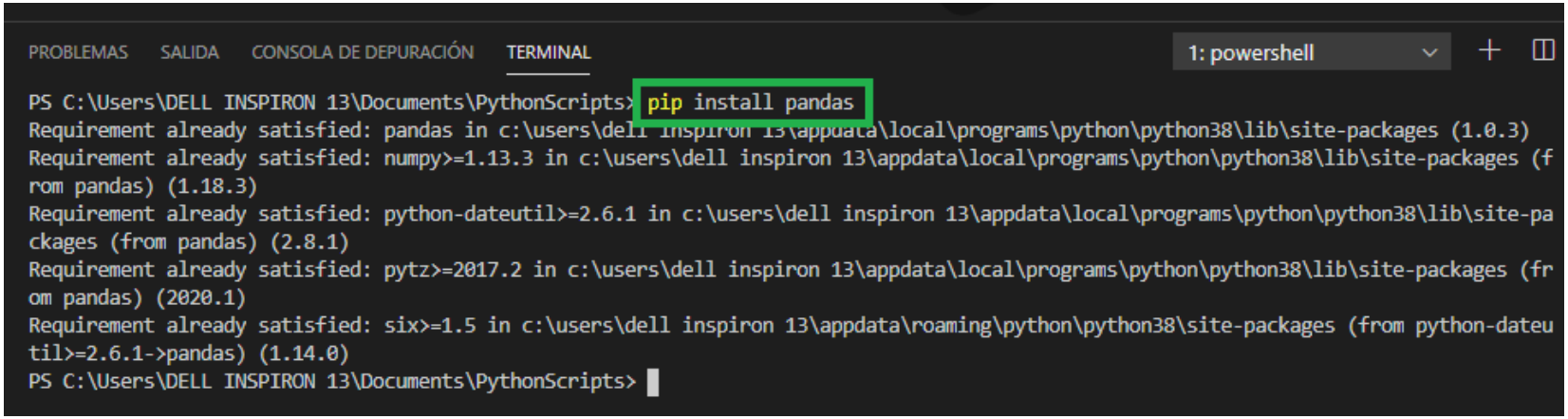
PYTHON
ANÁLISIS Y MANIPULACIÓN DE DATOS CON PANDAS

Pandas es una biblioteca de Python de código abierto que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar para el lenguaje de programación Python. Con Pandas, podemos lograr cinco pasos típicos en el procesamiento y análisis de datos, independientemente del origen de los datos: cargar, preparar, manipular, modelar y analizar.

La distribución estándar de Python no viene incluida con el módulo Pandas. Para instalar

pip install pandas



Estructura de datos	Dimensiones	Descripción
Serie	1	Matriz homogénea etiquetada 1D, tamaño inmutable.
Data Frame	2	Estructura tabular etiquetada en 2D general de tamaño variable con columnas con tipos potencialmente heterogéneos.

SERIES

Serie: Es una matriz unidimensional como estructura con datos homogéneos. Por ejemplo:

10	23	56	17	52	61	73	90	26	72

Se puede crear una serie de pandas utilizando el siguiente constructor:

pandas.Series (data, index, dtype, copy)

Se puede crear una serie usando varias entradas como: narray (array de numpy) , dict, valor escalar o constante.

Crear una serie vacía: Una serie básica, que se puede crear es una serie vacía.

```
#ejemplo 1
import pandas as pd
s = pd.Series()
print (s)
```

```
>>> s = pd.Series()
>>> print (s)
Series([], dtype: float64)
```

Crear una serie desde un array

```
# ejemplo 2
import pandas as pd

data = ['a','b','c','d']
s = pd.Series(data)
print (s)
```

```
>>> import pandas as pd
>>> data = ['a','b','c','d']
>>> s = pd.Series(data)
>>> print (s)
0    a
1    b
2    c
3    d
dtype: object
```

No pasamos ningún índice, por lo que, por defecto, asignó los índices que van desde 0 a len (datos) -1 , es decir, 0 a 3.

```
# ejemplo 3
import pandas as pd

data = ['a','b','c','d']
s = pd.Series(data, index=[100,101,102,103])
print (s)
```

```
>>> import pandas as pd
>>> data = ['a','b','c','d']
>>> s = pd.Series(data, index=[100,101,102,103])
>>> print (s)
100    a
101    b
102    c
103    d
dtype: object
```

Pasamos los valores del índice aquí.

Crear una serie desde dict : Se puede pasar un **dict** como entrada y, si no se especifica ningún índice, las claves del diccionario se toman para construir el índice. Si se pasa el **índice,** se extraerán los valores en los datos correspondientes a las etiquetas del índice.

```
# ejemplo 4
import pandas as pd

data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data)
print (s)
```

```
>>> import pandas as pd
>>> data = {'a' : 0., 'b' : 1., 'c' : 2.}
>>> s = pd.Series(data)
>>> print (s)
a    0.0
b    1.0
c    2.0
dtype: float64
```

Observa : las claves del diccionario se utilizan para construir el índice

```
# ejemplo 5
import pandas as pd

data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data,index=['b','c','d','a'])
print (s)
```

```
>>> import pandas as pd
>>> data = {'a' : 0., 'b' : 1., 'c' : 2.}
>>> s = pd.Series(data,index=['b','c','d','a'])
>>> print (s)
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

Observa : el orden del índice persiste y el elemento que falta se llena con NaN (no es un número).

Crear una serie desde escalar: Si los datos son un valor escalar, se debe proporcionar un índice. El valor se repetirá para que coincida con la longitud del **índice.**

```
# ejemplo  6
import pandas as pd

s = pd.Series(5, index=[0, 1, 2, 3])
print (s)
```

```
>>> import pandas as pd
>>> s = pd.Series(5, index=[0, 1, 2, 3])
>>> print (s)
0    5
1    5
2    5
3    5
dtype: int64
```

Acceso a datos de series con posición: Se puede acceder a los datos de la serie de forma similar a la de un **ndarray** (array de numpy)

```
# ejemplo  7
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

print(s[2])
```

```
>>> import pandas as pd
>>> s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
>>> print(s[2])
3
>>>
```

Recupera el tercer elemento. El conteo comienza desde cero .

```
#ejemplo 8
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

print(s[:3])
```

```
>>> import pandas as pd
>>> s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
>>> print(s[:3])
a    1
b    2
c    3
dtype: int64
```

Recupera los primeros tres elementos de la serie. Si se inserta un: (dos puntos) delante de él, se extraerán todos los elementos de ese índice en adelante. Si se utilizan dos parámetros (con: entre ellos), se recuperan los elementos entre los dos índices (sin incluir el índice de detención)

```
#ejemplo 9
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

print(s[-3:])
```

```
>>> import pandas as pd
>>> s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
>>> print(s[-3:])
c    3
d    4
e    5
dtype: int64
```

Recupera los últimos tres elementos.

Recuperar datos usando etiqueta (índice): Una serie es como un **dict de** tamaño fijo en el que puede obtener y establecer valores por etiqueta de índice.

```
#ejemplo 10
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

print(s['a'])
```

Recupera un solo elemento utilizando el valor de la etiqueta de índice. Si la etiqueta no existe genera una excepción.

```
>>> import pandas as pd
>>> s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
>>> print(s['a'])
1
```

```
#ejemplo 11
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

print(s[['a','c','d']])
```

```
>>> import pandas as pd
>>> s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
>>> print(s[['a','c','d']])
a    1
c    3
d    4
dtype: int64
```

Recupera múltiples elementos usando una lista de valores de etiquetas de índice.

FUNCIONES BÁSICAS

Creando una serie con numpy

```
#ejemplo 12
import pandas as pd
import numpy as np

#Crea una serie con 4 números aleatorios
s = pd.Series(np.random.randn(4))
print(s)
```

```
>>> import pandas as pd
>>>
>>> import numpy as np
>>> #Crea una serie con 4 números aleatorios
>>>
>>> s = pd.Series(np.random.randn(4))
>>> print(s)
0    0.647484
1   -1.060567
2    0.012874
3   -0.274968
dtype: float64
```

```
#ejemplo 13
import pandas as pd
import numpy as np

#Crea una serie con 4 números aleatorios
s = pd.Series(np.random.randn(4))
print (s)
print ("Los ejes son:")
print (s.axes)
```

```
>>> print (s)
0   -0.668413
1    0.080682
2   -0.036217
3   -1.114682
dtype: float64
>>> print ("Los ejes son:")
Los ejes son:
>>> print (s.axes)
[RangeIndex(start=0, stop=4, step=1)]
```

```
# ejemplo 14
import pandas as pd
import numpy as np

s = pd.Series(np.random.randn(4))
print ("Está el objeto vacío?")
print(s.empty)
```

```
>>> import pandas as pd
>>> import numpy as np
>>> s = pd.Series(np.random.randn(4))
>>> print ("Está el objeto vacío?")
Está el objeto vacío?
>>> print(s.empty)
False
```

Devuelve el valor booleano que indica si el objeto está vacío o no. Verdadero indica que el objeto está vacío

```
# ejemplo 15
import pandas as pd
import numpy as np

s = pd.Series(np.random.randn(4))
print(s)

print ("Las dimensiones del objeto:")
print(s.ndim)
```

```
>>> import pandas as pd
>>> import numpy as np
>>> s = pd.Series(np.random.randn(4))
>>> print(s)
0   -0.017973
1   -0.107946
2    0.957022
3   -0.563237
dtype: float64
>>> print ("Las dimensiones del objeto:")
Las dimensiones del objeto:
>>> print(s.ndim)
1
```

Devuelve el número de dimensiones del objeto. Por definición, una serie es una estructura de datos 1D, por lo que devuelve 1

```
# ejemplo 16
import pandas as pd
import numpy as np

s = pd.Series(np.random.randn(4))
print(s)

print ("El tamaño del objeto es:")
print(s.size)
```

```
>>> print(s)
0    0.003213
1   -0.507613
2    0.813989
3   -0.261951
dtype: float64
>>> print ("El tamaño del objeto es:")
El tamaño del objeto es:
>>> print(s.size)
4
```

Devuelve el tamaño (longitud) de la serie

```
# ejemplo 17
import pandas as pd
import numpy as np

s = pd.Series(np.random.randn(4))
print(s)

print ("La serie de datos es:")
print(s.values)
```

```
>>> print(s)
0    0.854273
1    0.213471
2    0.970398
3    0.462194
dtype: float64
>>> print ("La serie de datos es:")
La serie de datos es:
>>> print(s.values)
[0.85427327 0.21347114 0.97039839 0.462194  ]
```

Devuelve los datos reales de la serie como una matriz.

```
# ejemplo 18
import pandas as pd
import numpy as np

s = pd.Series(np.random.randn(4))
print(s)

print ("Las primeras dos filas son:")
print(s.head(2))
```

```
>>> print(s)
0    0.977151
1    1.363346
2   -1.056581
3    1.525959
dtype: float64
>>> print ("Las primeras 2 filas son:")
Las primeras 2 filas son:
>>> print(s.head(2))
0    0.977151
1    1.363346
dtype: float64
```

head () devuelve las primeras n filas (observa los valores de índice).

```
# ejemplo 19
import pandas as pd
import numpy as np

s = pd.Series(np.random.randn(4))
print(s)

print ("Las últimas dos filas son:")
print(s.tail(2))
```

```
>>> print(s)
0   -1.240003
1    1.379748
2   -0.288336
3    0.418449
dtype: float64
>>> print ("Las últimas dos filas son:")
Las últimas dos filas son:
>>> print(s.tail(2))
2   -0.288336
3    0.418449
dtype: float64
```

tail () devuelve las últimas n filas (observe los valores de índice). El número predeterminado de elementos para mostrar es 5, pero se puede pasar un número personalizado.

DATA FRAME

Data Frame

Es una matriz bidimensional con datos heterogéneos. Por ejemplo:

Columnas

Filas

Nombre (cadena)	Años (entero)	Género(cadena)	Clasificación(flotante)
Pepe	32	Masculino	3,45
Lia	28	Femenino	4.6
Vicente	45	Masculino	3.9
Karina	38	Femenino	2,78

Características de DataFrame

- Potencialmente las columnas son de diferentes tipos de datos
- Tamaño: mutable
- Ejes etiquetados (filas y columnas)
- Puede realizar operaciones aritméticas en filas y columnas

Se puede crear un DataFrame de pandas utilizando el siguiente constructor:

pandas.DataFrame(data, index, columns, dtype, copy)

Se puede crear un DataFrame de pandas usando varias entradas como: list, dict, Series, arrays de Numpy (ndarrays), otro DataFrame

Crear un Data Frame vacío

```
# ejemplo 1
import pandas as pd

df = pd.DataFrame()

print(df)
```

```
>>> import pandas as pd
>>> df = pd.DataFrame()
>>> print(df)
Empty DataFrame
Columns: []
Index: []
```

Crear un marco de datos a partir de listas

```
# ejemplo 2
import pandas as pd

data = [1,2,3,4,5]
df = pd.DataFrame(data)

print(df)
```

```
>>> import pandas as pd
>>> data = [1,2,3,4,5]
>>> df = pd.DataFrame(data)
>>> print(df)
   0
0  1
1  2
2  3
3  4
4  5
```

```
# ejemplo 3
import pandas as pd

data = [['Ale',10],['Pepe',12],['Zule',13]]
df = pd.DataFrame(data,columns=['Nombre','Edad'])

print (df)
```

```
>>> import pandas as pd
>>> data = [['Ale',10],['Pepe',12],['Zule',13]]
>>> df = pd.DataFrame(data,columns=['Nombre','Edad'])
>>> print (df)
  Nombre  Edad
0    Ale    10
1   Pepe    12
2   Zule    13
```

```
# ejemplo 4
import pandas as pd

data = [['Ale',10],['Pepe',12],['Zule',13]]
df = pd.DataFrame(data,columns=['Nombre','Edad'], dtype=float)

print (df)
```

```
>>> import pandas as pd
>>> data = [['Ale',10],['Pepe',12],['Zule',13]]
>>> df = pd.DataFrame(data,columns=['Nombre','Edad'], dtype=float)
>>> print (df)
  Nombre  Edad
0    Ale  10.0
1   Pepe  12.0
2   Zule  13.0
```

Observa que el parámetro dtype cambia el tipo de de datos de la columna

Crear un DataFrame desde dict :

Todos los **arrays** deben ser de la misma longitud. Si se pasa el índice, entonces la longitud del índice debe ser igual a la longitud de las matrices. Si no se pasa ningún índice, entonces, por defecto, el índice será el rango (n), donde **n** es la longitud de la matriz.

```
# ejemplo  5
import pandas as pd

data = {'Nombre':['Tomy', 'Juan', 'Silvio', 'Ricky'],'Edad':[28,34,29,42]}
df = pd.DataFrame(data)

print (df)
```

```
>>> import pandas as pd
>>> data = {'Nombre':['Tomy', 'Juan', 'Silvio', 'Ricky'],'Edad':[28,34,29,42]}
>>> df = pd.DataFrame(data)
>>> print (df)
   Nombre  Edad
0    Tomy    28
1    Juan    34
2  Silvio    29
3   Ricky    42
```

Observa los valores 0,1,2,3. Son el índice predeterminado asignado a cada uno utilizando el rango de funciones (n).

```
# ejemplo  6
import pandas as pd

data = {'Nombre':['Tomy', 'Juan', 'Silvio', 'Ricky'],'Edad':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])

print (df)
```

```
>>> import pandas as pd
>>> data = {'Nombre':['Tomy', 'Juan', 'Silvio', 'Ricky'],'Edad':[28,34,29,42]}
>>> df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
>>> print (df)
      Nombre  Edad
rank1    Tomy    28
rank2    Juan    34
rank3  Silvio    29
rank4   Ricky    42
```

Observa que index asigna una etiqueta a cada fila.

Crear un Data Frame de una lista de dict: La lista de diccionarios se puede pasar como datos de entrada para crear un DataFrame. Las claves del diccionario se toman por defecto como nombres de columna.

```
# ejemplo 7
import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)

print (df)
```

```
>>> import pandas as pd
>>> data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
>>> df = pd.DataFrame(data)
>>> print (df)
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

Observa que NaN (no es un número) se agrega en las áreas faltantes

```
# ejemplo 8
import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data, index=['primera', 'segunda'])

print (df)
```

```
>>> import pandas as pd
>>> data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
>>> df = pd.DataFrame(data, index=['primera', 'segunda'])
>>> print (df)
         a   b     c
primera  1   2   NaN
segunda  5  10  20.0
>>>
```

El ejemplo muestra cómo crear un DataFrame pasando una lista de diccionarios y las etiquetas de los índices de fila.

```
# ejemplo 9
import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

df1 = pd.DataFrame(data, index=['primera', 'segunda'], columns=['a', 'b'])

df2 = pd.DataFrame(data, index=['primera', 'segunda'], columns=['a', 'b1'])

print (df1)
print (df2)
```

```
>>> import pandas as pd
>>> data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
>>> df1 = pd.DataFrame(data, index=['primera', 'segunda'], columns=['a', 'b'])
>>> df2 = pd.DataFrame(data, index=['primera', 'segunda'], columns=['a', 'b1'])
>>> print (df1)
          a   b
primera   1   2
segunda   5  10
>>> print (df2)
          a  b1
primera   1 NaN
segunda   5 NaN
>>>
```

Observa que mientras el Data Frame df1 se crea con índices de columna iguales a las claves del diccionario, por lo que no se agrega NaN. el Data Frame df2 se crea con un índice de columna que no es clave del diccionario, así se agregaron los NaN.

Crear un Data Frame desde dict of Series: El diccionario de series se puede pasar para formar un Data Frame. El índice resultante es la unión de todos los índices de serie pasados.

```
# ejemplo  10
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print(df)
```

```
>>> import pandas as pd
>>>
>>> d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
...    'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
>>>
>>> df = pd.DataFrame(d)
>>> print(df)
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
```

Observa que, para la serie one, no se ha pasado la etiqueta 'd' , pero en el resultado, para la etiqueta d , se agrega un NaN.

Selección, adición y eliminación de columnas

Selección de columna

```
# ejemplo 11
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print (df)
print (df['one'])
```

```
>>> import pandas as pd
>>>
>>> d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
...    'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
>>>
>>> df = pd.DataFrame(d)
>>> print (df)
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
>>> print (df['one'])
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
>>>
```

Agregar una columna

```
# ejemplo 12
import pandas as pd

d = {'uno' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'dos' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print ("Agregar una nueva columna pasada como Series:\n")
df['tres']=pd.Series([10,20,30],index=['a','b','c'])
print(df)

print ("Agregar una nueva columna usando las columnas existentes en DataFrame:\n")
df['cuatro']=df['uno'] + df['tres']

print(df)
```

```
Agregar una nueva columna pasada como Series:

>>> df['tres']=pd.Series([10,20,30],index=['a','b','c'])
>>> print(df)
   uno  dos  tres
a  1.0    1  10.0
b  2.0    2  20.0
c  3.0    3  30.0
d  NaN    4   NaN
>>> print ("Agregar una nueva columna usando las columnas existentes en DataFrame:\n")
Agregar una nueva columna usando las columnas existentes en DataFrame:

>>> df['cuatro']=df['uno']+df['tres']
>>> print(df)
   uno  dos  tres  cuatro
a  1.0    1  10.0    11.0
b  2.0    2  20.0    22.0
c  3.0    3  30.0    33.0
d  NaN    4   NaN     NaN
```

Eliminar una columna

```
# ejemplo 13
import pandas as pd

d = {'uno' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'dos' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd']),
   'tres' : pd.Series([10,20,30], index=['a','b','c'])}

df = pd.DataFrame(d)
print ("Nuestro Data Frame es:")
print (df)

print ("Eliminar la primera columna usando la función del:\n")
del df['uno']
print(df)

print ("Eliminar otra columna usando la función pop:\n")
df.pop('dos')
print(df)
```

```
Nuestro Data Frame es:
>>> print (df)
   uno  dos  tres
a  1.0    1  10.0
b  2.0    2  20.0
c  3.0    3  30.0
d  NaN    4   NaN
>>> print ("Eliminar la primera columna usando la función del:\n")
Eliminar la primera columna usando la función del:

>>> del df['uno']
>>> print(df)
   dos  tres
a    1  10.0
b    2  20.0
c    3  30.0
d    4   NaN
>>> print ("Eliminar otra columna usando la función pop:\n")
Eliminar otra columna usando la función pop:

>>> df.pop('dos')
a    1
b    2
c    3
d    4
Name: dos, dtype: int64
>>> print(df)
   tres
a  10.0
b  20.0
c  30.0
d   NaN
```

Selección de fila, adición y eliminación

Selección por etiqueta
Las filas se pueden seleccionar pasando la etiqueta de fila a una función **loc**

```
# ejemplo 14
import pandas as pd

d = {'uno' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'dos' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print (df)
print (df.loc['b'])
```

```
>>> print (df)
   uno  dos
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
>>> print (df.loc['b'])
uno    2.0
dos    2.0
Name: b, dtype: float64
```

El resultado es una serie con etiquetas como los nombres de columna del DataFrame. Y el nombre de la serie es la etiqueta con la que se recupera

Selección por ubicación entera

Las filas se pueden seleccionar pasando la ubicación entera a una función **iloc**

```
# ejemplo 15
import pandas as pd

d = {'uno' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'dos' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print(df)
print (df.iloc[2])
```

```
>>> print(df)
   uno  dos
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
>>> print (df.iloc[2])
uno    3.0
dos    3.0
Name: c, dtype: float64
>>>
```

Selección de varias filas

Se pueden seleccionar varias filas con el operador ':‘.

```
# ejemplo 16
import pandas as pd

d = {'uno' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'dos' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print(df)
print (df[2:4])
```

```
>>> print(df)
   uno  dos
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
>>> print (df[2:4])
   uno  dos
c  3.0    3
d  NaN    4
```

Adición de filas

Agregue nuevas filas a un DataFrame usando la función **append** . Esta función agregará las filas al final.

```
# ejemplo 17
import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
print(df)

df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])

df = df.append(df2)
print (df)
```

```
>>> print(df)
   a  b
0  1  2
1  3  4
>>> df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])
>>> df = df.append(df2)
>>> print (df)
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
```

Supresión de filas
Si la etiqueta está duplicada, se eliminarán varias filas. Si observas el ejemplo anterior, las etiquetas están duplicadas.

```
# ejemplo 18
import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])
print(df)

df = df.append(df2)
print(df)

df = df.drop(0)
print(df)
```

```
>>> print(df)
   a  b
0  1  2
1  3  4
>>> df = df.append(df2)
>>> print(df)
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
>>> df = df.drop(0)
>>> print(df)
   a  b
1  3  4
1  7  8
```

Se eliminaron 2 filas porque esas dos contienen la misma etiqueta 0

FUNCIONES BÁSICAS

```
# ejemplo 19
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestra serie de datos es:")
print(df)
```

```
Nuestro Data Frame de datos es:
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>>
```

Creamos un DataFrame a partir de 3 series.

```
# ejemplo 20
import pandas as pd
import numpy as np

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio'
,'Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("La transposición de datos es:")
print(df.T)
```

```
>>> print(df.T)
           0     1      2      3       4     5     6
Nombre  Tomy  Juan  Ricky  Vilma  Silvio  Sara  José
Edad      25    26     25     23      30    29    23
Rating  4.23  3.24   3.98   2.56     3.2   4.6   3.8
>>>
```

```
# ejemplo 21
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Las etiquetas de eje de fila y las etiquetas de eje de columna son:")
print(df.axes)
```

axes devuelve la lista de etiquetas de eje de fila y etiquetas de eje de columna.

```
>>> print ("Las etiquetas de eje de fila y las etiquetas de eje de columna son:")
Las etiquetas de eje de fila y las etiquetas de eje de columna son:
>>> print(df.axes)
[RangeIndex(start=0, stop=7, step=1), Index(['Nombre', 'Edad', 'Rating'], dtype='object')]
>>>
```

```
# ejemplo 22
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Los tipos de datos de cada columna son:")
print(df.dtypes)
```

```
Los tipos de datos de cada columna son:
>>> print(df.dtypes)
Nombre     object
Edad        int64
Rating    float64
dtype: object
```

```
# ejemplo 23
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Está vacío el objeto?")
print(df.empty)
```

```
>>> print ("Está vacío el objeto?")
Está vacío el objeto?
>>> print(df.empty)
False
```

```
# ejemplo 24
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print ("La dimensión del objeto es:")
print(df.ndim)
```

```
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print ("La dimensión del objeto es:")
La dimensión del objeto es:
>>> print(df.ndim)
2
```

```
# ejemplo 25
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print ("La forma del objeto es:")
print(df.shape)
```

```
Nuestro objeto es:
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print ("La forma del objeto es:")
La forma del objeto es:
>>> print(df.shape)
(7, 3)
```

shape devuelve una tupla que representa la dimensión del Data Frame. Tupla (a, b), donde a representa el número de filas y b representa el número de columnas

```
# ejemplo 26
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print ("El número total de elementos en nuestro objeto es:")
print(df.size)
```

```
Nuestro objeto es:
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print ("El número total de elementos en nuestro objeto es:")
El número total de elementos en nuestro objeto es:
>>> print(df.size)
21
```

size, devuelve la cantidad de elementos del DataFrame

```
# ejemplo 27
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print ("Los datos en nuestro data frame son:")
print(df.values)
```

```
Nuestro objeto es:
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print ("Los datos en nuestro data frame son:")
Los datos en nuestro data frame son:
>>> print(df.values)
[['Tomy' 25 4.23]
 ['Juan' 26 3.24]
 ['Ricky' 25 3.98]
 ['Vilma' 23 2.56]
 ['Silvio' 30 3.2]
 ['Sara' 29 4.6]
 ['José' 23 3.8]]
```

Devuelve los datos del DataFrame como un ndarray

```
# ejemplo 28
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print ("Las primeras dos filas son:")
print(df.head(2))
print ("Las últimas tres filas son:")
print(df.tail(3))
```

```
>>> print ("Nuestro objeto es:")
Nuestro objeto es:
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print ("Las primeras dos filas son:")
Las primeras dos filas son:
>>> print(df.head(2))
  Nombre  Edad  Rating
0   Tomy    25    4.23
1   Juan    26    3.24
>>> print ("Las últimas tres filas son:")
Las últimas tres filas son:
>>> print(df.tail(3))
   Nombre  Edad  Rating
4  Silvio    30     3.2
5    Sara    29     4.6
6    José    23     3.8
```

head () devuelve las primeras n filas (observa los valores de índice). El número predeterminado de elementos para mostrar es 5, pero puedes pasar un número personalizado, **tail()** devuelve las últimas n filas.

Una gran cantidad de métodos calculan **estadísticas descriptivas** y otras operaciones relacionadas con el DataFrame.

orden	Función	Descripción
1	count()	Número de observaciones no nulas
2	sum()	Suma de valores
3	mean()	Media de valores
4	median()	Mediana de valores
5	mode()	Modo de valores
6	std ()	Desviación estándar de Bressel de los valores
7	min ()	Valor mínimo
8	max ()	Valor máximo
9	abs()	Valor absoluto
10	prod()	Producto de valores
11	cumsum ()	Suma acumulativa
12	cumprod ()	Producto acumulativo

```python
# ejemplo 29
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print ("Suma:")
print(df.sum())
print ("Suma del eje 1:")
print(df.sum(1))
print ("La media es:")
print(df.mean())
print ("La desviación estándar es:")
print(df.std())
```

```
>>> print(df.sum())
Nombre    TomyJuanRickyVilmaSilvioSaraJosé
Edad                                   181
Rating                               25.61
dtype: object
>>> print ("Suma del eje 1:")
Suma del eje 1:
>>> print(df.sum(1))
0    29.23
1    29.24
2    28.98
3    25.56
4    33.20
5    33.60
6    26.80
dtype: float64
>>> print ("La media es:")
La media es:
>>> print(df.mean())
Edad      25.857143
Rating     3.658571
dtype: float64
>>> print ("La desviación estándar es:")
La desviación estándar es:
>>> print(df.std())
Edad      2.734262
Rating    0.698628
dtype: float64
```

```
# ejemplo 30
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print(df.describe())
```

```
Nuestro objeto es:
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print(df.describe())
            Edad    Rating
count   7.000000  7.000000
mean   25.857143  3.658571
std     2.734262  0.698628
min    23.000000  2.560000
25%    24.000000  3.220000
50%    25.000000  3.800000
75%    27.500000  4.105000
max    30.000000  4.600000
>>>
```

La función **describe()** calcula un resumen de estadísticas pertenecientes a las columnas DataFrame.

Esta función proporciona
los valores **medios, estándar** e **IQR** . Excluye las columnas de caracteres y el resumen dado es sobre columnas numéricas

```
# ejemplo 31
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print(df.describe(include=['object']))
```

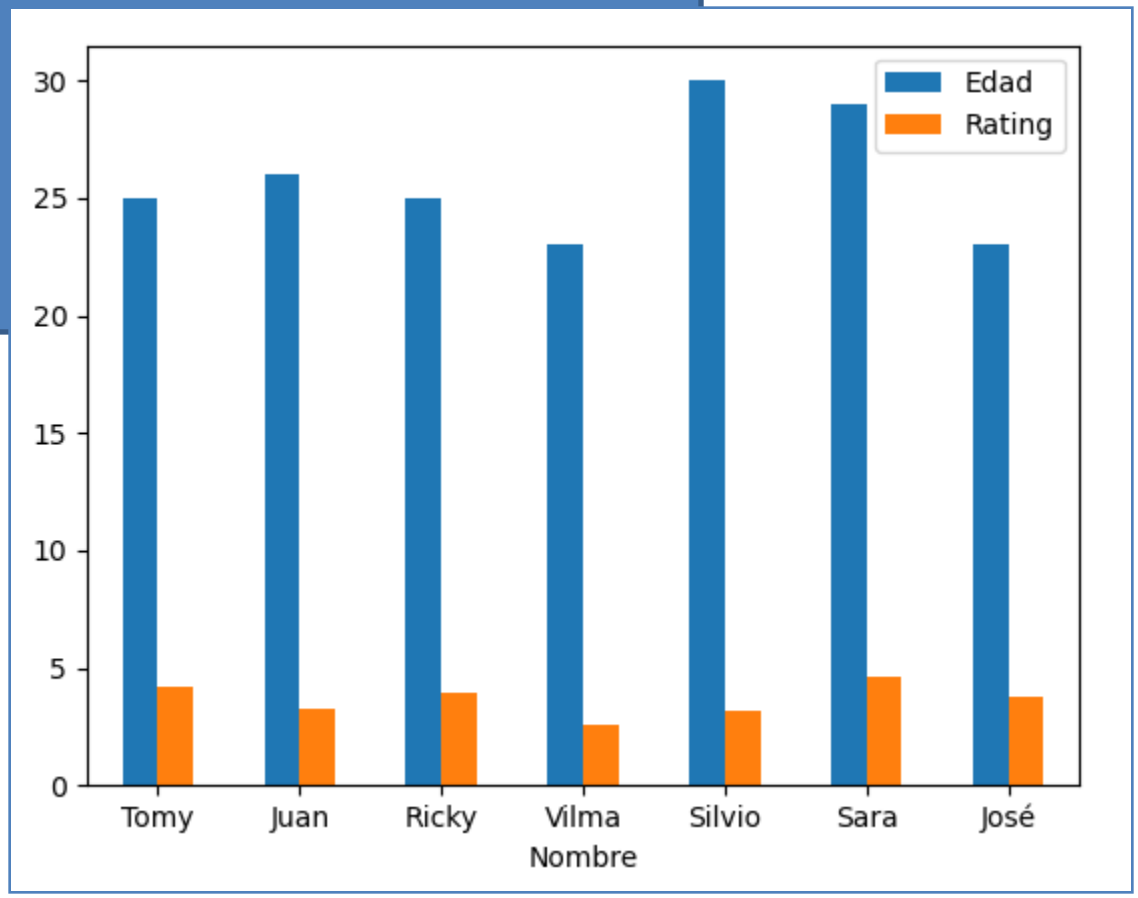
```
>>> print(df)
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print(df.describe(include=['object']))
        Nombre
count        7
unique       7
top       Tomy
freq         1
>>>
```

include()es el argumento que se utiliza para transmitir la información necesaria sobre qué columnas se debe resumir.

```
# ejemplo 32
import pandas as pd

d = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)
print ("Nuestro objeto es:")
print(df)
print(df.describe(include='all'))
```

```
   Nombre  Edad  Rating
0    Tomy    25    4.23
1    Juan    26    3.24
2   Ricky    25    3.98
3   Vilma    23    2.56
4  Silvio    30    3.20
5    Sara    29    4.60
6    José    23    3.80
>>> print(df.describe(include='all'))
       Nombre       Edad    Rating
count       7   7.000000  7.000000
unique      7        NaN       NaN
top      Tomy        NaN       NaN
freq        1        NaN       NaN
mean      NaN  25.857143  3.658571
std       NaN   2.734262  0.698628
min       NaN  23.000000  2.560000
25%       NaN  24.000000  3.220000
50%       NaN  25.000000  3.800000
75%       NaN  27.500000  4.105000
max       NaN  30.000000  4.600000
>>>
```

```
# ejemplo 33

import matplotlib.pyplot as plt
import pandas as pd

data = {'Nombre':pd.Series(['Tomy','Juan','Ricky','Vilma','Silvio','Sara','José']),
   'Edad':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}
df = pd.DataFrame(data)

df.plot.bar('Nombre', rot=0)

plt.show()
```



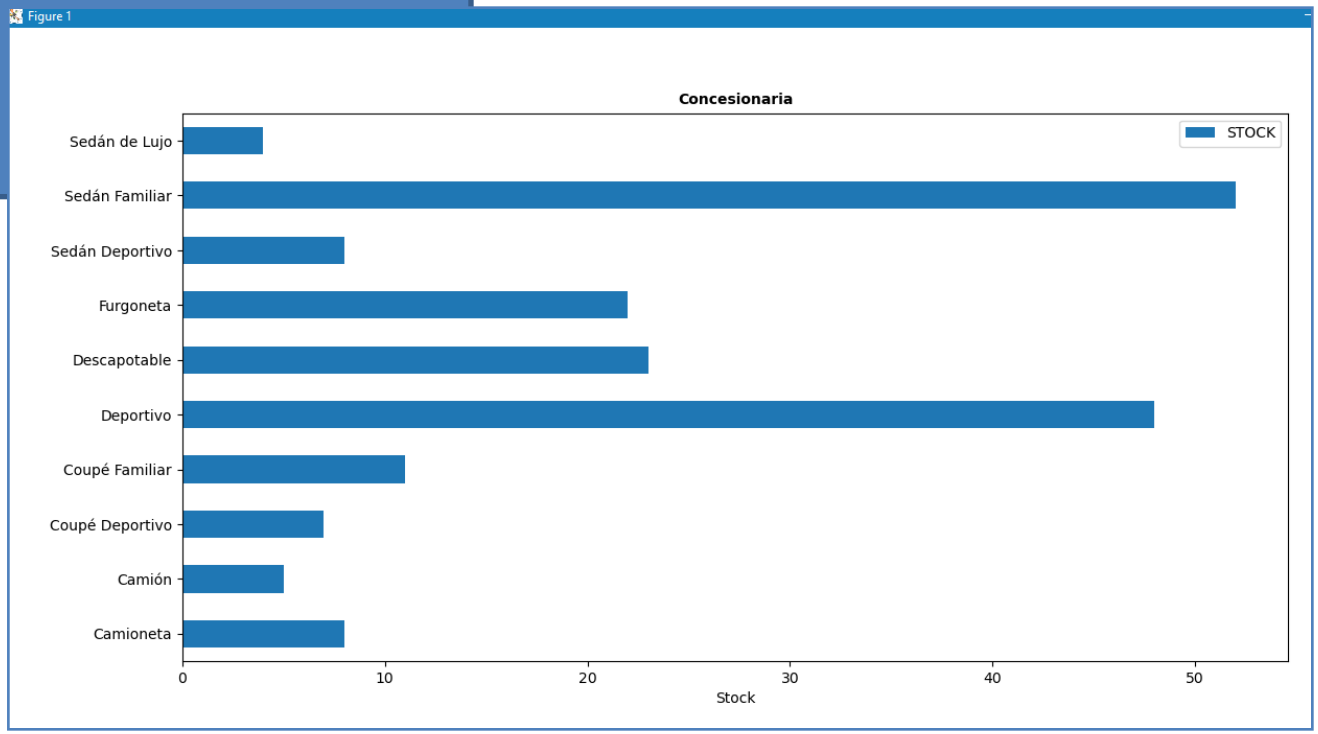
```
# ejemplo 34
import pandas as pd
import matplotlib.pyplot as plt

url='https://www.tiobe.com/tiobe-index/‘

web=pd.read_html(url, header=0)[2]

df = pd.DataFrame(web)

print(df)
```

```
>>> print(df)
                 Type Mutability                                        Description                                      Syntax examples
0                bool  immutable                                      Boolean value                                            TrueFalse
1           bytearray    mutable                                  Sequence of bytes  bytearray(b'Some ASCII')bytearray(b"Some ASCII...
2               bytes  immutable                                  Sequence of bytes  b'Some ASCII'b"Some ASCII"bytes([119, 105, 107...
3             complex  immutable       Complex number with real and imaginary parts                                            3+2.7j
4                dict    mutable  Associative array (or dictionary) of key and v...                       {'key1': 1.0, 3: False}{}
5           ellipsisa  immutable  An ellipsis placeholder to be used as an index...                                       ...Ellipsis
6               float  immutable  Double precision floating point number. The pr...                                         3.1415927
7           frozenset  immutable  Unordered set, contains no duplicates; can con...                frozenset([4.0, 'string', True])
8                 int  immutable                Integer of unlimited magnitude[86]                                              42
9                list    mutable                     List, can contain mixed types                       [4.0, 'string', True][]
10           NoneTypea  immutable  An object representing the absence of a value,...                                            None
11  NotImplementedTypea  immutable  A placeholder that can be returned from overlo...                                   NotImplemented
12               range  immutable  A Sequence of numbers commonly used for loopin...                 range(1, 10)range(10, -5, -2)
13                 set    mutable  Unordered set, contains no duplicates; can con...                       {4.0, 'string', True}set()
14                 str  immutable  A character string: sequence of Unicode codepo...  'Wikipedia'"Wikipedia"""Spanning multiple lin...
15               tuple  immutable                          Can contain mixed types         (4.0, 'string', True)('single element',)()
```

Nota: según el caso puede necesitar instalar alguno de los siguientes módulos:

```
pip install lxml
pip install html5lib
pip install beautifulsoup4
```

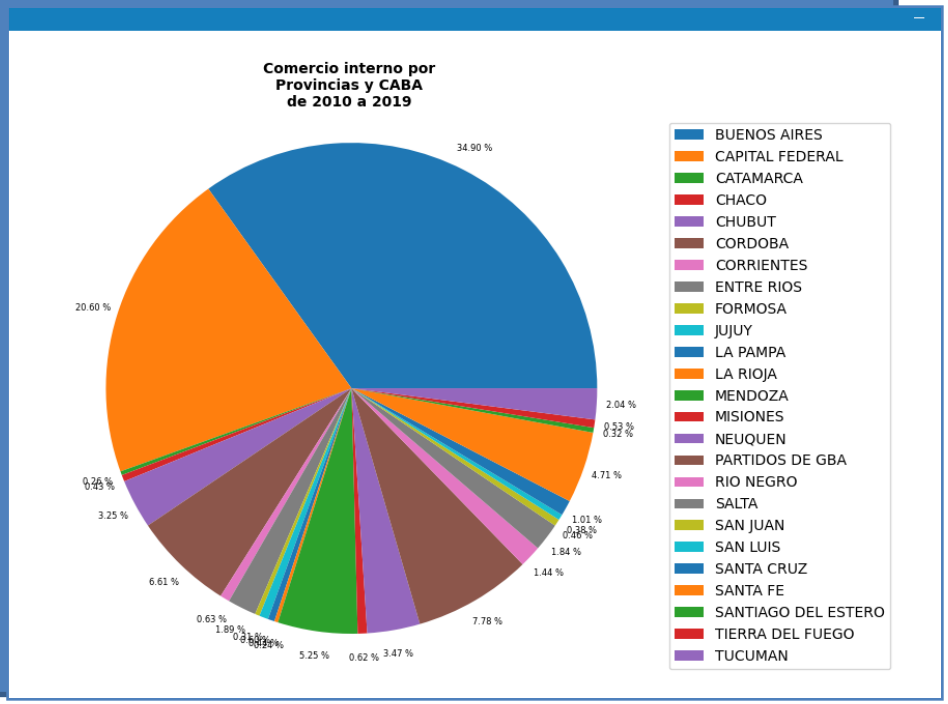
```
# ejemplo 35
import pandas as pd
import matplotlib.pyplot as plt

xls = pd.ExcelFile(‘<ruta>/autos.xlsx')
print(xls.sheet_names)

df = xls.parse('autos')

print(df)

df.groupby('TIPO')['STOCK'].sum().plot(kind='barh',
                        legend='Reverse')
plt.xlabel('Stock')
plt.title('Concesionaria',
        weight='bold',
        size=10)
plt.show()
```



Nota: según el caso puede necesitar instalar el siguiente módulo:

```
pip install xlrd
pip install openpyxl
```

```
# ejemplo 36
import pandas as pd
import matplotlib.pyplot as plt
import datetime
datos = pd.read_csv('<ruta>/comercio_interno.csv', encoding='latin-1')
df = pd.DataFrame(datos)
df = df.drop(df[df['alcance_nombre'] == 'Argentina'].index)
df = df.drop(df[df['alcance_nombre'] == 'GRAN BUENOS AIRES'].index)
df = df.drop(df[df['alcance_nombre'] == 'INDETERMINADA'].index)
df = df.drop(df[df['alcance_nombre'] == 'PARTIDOS DEL GBA'].index)
df = df.drop(df[df['alcance_nombre'] == 'RESTO DE BUENOS AIRES'].index)
df['indice_tiempo'] = pd.to_datetime(df['indice_tiempo'], format='%d/%m/%Y')
df['año'], df['mes'] = df['indice_tiempo'].dt.year, df['indice_tiempo'].dt.month
df = df.drop(df[df['año'] < 2010].index)
del df['sector_id']
df.to_csv('comercio-interno-2.csv')
df.groupby('alcance_nombre')['valor'].sum().plot
                                    (kind='pie',
                                     legend='Reverse',
                                     autopct='%0.2f %%',
                                     fontsize=6,
                                     labels=None,
                                     pctdistance=1.10)
plt.axis('equal')
plt.ylabel('')
plt.title('Comercio interno por\nProvincias y CABA\n
de 2010 a 2019',
        weight='bold',
        size=10)
plt.show()
```



```
# ejemplo 37
import pandas as pd

userHeader = ['user_id', 'gender', 'age', 'ocupation', 'zip']
users = pd.read_table('<ruta>/dataset/users.txt', engine='python', sep='::', header=None, names=userHeader)
print(users.head())

movieHeader = ['movie_id', 'title', 'genders']
movies = pd.read_table('<ruta>/dataset/movies.txt', engine='python', sep='::', header=None, names=movieHeader)
print(movies.head())

ratingHeader = ['user_id', 'movie_id', 'rating', 'timestamp']
ratings = pd.read_table('<ruta>/dataset/ratings.txt', engine='python', sep='::', header=None, names=ratingHeader)
print(ratings.head())

mergeRatings = pd.merge(users, ratings)
print(mergeRatings.head())
mergeRatings = pd.merge(pd.merge(users, ratings), movies)
print(mergeRatings.head())

info1000 = mergeRatings.loc[1000]
print ('Info de la posición 1000  en la tabla:',info1000)
info5_97 = mergeRatings[5:97]
print ('Info de la posición 5 a la 96 en la tabla::',info5_97)

numberRatings = mergeRatings.groupby('title').size().sort_values(ascending=False)
print ('Primeras 10 películas con más votos:', numberRatings[:10])

avgRatings = mergeRatings.groupby(['movie_id', 'title']).mean()
print ('Media del rating:', avgRatings['rating'][:10])

dataRatings = mergeRatings.groupby(['movie_id', 'title'])['rating'].agg(['mean', 'sum', 'count', 'std'])
print ('Info estadística del rating:', dataRatings[:10])
```

https://pypi.org/project/pandas/

https://pandas.pydata.org/

https://pandas.pydata.org/docs/pandas.pdf