Tree']
 ['16.3' '77' '42.6' 'Cherry Tree']
 ['17.3' '81' '55.4' 'Apple Tree']
 ['17.5' '' '55.7' 'Cherry Tree']
 ['17.9' '80' '58.3' 'Cherry Tree']
 ['18.0' '80' '51.5' 'Cherry Tree']
 ['18.0' '' '51.0' '']
 ['20.6' '' '' 'Cherry Tree']]
```

In [134]:

```
import pandas as pd
df = pd.read_csv("data/data.csv", sep=";")
print( df )
#df
```

```
    diametro  altura  volumen tipo_de_arbol
0       11.2    75.0     19.9  Cherrie Tree
1       11.3    79.0     24.2   Cherry Tree
2       11.4    76.0     21.0   Cherry Tree
3       11.4    76.0     21.4    Apple Tree
4       13.7    71.0     25.7   Cherry Tree
5       13.8    64.0     24.9   Cherry Tree
6       14.0    78.0     34.5  Cherrie Tree
7       14.2    80.0     31.7   Cherry Tree
8        NaN    74.0     36.3    Apple Tree
9       16.0    72.0     38.3   Cherry Tree
10      16.3    77.0     42.6   Cherry Tree
11      17.3    81.0     55.4    Apple Tree
12      17.5     NaN     55.7   Cherry Tree
13      17.9    80.0     58.3   Cherry Tree
14      18.0    80.0     51.5   Cherry Tree
15      18.0     NaN     51.0           NaN
16      20.6     NaN      NaN   Cherry Tree
```

In [135]:

```
inch2m = 0.0254
feet2m = 0.3048
df.diametro = df.diametro * inch2m
df.altura = df.altura * feet2m
df.volumen = df.volumen * (feet2m**3)
df.tipo_de_arbol = "Cherry Tree"
df
```

Out[135]:

	diametro	altura	volumen	tipo_de_arbol
0	0.28448	22.8600	0.563505	Cherry Tree
1	0.28702	24.0792	0.685268	Cherry Tree
2	0.28956	23.1648	0.594654	Cherry Tree
3	0.28956	23.1648	0.605981	Cherry Tree
4	0.34798	21.6408	0.727743	Cherry Tree
5	0.35052	19.5072	0.705089	Cherry Tree
6	0.35560	23.7744	0.976931	Cherry Tree
7	0.36068	24.3840	0.897644	Cherry Tree
8	NaN	22.5552	1.027902	Cherry Tree
9	0.40640	21.9456	1.084535	Cherry Tree
10	0.41402	23.4696	1.206298	Cherry Tree
11	0.43942	24.6888	1.568753	Cherry Tree
12	0.44450	NaN	1.577248	Cherry Tree
13	0.45466	24.3840	1.650872	Cherry Tree
14	0.45720	24.3840	1.458318	Cherry Tree
15	0.45720	NaN	1.444159	Cherry Tree
16	0.52324	NaN	NaN	Cherry Tree

In [136]:
```
print( df.columns )
```

```
Index(['diametro', 'altura', 'volumen', 'tipo_de_arbol'], dtype='object')
```

In [137]:
```
print( df.index )
```

```
RangeIndex(start=0, stop=17, step=1)
```

In [138]:
```
print( df["altura"]*2 )
```

```
0     45.7200
1     48.1584
2     46.3296
3     46.3296
4     43.2816
5     39.0144
6     47.5488
7     48.7680
8     45.1104
9     43.8912
10    46.9392
11    49.3776
12        NaN
13    48.7680
14    48.7680
15        NaN
16        NaN
Name: altura, dtype: float64
```

In [139]:
```
print( df["diametro"]**2 * df["altura"] / df.volumen )
```

```
0     3.283082
1     2.894717
2     3.266190
3     3.205140
4     3.600840
5     3.399197
6     3.077295
7     3.533824
8          NaN
9     3.342037
10    3.334985
11    3.038820
12         NaN
13    3.053269
14    3.495146
15         NaN
16         NaN
dtype: float64
```

2. Lo básico de pandas

- Pandas imita los dataframes de R, pero en python. Todo lo que no tiene sentido es porque se parece demasiado a R.
- Pandas permite tener datos como en tablas de excel: datos en una columna pueden ser mixtos.
- La idea central es que la indexación es "a medida": las columnas y las filas (index) pueden ser enteros o floats, pero también pueden ser strings. Depende de lo que tenga sentido.
- Los elementos básicos de pandas son:
    - **Series**: Conjunto de valores con indexación variable.
    - **DataFrames**: Conjunto de Series.

2.1 Series

Una serie es un conveniente conjunto de datos, como una columna de datos de excel, pero con indexación más genérica.

```
pd.Series(self, data=None, index=None, dtype=None, name=None, copy=False,
fastpath=False)
```

In [140]:
```
import pandas as pd
s1 = pd.Series([False, 1, 2., "3", 4 + 0j])
print( s1 )
```

```
0     False
1         1
2       2.0
3         3
4    (4+0j)
dtype: object
```

In [141]:
```
# Casting a otros tipos
print( list(s1) )
print( set(s1) )
print( np.array(s1) )
```

```
[False, 1, 2.0, '3', (4+0j)]
{False, 1, 2.0, (4+0j), '3'}
[False 1 2.0 '3' (4+0j)]
```

In [142]:
```
# Ejemplo de operatoria
s0 = pd.Series(range(6), index=range(6))
s1 = pd.Series([1,2,3], index=[1,2,3])
s2 = pd.Series([4,5,6], index=[4,5,6])
s3 = pd.Series([10,10,10], index=[1,4,6])
```

In [143]:
```
print( s0 )
```

```
0    0
1    1
2    2
3    3
4    4
5    5
dtype: int64
```

In [144]:
```
print( s0 + s1 )
```

```
0    NaN
1    2.0
2    4.0
3    6.0
4    NaN
5    NaN
dtype: float64
```

In [145]:
```
print( s0 + s1 + s2 )
```

```
0   NaN
1   NaN
```

```
2   NaN
3   NaN
4   NaN
5   NaN
6   NaN
dtype: float64
```

In [146]:
```
print( s0.add(s1, fill_value=0) )
```

```
0    0.0
1    2.0
2    4.0
3    6.0
4    4.0
5    5.0
dtype: float64
```

2.2 DataFrames

Un Dataframe es una colección de Series con una indexación común. Como una planilla de excel.

```
pd.DataFrame(self, data=None, index=None,
             columns=None, dtype=None, copy=False)
```

In [147]:

```
# dict
df = pd.DataFrame({"col1":[1,2,3,4],
                   "col2":[1., 2., 3., 4.],
                   "col3":["uno", "dos", "tres", "cuatro"]})
df
```

Out[147]:

	col1	col2	col3
0	1	1.0	uno
1	2	2.0	dos
2	3	3.0	tres
3	4	4.0	cuatro

3.1 Obteniendo datos

1. **Archivo csv**
2. **Archivo json**
3. **Archivo de excel**: convertir a csv cuidadosamente (elegir un separador apropiado, no comentar strings).

In [148]:

```
# csv
df = pd.read_csv("data/data.csv", sep=";")
df
```

Out[148]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree

In [149]:

```
df = pd.read_json("data/data.json")
df
```

Out[149]:

	diametro	altura	volumen	tipo_de_arbol
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
7	14.2	80.0	31.7	Cherry Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
16	20.6	NaN	NaN	Cherry Tree

4.- Inspeccionando datos

1. Accesando las columnas
2. shape
3. head, tail, describe
4. histogram
5. pd.scatter_matrix
6. count_values

In [150]:
```
df = pd.read_csv("data/data.csv", sep=";")
df.columns
```

Out[150]:
```
Index(['diametro', 'altura', 'volumen', 'tipo_de_arbol'], dtype='object')
```

In [151]:
```
df['altura']
```

Out[151]:
```
0     75.0
1     79.0
2     76.0
3     76.0
4     71.0
5     64.0
6     78.0
7     80.0
8     74.0
9     72.0
10    77.0
11    81.0
12     NaN
13    80.0
14    80.0
15     NaN
16     NaN
Name: altura, dtype: float64
```

In [152]:
```
df.shape
```

Out[152]:
```
(17, 4)
```

In [153]:
```
df.head()
```

Out[153]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree

In [154]:
```
df.tail()
```

Out[154]:

	diametro	altura	volumen	tipo_de_arbol
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree

In [155]:
```
df.describe()
```

Out[155]:

	diametro	altura	volumen
count	16.000000	14.000000	16.000000
mean	15.162500	75.928571	37.025000
std	2.948418	4.615430	13.773816
min	11.200000	64.000000	19.900000
25%	13.125000	74.250000	24.725000
50%	15.100000	76.500000	35.400000
75%	17.600000	79.750000	51.125000
max	20.600000	81.000000	58.300000

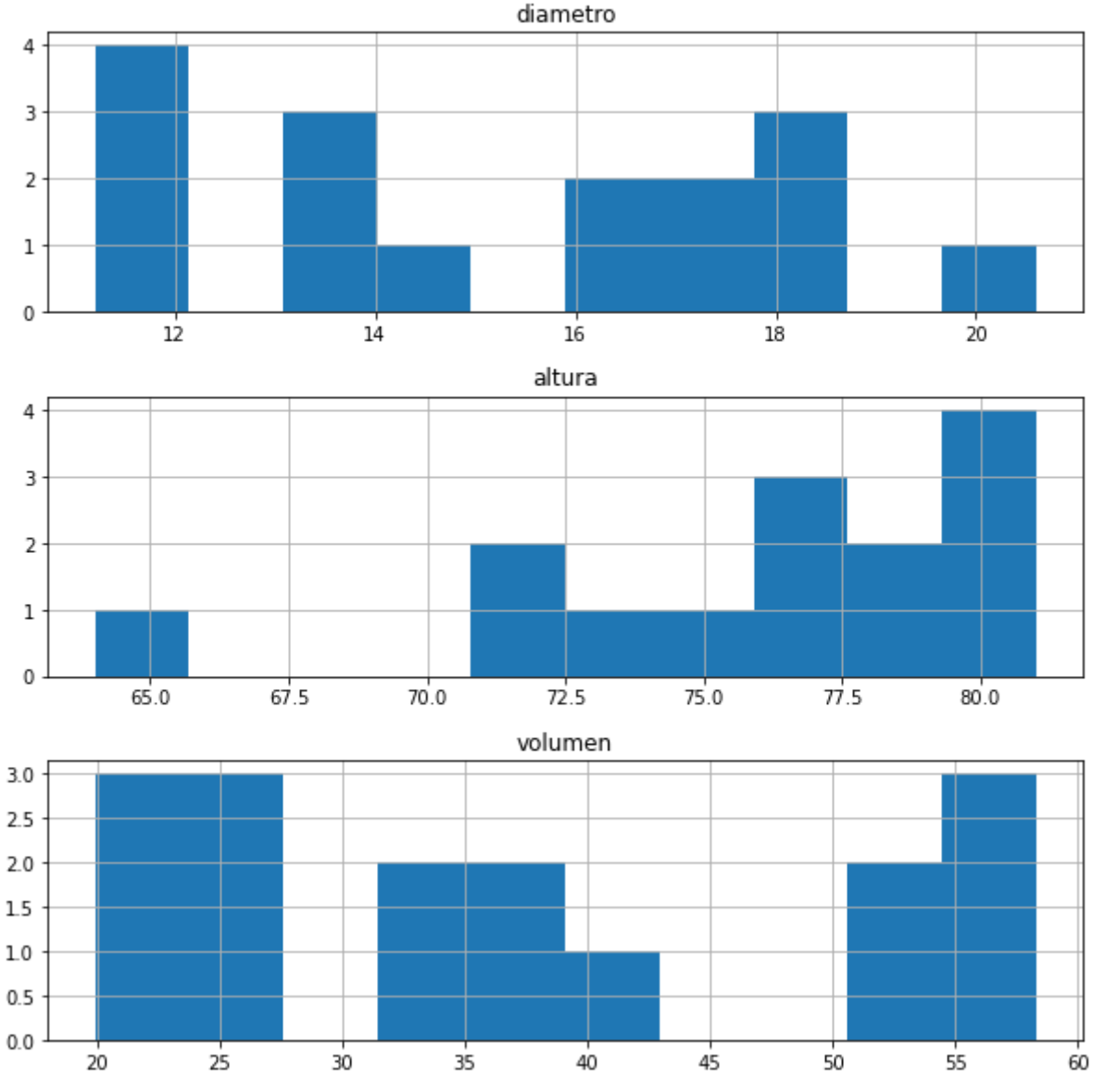
In [156]:
```
df.describe(include="all")
```

Out[156]:

	diametro	altura	volumen	tipo_de_arbol
count	16.000000	14.000000	16.000000	16
unique	NaN	NaN	NaN	3
top	NaN	NaN	NaN	Cherry Tree
freq	NaN	NaN	NaN	11
mean	15.162500	75.928571	37.025000	NaN
std	2.948418	4.615430	13.773816	NaN
min	11.200000	64.000000	19.900000	NaN
25%	13.125000	74.250000	24.725000	NaN
50%	15.100000	76.500000	35.400000	NaN
75%	17.600000	79.750000	51.125000	NaN
max	20.600000	81.000000	58.300000	NaN

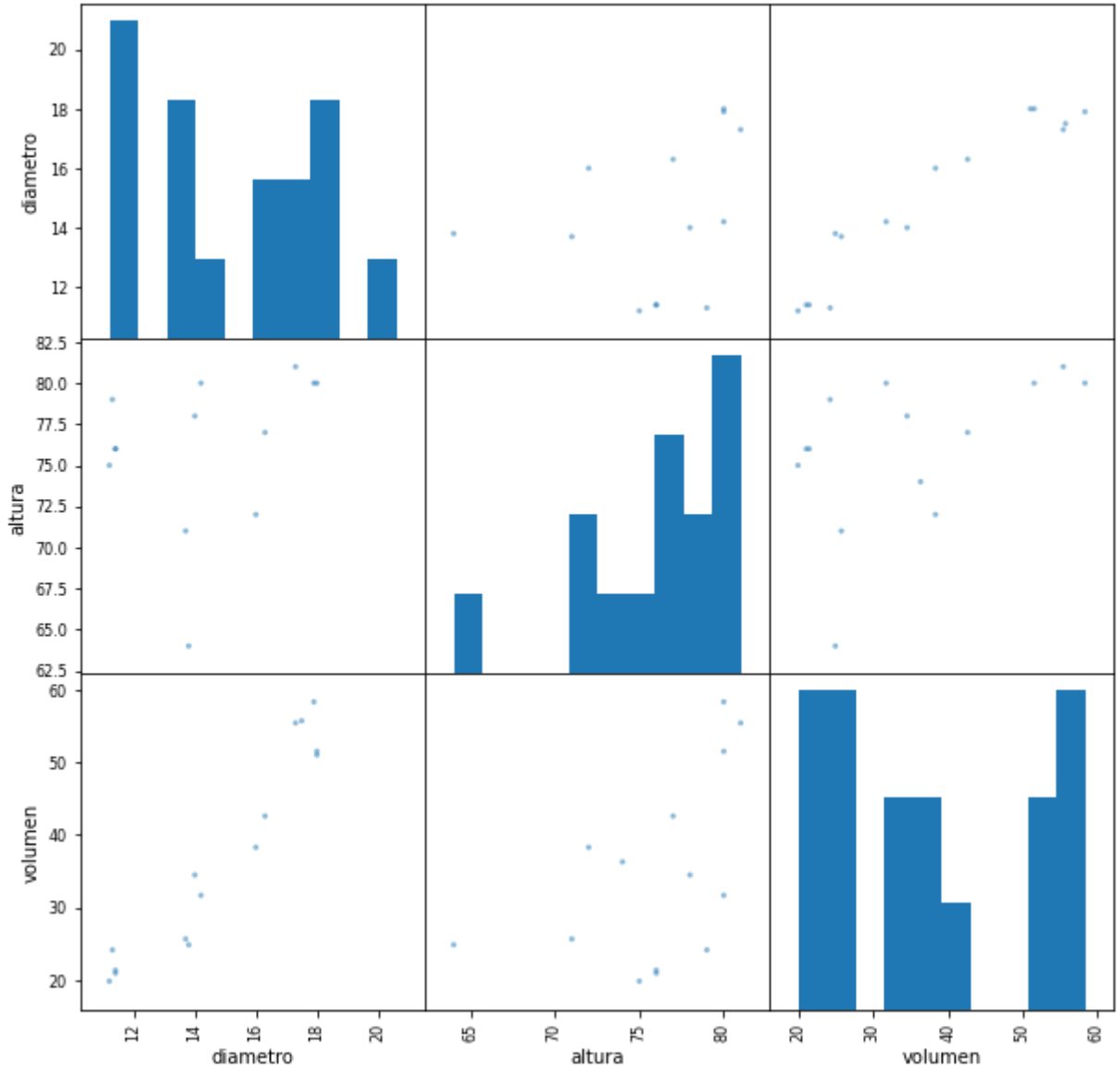
In [157]:

```
from matplotlib import pyplot as plt
df.hist(figsize=(10,10), layout=(3,1))
#df.hist(figsize=(8,8), layout=(3,1), by="tipo_de_arbol")
plt.show()
```


diametro
0
1
2
3
4
12
14
16
18
20
altura
0
1
2
3
4
65.0
67.5
70.0
72.5
75.0
77.5
80.0
volumen
0.0
0.5
1.0
1.5
2.0
2.5
3.0
20
25
30
35
40
45
50
55
60

In [158]:

```
from matplotlib import pyplot as plt
pd.plotting.scatter_matrix(df, figsize=(10,10), range_padding=0.2)
plt.show()
```


diametro
altura
volumen
20
18
16
14
12
82.5
80.0
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60
50
40
30
20
12
14
16
18
20
65
70
75
80
20
30
40
50
60

In [159]:
```
df.tipo_de_arbol.value_counts()
```

Out[159]:
```
Cherry Tree     11
Apple Tree       3
Cherrie Tree     2
Name: tipo_de_arbol, dtype: int64
```

5.- Manipulando DataFrames

1. Agregando columnas
2. Borrando columnas
3. Agregando filas
4. Borrando filas
5. Mask
6. Grouping
7. Imputación de datos
8. Apply
9. Merge (a la SQL)
10. Accesamiento

5.1 Agregando columnas

In [160]:
```
df = pd.read_csv("data/data.csv", sep=";")
df["radio"] = .5 * df.diametro
df
```

Out[160]:

	diametro	altura	volumen	tipo_de_arbol	radio
0	11.2	75.0	19.9	Cherrie Tree	5.60
1	11.3	79.0	24.2	Cherry Tree	5.65
2	11.4	76.0	21.0	Cherry Tree	5.70
3	11.4	76.0	21.4	Apple Tree	5.70
4	13.7	71.0	25.7	Cherry Tree	6.85
5	13.8	64.0	24.9	Cherry Tree	6.90
6	14.0	78.0	34.5	Cherrie Tree	7.00
7	14.2	80.0	31.7	Cherry Tree	7.10
8	NaN	74.0	36.3	Apple Tree	NaN
9	16.0	72.0	38.3	Cherry Tree	8.00
10	16.3	77.0	42.6	Cherry Tree	8.15
11	17.3	81.0	55.4	Apple Tree	8.65
12	17.5	NaN	55.7	Cherry Tree	8.75
13	17.9	80.0	58.3	Cherry Tree	8.95
14	18.0	80.0	51.5	Cherry Tree	9.00
15	18.0	NaN	51.0	NaN	9.00
16	20.6	NaN	NaN	Cherry Tree	10.30

In [161]:
```
df.area = np.pi * df.radio **2
df.columns
```

```
<ipython-input-161-79432ada6081>:1: UserWarning: Pandas doesn't allow colu
mns to be created via a new attribute name - see https://pandas.pydata.or
g/pandas-docs/stable/indexing.html#attribute-access
  df.area = np.pi * df.radio **2
```

Out[161]:
```
Index(['diametro', 'altura', 'volumen', 'tipo_de_arbol', 'radio'], dtype
='object')
```

5.2 Renombrando columnas

In [162]:
```
df = pd.read_csv("data/data.csv", sep=";")
print( df.columns )
df.columns = ["RaDiO","AlTuRa","VoLuMeN","TiPo_De_ArBoL"]
```

```
Index(['diametro', 'altura', 'volumen', 'tipo_de_arbol'], dtype='object')
```

In [163]:
```
print( df.columns )
```

```
Index(['RaDiO', 'AlTuRa', 'VoLuMeN', 'TiPo_De_ArBoL'], dtype='object')
```

In [164]:
```
df.columns = [col.lower() for col in df.columns]
print( df.columns )
```

```
Index(['radio', 'altura', 'volumen', 'tipo_de_arbol'], dtype='object')
```

5.3 Borrando columnas

In [165]:
```
df = pd.read_csv("data/data.csv", sep=";")
print( df.columns )
```

```
Index(['diametro', 'altura', 'volumen', 'tipo_de_arbol'], dtype='object')
```

In [166]:
```
df = df[["tipo_de_arbol","volumen", "diametro"]]
df
```

Out[166]:

	tipo_de_arbol	volumen	diametro
0	Cherrie Tree	19.9	11.2
1	Cherry Tree	24.2	11.3
2	Cherry Tree	21.0	11.4
3	Apple Tree	21.4	11.4
4	Cherry Tree	25.7	13.7
5	Cherry Tree	24.9	13.8
6	Cherrie Tree	34.5	14.0
7	Cherry Tree	31.7	14.2
8	Apple Tree	36.3	NaN
9	Cherry Tree	38.3	16.0
10	Cherry Tree	42.6	16.3
11	Apple Tree	55.4	17.3
12	Cherry Tree	55.7	17.5
13	Cherry Tree	58.3	17.9
14	Cherry Tree	51.5	18.0
15	NaN	51.0	18.0
16	Cherry Tree	NaN	20.6

In [167]:
```
df = df.drop("tipo_de_arbol", axis=1)
df
```

Out[167]:

	volumen	diametro
0	19.9	11.2
1	24.2	11.3
2	21.0	11.4
3	21.4	11.4
4	25.7	13.7
5	24.9	13.8
6	34.5	14.0
7	31.7	14.2
8	36.3	NaN
9	38.3	16.0
10	42.6	16.3
11	55.4	17.3
12	55.7	17.5
13	58.3	17.9
14	51.5	18.0
15	51.0	18.0
16	NaN	20.6

In [168]:

```
df.drop("diametro", axis=1, inplace=True)
df
```

Out[168]:

	volumen
0	19.9
1	24.2
2	21.0
3	21.4
4	25.7
5	24.9
6	34.5
7	31.7
8	36.3
9	38.3
10	42.6
11	55.4

	volumen
12	55.7
13	58.3
14	51.5
15	51.0
16	NaN

5.4 Agregando filas (indices)

In [169]:
```
df = pd.read_csv("data/data.csv", sep=";")
print( df.index )
df
```

```
RangeIndex(start=0, stop=17, step=1)
```

Out[169]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree

In [170]:
```
df = df.reindex( range(20) )
df
```

Out[170]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree

	diametro	altura	volumen	tipo_de_arbol
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree
17	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN
19	NaN	NaN	NaN	NaN

In [171]:

```
# Usando loc para acceder con notación de indices tradicional
df.loc[20, :] = [10, 20, 30, "CT"]
df
```

Out[171]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree

	diametro	altura	volumen	tipo_de_arbol
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree
17	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN
19	NaN	NaN	NaN	NaN
20	10.0	20.0	30.0	CT

5.5 Renombrando filas (índices)

In [172]:
```
df = pd.read_csv("data/data.csv", sep=";")
print(df.index)
```

```
RangeIndex(start=0, stop=17, step=1)
```

In [173]:
```
df.index = df.index + 10
print(df.index)
```

```
RangeIndex(start=10, stop=27, step=1)
```

In [174]:
```
df.index = ["i_%d"%idx for idx in df.index]
print(df.index)
```

```
Index(['i_10', 'i_11', 'i_12', 'i_13', 'i_14', 'i_15', 'i_16', 'i_17', 'i_
18',
       'i_19', 'i_20', 'i_21', 'i_22', 'i_23', 'i_24', 'i_25', 'i_26'],
      dtype='object')
```

5.6 Borrando indices

In [175]:
```
print(df.index)
print(df)
```

```
Index(['i_10', 'i_11', 'i_12', 'i_13', 'i_14', 'i_15', 'i_16', 'i_17', 'i_
18',
       'i_19', 'i_20', 'i_21', 'i_22', 'i_23', 'i_24', 'i_25', 'i_26'],
      dtype='object')
      diametro  altura  volumen tipo_de_arbol
i_10      11.2    75.0     19.9  Cherrie Tree
i_11      11.3    79.0     24.2   Cherry Tree
i_12      11.4    76.0     21.0   Cherry Tree
i_13      11.4    76.0     21.4    Apple Tree
i_14      13.7    71.0     25.7   Cherry Tree
i_15      13.8    64.0     24.9   Cherry Tree
i_16      14.0    78.0     34.5  Cherrie Tree
i_17      14.2    80.0     31.7   Cherry Tree
i_18       NaN    74.0     36.3    Apple Tree
i_19      16.0    72.0     38.3   Cherry Tree
i_20      16.3    77.0     42.6   Cherry Tree
i_21      17.3    81.0     55.4    Apple Tree
i_22      17.5     NaN     55.7   Cherry Tree
i_23      17.9    80.0     58.3   Cherry Tree
i_24      18.0    80.0     51.5   Cherry Tree
i_25      18.0     NaN     51.0          NaN
i_26      20.6     NaN      NaN   Cherry Tree
```

In [176]:
```
df = df.drop(["i_11","i_13","i_19"], axis=0)
print(df.index)
df
```

```
Index(['i_10', 'i_12', 'i_14', 'i_15', 'i_16', 'i_17', 'i_18', 'i_20', 'i_
21',
       'i_22', 'i_23', 'i_24', 'i_25', 'i_26'],
      dtype='object')
```

Out[176]:

	diametro	altura	volumen	tipo_de_arbol
i_10	11.2	75.0	19.9	Cherrie Tree
i_12	11.4	76.0	21.0	Cherry Tree
i_14	13.7	71.0	25.7	Cherry Tree
i_15	13.8	64.0	24.9	Cherry Tree
i_16	14.0	78.0	34.5	Cherrie Tree
i_17	14.2	80.0	31.7	Cherry Tree
i_18	NaN	74.0	36.3	Apple Tree
i_20	16.3	77.0	42.6	Cherry Tree
i_21	17.3	81.0	55.4	Apple Tree
i_22	17.5	NaN	55.7	Cherry Tree
i_23	17.9	80.0	58.3	Cherry Tree
i_24	18.0	80.0	51.5	Cherry Tree
i_25	18.0	NaN	51.0	NaN
i_26	20.6	NaN	NaN	Cherry Tree

In [177]:

```
df.drop(["i_24","i_25","i_26"], axis=0, inplace=True)
df
```

Out[177]:

	diametro	altura	volumen	tipo_de_arbol
i_10	11.2	75.0	19.9	Cherrie Tree
i_12	11.4	76.0	21.0	Cherry Tree
i_14	13.7	71.0	25.7	Cherry Tree
i_15	13.8	64.0	24.9	Cherry Tree
i_16	14.0	78.0	34.5	Cherrie Tree
i_17	14.2	80.0	31.7	Cherry Tree
i_18	NaN	74.0	36.3	Apple Tree
i_20	16.3	77.0	42.6	Cherry Tree
i_21	17.3	81.0	55.4	Apple Tree
i_22	17.5	NaN	55.7	Cherry Tree
i_23	17.9	80.0	58.3	Cherry Tree

In [178]:

```
df = df[-5:]
df
```

Out[178]:

	diametro	altura	volumen	tipo_de_arbol
i_18	NaN	74.0	36.3	Apple Tree
i_20	16.3	77.0	42.6	Cherry Tree
i_21	17.3	81.0	55.4	Apple Tree
i_22	17.5	NaN	55.7	Cherry Tree
i_23	17.9	80.0	58.3	Cherry Tree

Observación

```
# seleccionar la columna col
# regresa una serie
df[col]

# seleccionar las columnas col1, col2, ..., coln
# regresa dataframe
df[[col1,col2,.., coln]]

# selecciona solo el indice inicio
# regresa un dataframe
df[inicio:(inicio+1)]

# selecciona los indices en notacion
#regresa un dataframe
df[inicio:fin:salto]

# seleccion mixta
# regresa un dataframe
df.loc[inicio:fin:salto, col1:col2]
```

5.7 Masking

In [179]:
```
df = pd.read_csv("data/data.csv", sep=";")
vol_mean = df.volumen.mean()
vol_std = df.volumen.std()
```

In [180]:
```
mask_1 = df.altura < 80
mask_2 =  df.volumen <= vol_mean + vol_std
df1 = df[ mask_1 & mask_2 ]
df1
```

Out[180]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree

In [181]:
```
# Si se hace dinamicamente, utilizar suficientes parentesis
#df2 = df[ ((vol_mean - vol_std) <= df.volumen) & (df.volumen <= (vol_mean + vol_std) ) ]
df2 = df[ (df.volumen >=(vol_mean - vol_std)) & (df.volumen <= (vol_mean + vol_std) ) ]
df2
```

Out[181]:

	diametro	altura	volumen	tipo_de_arbol
1	11.3	79.0	24.2	Cherry Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree

	diametro	altura	volumen	tipo_de_arbol
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree

In [182]:

```
# A veces para simplificar numpy ayuda
mask_1 = df.volumen >= (vol_mean - vol_std)
mask_2 = df.volumen <= (vol_mean + vol_std)
mask = np.logical_and(mask_1, mask_2)
df3 = df[np.logical_not(mask)]
df3
```

Out[182]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree

5.8.- Grouping

In [183]:
```
df = pd.read_csv("data/data.csv", sep=";")
df.columns
```

Out[183]:
```
Index(['diametro', 'altura', 'volumen', 'tipo_de_arbol'], dtype='object')
```

In [184]:
```
g = df.groupby("tipo_de_arbol")
print( g )
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x125c3a5e0>
```

In [185]:
```
print( g.count() )
```

```
              diametro  altura  volumen
tipo_de_arbol
Apple Tree           2       3        3
Cherrie Tree         2       2        2
Cherry Tree         11       9       10
```

In [186]:
```
print( g.sum() ) # .mean(), .std()
```

```
              diametro  altura  volumen
tipo_de_arbol
Apple Tree        28.7   231.0    113.1
Cherrie Tree      25.2   153.0     54.4
Cherry Tree      170.7   679.0    373.9
```

In [187]:
```
# Ejemplo real
```

```
df[["tipo_de_arbol","diametro", "altura"]].groupby("tipo_de_arbol").mean()
```

Out[187]:

tipo_de_arbol	diametro	altura
Apple Tree	14.350000	77.000000
Cherrie Tree	12.600000	76.500000
Cherry Tree	15.518182	75.444444

5.9.- Imputación de datos

In [188]:

```
# Antes de imputar datos, siempre explorar
df.describe(include="all")
```

Out[188]:

	diametro	altura	volumen	tipo_de_arbol
count	16.000000	14.000000	16.000000	16
unique	NaN	NaN	NaN	3
top	NaN	NaN	NaN	Cherry Tree
freq	NaN	NaN	NaN	11
mean	15.162500	75.928571	37.025000	NaN
std	2.948418	4.615430	13.773816	NaN
min	11.200000	64.000000	19.900000	NaN
25%	13.125000	74.250000	24.725000	NaN
50%	15.100000	76.500000	35.400000	NaN
75%	17.600000	79.750000	51.125000	NaN
max	20.600000	81.000000	58.300000	NaN

In [189]:

```
# Imputación manual de datos (incorrecto)
df["tipo_de_arbol"][df.tipo_de_arbol=="Cherrie Tree"] = "Cherry Tree"
df
```

```
<ipython-input-189-f7d8d8ebc30b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df["tipo_de_arbol"][df.tipo_de_arbol=="Cherrie Tree"] = "Cherry Tree"
```

Out[189]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherry Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherry Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree

In [190]:

```
# Imputación manual de datos
df = pd.read_csv("data/data.csv", sep=";")
index_mask = (df.tipo_de_arbol=="Cherrie Tree")
df.loc[index_mask, "tipo_de_arbol"] = "Cherry Tree" # .loc es esencial
df
```

Out[190]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherry Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherry Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree

In [191]:

```
# Imputación de datos: llenar NaNs con promedio
df = pd.read_csv("data/data.csv", sep=";")
df1 = df.fillna(df.mean())
df1
```

Out[191]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2000	75.000000	19.900	Cherrie Tree
1	11.3000	79.000000	24.200	Cherry Tree
2	11.4000	76.000000	21.000	Cherry Tree
3	11.4000	76.000000	21.400	Apple Tree
4	13.7000	71.000000	25.700	Cherry Tree
5	13.8000	64.000000	24.900	Cherry Tree
6	14.0000	78.000000	34.500	Cherrie Tree
7	14.2000	80.000000	31.700	Cherry Tree
8	15.1625	74.000000	36.300	Apple Tree
9	16.0000	72.000000	38.300	Cherry Tree
10	16.3000	77.000000	42.600	Cherry Tree
11	17.3000	81.000000	55.400	Apple Tree
12	17.5000	75.928571	55.700	Cherry Tree
13	17.9000	80.000000	58.300	Cherry Tree
14	18.0000	80.000000	51.500	Cherry Tree
15	18.0000	75.928571	51.000	NaN
16	20.6000	75.928571	37.025	Cherry Tree

In [192]:

```
# Imputación de datos: llenar NaNs con valor
df2 = df.fillna(0)
df2
```

Out[192]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
8	0.0	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	0.0	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	0.0	51.0	0
16	20.6	0.0	0.0	Cherry Tree

In [193]:

```
# Imputación de datos: desechar filas con NaN
df3 = df.dropna()
df3
```

Out[193]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree

5.10 Apply

In [194]:
```
df = pd.read_csv("data/data.csv", sep=";")
df1 = df.diametro.apply(lambda x: x*2)
df1
```

Out[194]:
```
0     22.4
1     22.6
2     22.8
3     22.8
4     27.4
5     27.6
6     28.0
7     28.4
8      NaN
9     32.0
10    32.6
11    34.6
12    35.0
13    35.8
14    36.0
15    36.0
16    41.2
Name: diametro, dtype: float64
```

In [195]:

```
# Aplicación incorrecta
df2 = df["tipo_de_arbol"].apply(str.upper) # Error
df2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-195-1f598baa0fd5> in <module>
      1 # Aplicación incorrecta
----> 2 df2 = df["tipo_de_arbol"].apply(str.upper) # Error
      3 df2

/miniconda3/envs/meetup/lib/python3.9/site-packages/pandas/core/series.py in apply(self, func, convert_dtype, args, **kwds)
   4136             else:
   4137                 values = self.astype(object)._values
-> 4138                 mapped = lib.map_infer(values, f, convert=convert_dtype)
   4139
   4140         if len(mapped) and isinstance(mapped[0], Series):

pandas/_libs/lib.pyx in pandas._libs.lib.map_infer()

TypeError: descriptor 'upper' for 'str' objects doesn't apply to a 'float' object
```

In [196]:

```
# Aplicación correcta
df2 = df["tipo_de_arbol"].apply(lambda s: str(s).upper() )
df2
```

```
0      CHERRIE TREE
```

Out[196]:
```
1      CHERRY TREE
2      CHERRY TREE
3       APPLE TREE
4      CHERRY TREE
5      CHERRY TREE
6     CHERRIE TREE
7      CHERRY TREE
8       APPLE TREE
9      CHERRY TREE
10     CHERRY TREE
11      APPLE TREE
12     CHERRY TREE
13     CHERRY TREE
14     CHERRY TREE
15              NAN
16     CHERRY TREE
Name: tipo_de_arbol, dtype: object
```

In [197]:

```
# Error (o no?)
df3 = df.apply(lambda x: x*2)
df3
```

Out[197]:

	diametro	altura	volumen	tipo_de_arbol
0	22.4	150.0	39.8	Cherrie TreeCherrie Tree
1	22.6	158.0	48.4	Cherry TreeCherry Tree
2	22.8	152.0	42.0	Cherry TreeCherry Tree
3	22.8	152.0	42.8	Apple TreeApple Tree
4	27.4	142.0	51.4	Cherry TreeCherry Tree
5	27.6	128.0	49.8	Cherry TreeCherry Tree
6	28.0	156.0	69.0	Cherrie TreeCherrie Tree
7	28.4	160.0	63.4	Cherry TreeCherry Tree
8	NaN	148.0	72.6	Apple TreeApple Tree
9	32.0	144.0	76.6	Cherry TreeCherry Tree
10	32.6	154.0	85.2	Cherry TreeCherry Tree
11	34.6	162.0	110.8	Apple TreeApple Tree
12	35.0	NaN	111.4	Cherry TreeCherry Tree
13	35.8	160.0	116.6	Cherry TreeCherry Tree
14	36.0	160.0	103.0	Cherry TreeCherry Tree
15	36.0	NaN	102.0	NaN
16	41.2	NaN	NaN	Cherry TreeCherry Tree

Atajo

Para usar las operaciones de string en una columna de strings, es posible utilizar la siguiente notación para ahorrar espacio.

In [198]:
```
df.tipo_de_arbol.str.upper()
```

Out[198]:
```
0      CHERRIE TREE
1       CHERRY TREE
2       CHERRY TREE
3        APPLE TREE
4       CHERRY TREE
5       CHERRY TREE
6      CHERRIE TREE
7       CHERRY TREE
8        APPLE TREE
9       CHERRY TREE
10      CHERRY TREE
11       APPLE TREE
12      CHERRY TREE
13      CHERRY TREE
14      CHERRY TREE
15              NaN
16      CHERRY TREE
Name: tipo_de_arbol, dtype: object
```

In [199]:
```
df.tipo_de_arbol.str.len()
```

Out[199]:
```
0     12.0
1     11.0
2     11.0
3     10.0
4     11.0
5     11.0
6     12.0
7     11.0
8     10.0
9     11.0
10    11.0
11    10.0
12    11.0
13    11.0
14    11.0
15     NaN
16    11.0
Name: tipo_de_arbol, dtype: float64
```

In [200]:
```
df.tipo_de_arbol.str[3:-3]
```

Out[200]:
```
0     rrie T
1      rry T
2      rry T
3       le T
4      rry T
5      rry T
6     rrie T
7      rry T
8       le T
9      rry T
10     rry T
11      le T
12     rry T
13     rry T
14     rry T
15       NaN
16     rry T
Name: tipo_de_arbol, dtype: object
```

5.11 Merge

In [201]:
```
df1 = pd.read_csv("data/data.csv", sep=";")
df1
```

Out[201]:

	diametro	altura	volumen	tipo_de_arbol
0	11.2	75.0	19.9	Cherrie Tree
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
3	11.4	76.0	21.4	Apple Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
6	14.0	78.0	34.5	Cherrie Tree
7	14.2	80.0	31.7	Cherry Tree
8	NaN	74.0	36.3	Apple Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
11	17.3	81.0	55.4	Apple Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
15	18.0	NaN	51.0	NaN
16	20.6	NaN	NaN	Cherry Tree

In [202]:

```
df2 = pd.DataFrame(data={"tipo_de_arbol":["Cherry Tree", "Apple Tree", "Pear Tree"],
                         "fruto":["guinda","manzana", "pera"],
                         "precio_pesos_por_kg":[500, 2000, np.nan]})
df2
```

Out[202]:

	tipo_de_arbol	fruto	precio_pesos_por_kg
0	Cherry Tree	guinda	500.0
1	Apple Tree	manzana	2000.0
2	Pear Tree	pera	NaN

In [203]:

```
df3 = df1.merge(df2, how="left", on="tipo_de_arbol")
df3
```

Out[203]:

	diametro	altura	volumen	tipo_de_arbol	fruto	precio_pesos_por_kg
0	11.2	75.0	19.9	Cherrie Tree	NaN	NaN
1	11.3	79.0	24.2	Cherry Tree	guinda	500.0
2	11.4	76.0	21.0	Cherry Tree	guinda	500.0
3	11.4	76.0	21.4	Apple Tree	manzana	2000.0
4	13.7	71.0	25.7	Cherry Tree	guinda	500.0
5	13.8	64.0	24.9	Cherry Tree	guinda	500.0
6	14.0	78.0	34.5	Cherrie Tree	NaN	NaN
7	14.2	80.0	31.7	Cherry Tree	guinda	500.0
8	NaN	74.0	36.3	Apple Tree	manzana	2000.0
9	16.0	72.0	38.3	Cherry Tree	guinda	500.0
10	16.3	77.0	42.6	Cherry Tree	guinda	500.0
11	17.3	81.0	55.4	Apple Tree	manzana	2000.0
12	17.5	NaN	55.7	Cherry Tree	guinda	500.0
13	17.9	80.0	58.3	Cherry Tree	guinda	500.0
14	18.0	80.0	51.5	Cherry Tree	guinda	500.0
15	18.0	NaN	51.0	NaN	NaN	NaN
16	20.6	NaN	NaN	Cherry Tree	guinda	500.0

In [204]:

```
df3 = df1.merge(df2, how="right", on="tipo_de_arbol")
df3
```

Out[204]:

	diametro	altura	volumen	tipo_de_arbol	fruto	precio_pesos_por_kg
0	11.3	79.0	24.2	Cherry Tree	guinda	500.0
1	11.4	76.0	21.0	Cherry Tree	guinda	500.0
2	13.7	71.0	25.7	Cherry Tree	guinda	500.0
3	13.8	64.0	24.9	Cherry Tree	guinda	500.0
4	14.2	80.0	31.7	Cherry Tree	guinda	500.0
5	16.0	72.0	38.3	Cherry Tree	guinda	500.0
6	16.3	77.0	42.6	Cherry Tree	guinda	500.0
7	17.5	NaN	55.7	Cherry Tree	guinda	500.0
8	17.9	80.0	58.3	Cherry Tree	guinda	500.0
9	18.0	80.0	51.5	Cherry Tree	guinda	500.0
10	20.6	NaN	NaN	Cherry Tree	guinda	500.0
11	11.4	76.0	21.4	Apple Tree	manzana	2000.0
12	NaN	74.0	36.3	Apple Tree	manzana	2000.0
13	17.3	81.0	55.4	Apple Tree	manzana	2000.0
14	NaN	NaN	NaN	Pear Tree	pera	NaN

In [205]:

```
df3 = df1.merge(df2, how="inner", on="tipo_de_arbol")
df3
```

Out[205]:

	diametro	altura	volumen	tipo_de_arbol	fruto	precio_pesos_por_kg
0	11.3	79.0	24.2	Cherry Tree	guinda	500.0
1	11.4	76.0	21.0	Cherry Tree	guinda	500.0
2	13.7	71.0	25.7	Cherry Tree	guinda	500.0
3	13.8	64.0	24.9	Cherry Tree	guinda	500.0
4	14.2	80.0	31.7	Cherry Tree	guinda	500.0
5	16.0	72.0	38.3	Cherry Tree	guinda	500.0
6	16.3	77.0	42.6	Cherry Tree	guinda	500.0
7	17.5	NaN	55.7	Cherry Tree	guinda	500.0
8	17.9	80.0	58.3	Cherry Tree	guinda	500.0
9	18.0	80.0	51.5	Cherry Tree	guinda	500.0
10	20.6	NaN	NaN	Cherry Tree	guinda	500.0
11	11.4	76.0	21.4	Apple Tree	manzana	2000.0
12	NaN	74.0	36.3	Apple Tree	manzana	2000.0
13	17.3	81.0	55.4	Apple Tree	manzana	2000.0

In [206]:

```
df3 = df1.merge(df2, how="outer", on="tipo_de_arbol")
df3
```

Out[206]:

	diametro	altura	volumen	tipo_de_arbol	fruto	precio_pesos_por_kg
0	11.2	75.0	19.9	Cherrie Tree	NaN	NaN
1	14.0	78.0	34.5	Cherrie Tree	NaN	NaN
2	11.3	79.0	24.2	Cherry Tree	guinda	500.0
3	11.4	76.0	21.0	Cherry Tree	guinda	500.0
4	13.7	71.0	25.7	Cherry Tree	guinda	500.0
5	13.8	64.0	24.9	Cherry Tree	guinda	500.0
6	14.2	80.0	31.7	Cherry Tree	guinda	500.0
7	16.0	72.0	38.3	Cherry Tree	guinda	500.0
8	16.3	77.0	42.6	Cherry Tree	guinda	500.0
9	17.5	NaN	55.7	Cherry Tree	guinda	500.0
10	17.9	80.0	58.3	Cherry Tree	guinda	500.0
11	18.0	80.0	51.5	Cherry Tree	guinda	500.0
12	20.6	NaN	NaN	Cherry Tree	guinda	500.0
13	11.4	76.0	21.4	Apple Tree	manzana	2000.0
14	NaN	74.0	36.3	Apple Tree	manzana	2000.0
15	17.3	81.0	55.4	Apple Tree	manzana	2000.0
16	18.0	NaN	51.0	NaN	NaN	NaN
17	NaN	NaN	NaN	Pear Tree	pera	NaN

Guardando datos

1. **csv**
2. **json**
3. **excel**

   Lo más importante es tener cuidado de cómo se guardan los nombres de las columnas (header), y el indice (index).

   Depende de la utilización, pero mi recomendación es guardar el header explícitamente y guardar el index como una columna.

In [207]:

```
# guardar un csv
df = pd.read_csv("data/data.csv", sep=";")
df = df[df.tipo_de_arbol=="Cherry Tree"]
df.to_csv("data/output.csv", sep="|", index=True) # header=True by default
df
```

Out[207]:

	diametro	altura	volumen	tipo_de_arbol
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
7	14.2	80.0	31.7	Cherry Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
16	20.6	NaN	NaN	Cherry Tree

In [208]:

```
# Leer el csv anterior
df2 = pd.read_csv("data/output.csv", sep="|", index_col=0) # get index from first column
df2
```

Out[208]:

	diametro	altura	volumen	tipo_de_arbol
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
7	14.2	80.0	31.7	Cherry Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
16	20.6	NaN	NaN	Cherry Tree

In [209]:

```
%%bash
cat data/output.csv
```

```
|diametro|altura|volumen|tipo_de_arbol
1|11.3|79.0|24.2|Cherry Tree
2|11.4|76.0|21.0|Cherry Tree
4|13.7|71.0|25.7|Cherry Tree
5|13.8|64.0|24.9|Cherry Tree
7|14.2|80.0|31.7|Cherry Tree
9|16.0|72.0|38.3|Cherry Tree
10|16.3|77.0|42.6|Cherry Tree
12|17.5||55.7|Cherry Tree
13|17.9|80.0|58.3|Cherry Tree
14|18.0|80.0|51.5|Cherry Tree
16|20.6|||Cherry Tree
```

In [210]:

```
# guardar un json
df = pd.read_csv("data/data.csv", sep=";")
df = df[df.tipo_de_arbol=="Cherry Tree"]
df.to_json("data/output.json")
df
```

Out[210]:

	diametro	altura	volumen	tipo_de_arbol
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
7	14.2	80.0	31.7	Cherry Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
16	20.6	NaN	NaN	Cherry Tree

In [211]:

```
# Leyendo el json anterior
df2 = pd.read_json("data/output.json")
df2
```

Out[211]:

	diametro	altura	volumen	tipo_de_arbol
1	11.3	79.0	24.2	Cherry Tree
2	11.4	76.0	21.0	Cherry Tree
4	13.7	71.0	25.7	Cherry Tree
5	13.8	64.0	24.9	Cherry Tree
7	14.2	80.0	31.7	Cherry Tree
9	16.0	72.0	38.3	Cherry Tree
10	16.3	77.0	42.6	Cherry Tree
12	17.5	NaN	55.7	Cherry Tree
13	17.9	80.0	58.3	Cherry Tree
14	18.0	80.0	51.5	Cherry Tree
16	20.6	NaN	NaN	Cherry Tree

In [212]:

```
%%bash
cat data/output.json
```


```
{"diametro":{"1":11.3,"2":11.4,"4":13.7,"5":13.8,"7":14.2,"9":16.0,"10":1
6.3,"12":17.5,"13":17.9,"14":18.0,"16":20.6},"altura":{"1":79.0,"2":76.
0,"4":71.0,"5":64.0,"7":80.0,"9":72.0,"10":77.0,"12":null,"13":80.0,"14":8
0.0,"16":null},"volumen":{"1":24.2,"2":21.0,"4":25.7,"5":24.9,"7":31.
7,"9":38.3,"10":42.6,"12":55.7,"13":58.3,"14":51.5,"16":null},"tipo_de_arb
ol":{"1":"Cherry Tree","2":"Cherry Tree","4":"Cherry Tree","5":"Cherry Tre
e","7":"Cherry Tree","9":"Cherry Tree","10":"Cherry Tree","12":"Cherry Tre
e","13":"Cherry Tree","14":"Cherry Tree","16":"Cherry Tree"}}
```

Desafío para la casa

Descargar algún archivo de interés:

- Abrir el archivo.
- Explorar los datos
- Visualizar los datos
- Completar los datos incompletos
- Guardar el archivo

¡Gracias por la atención!