Librería PANDAS

Creación de una serie a partir de una lista

Series(data=lista, index=indices, dtype=tipo) : Devuelve un objeto de tipo Series con los datos de la lista lista, las filas especificados en la lista índices y el tipo de datos indicado en tipo. Si no se pasa la lista de índices se utilizan como índices los enteros del 0 al , donde es el tamaño de la serie. Si no se pasa el tipo de dato se infiere.

```
>>> import pandas as pd
>>> s = pd.Series(['Matemáticas', 'Historia', 'Economía', 'Programación', 'Inglés'],
dtype='string')
>>> print(s)
0     Matemáticas
1        Historia
2        Economía
3    Programación
4          Inglés
dtype: string
```

Creación de una serie a partir de un diccionario

Series(data=diccionario, index=indices): Devuelve un objeto de tipo Series con los valores del diccionario diccionario y las filas especificados en la lista indices. Si no se pasa la lista de índices se utilizan como índices las claves del diccionario.

```
>>> import pandas as pd
>>> s = pd.Series({'Matemáticas': 6.0,  'Economía': 4.5, 'Programación': 8.5})
>>> print(s)
Matemáticas     6.0
Economía        4.5
Programación    8.5
dtype: float64
```

Atributos de una serie

Existen varias propiedades o métodos para ver las características de una serie.

s.size : Devuelve el número de elementos de la serie s.
s.index : Devuelve una lista con los nombres de las filas del DataFrame s.
s.dtype : Devuelve el tipo de datos de los elementos de la serie s.

```
>>> import pandas as pd
>>> s = pd.Series([1, 2, 2, 3, 3, 3, 4, 4, 4, 4])
>>> s.size
```

```
10
>>> s.index
RangeIndex(start=0, stop=10, step=1)
>>> s.dtype
dtype('int64')
```

Acceso a los elementos de una serie

El acceso a los elementos de un objeto del tipo Series puede ser a través de posiciones o través de índices (nombres).

Acceso por posición

Se realiza de forma similar a como se accede a los elementos de un array.

s[i] : Devuelve el elemento que ocupa la posición i+1 en la serie s.
s[posiciones]: Devuelve otra serie con los elementos que ocupan las posiciones de la lista posiciones.

Acceso por índice

s[nombre] : Devuelve el elemento con el nombre nombre en el índice.
s[nombres] : Devuelve otra serie con los elementos correspondientes a los nombres indicadas en la lista nombres en el índice.

```
>>> s[1:3]
Economía       4.5
Programación   8.5
dtype: float64

>>> s['Economía']
4.5
>>> s[['Programación', 'Matemáticas']]
Programación   8.5
Matemáticas    6.0
dtype: float64
```

Resumen descriptivo de una serie

Las siguientes funciones permiten resumir varios aspectos de una serie:

s.count() : Devuelve el número de elementos que no son nulos ni NaN en la serie s.
s.sum() : Devuelve la suma de los datos de la serie s cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str.

s.cumsum() : Devuelve una serie con la suma acumulada de los datos de la serie s cuando los datos son de un tipo numérico.
s.value_counts() : Devuelve una serie con la frecuencia (número de repeticiones) de cada valor de la serie s.
s.min() : Devuelve el menor de los datos de la serie s.
s.max() : Devuelve el mayor de los datos de la serie s.
s.mean() : Devuelve la media de los datos de la serie s cuando los datos son de un tipo numérico.
s.std() : Devuelve la desviación típica de los datos de la serie s cuando los datos son de un tipo numérico.
s.describe(): Devuelve una serie con un resumen descriptivo que incluye el número de datos, su suma, el mínimo, el máximo, la media, la desviación típica y los cuartiles.

```
>>> import pandas as pd
>>> s = pd.Series([1, 1, 1, 1, 2, 2, 2, 3, 3, 4])
>>> s.count()
10
>>> s.sum()
20
>>> s.cumsum()
0    1
1    2
2    3
3    4
4    6
5    8
6   10
7   13
8   16
9   20
dtype: int64

>>> s.value_counts()
1   4
2   3
3   2
4   1
dtype: int64

>>> s.value_counts(normalize=True)
1   0.4
2   0.3
3   0.2
4   0.1
```

```
dtype: float64

>>> s.min()
1

>>> s.max()
4

>>> s.mean()
2.0

>>> s.std()
1.0540925533894598

>>> s.describe()
count    10.000000
mean      2.000000
std       1.054093
min       1.000000
25%       1.000000
50%       2.000000
75%       2.750000
max       4.000000

dtype: float64
```

Aplicar operaciones a una serie

Los operadores aritméticos (+, *, /, etc.) pueden utilizarse con una serie, y devuelven otra serie con el resultado de aplicar la operación a cada elemento de la serie.

```
>>> import pandas as pd
s = pd.Series([1, 2, 3, 4])
>>> s*2
0    2
1    4
2    6
3    8
dtype: int64

>>> s%2
0    1
1    0
2    1
```

```
3    0
dtype: int64

>>> s = pd.Series(['a', 'b', 'c'])

>>> s*5
0    aaaaa
1    bbbbb
2    ccccc
dtype: object
```

Aplicar funciones a una serie

También es posible aplicar una función a cada elemento de la serie mediante el siguiente método:

s.apply(f) : Devuelve una serie con el resultado de aplicar la función f a cada uno de los elementos de la serie s.

```
>>> import pandas as pd
>>> from math import log
>>> s = pd.Series([1, 2, 3, 4])
>>> s.apply(log)
0    0.000000
1    0.693147
2    1.098612
3    1.386294
dtype: float64

>>> s = pd.Series(['a', 'b', 'c'])
>>> s.apply(str.upper)
0    A
1    B
2    C
dtype: object
```

Filtrado de una serie

Para filtrar una serie y quedarse con los valores que cumplen una determinada condición se utiliza el siguiente método:

s[condicion] : Devuelve una serie con los elementos de la serie s que se corresponden con el valor True de la lista booleana condicion. condicion debe ser una lista de valores booleanos de la misma longitud que la serie.

```
>>> import pandas as pd
>>> s = pd.Series({'Matemáticas': 6.0,  'Economía': 4.5, 'Programación': 8.5})

>>> print(s[s > 5])
Matemáticas     6.0
Programación    8.5
dtype: float64
```

Ordenar una serie

Para ordenar una serie se utilizan los siguientes métodos:

s.sort_values(ascending=booleano) : Devuelve la serie que resulta de ordenar los valores la serie s. Si argumento del parámetro ascending es True el orden es creciente y si es False decreciente.

df.sort_index(ascending=booleano) : Devuelve la serie que resulta de ordenar el índice de la serie s. Si el argumento del parámetro ascending es True el orden es creciente y si es False decreciente.

```
>>> import pandas as pd
>>> s = pd.Series({'Matemáticas': 6.0,  'Economía': 4.5, 'Programación': 8.5})
>>> print(s.sort_values())
Economía        4.5
Matemáticas     6.0
Programación    8.5
dtype: float64

>>> print(s.sort_index(ascending = False))
Programación    8.5
Matemáticas     6.0
Economía        4.5
dtype: float64
```

Eliminar los dados desconocidos en una serie

Los datos desconocidos representan en Pandas por NaN y los nulos por None. Tanto unos como otros suelen ser un problema a la hora de realizar algunos análisis de datos, por lo que es habitual eliminarlos. Para eliminarlos de una serie se utiliza el siguiente método:

s.dropna() : Elimina los datos desconocidos o nulos de la serie s.

```
>>> import pandas as pd
```

```
>>> import numpy as np
>>> s = pd.Series(['a', 'b', None, 'c', np.NaN,  'd'])
>>> s
0      a
1      b
2   None
3      c
4    NaN
5      d
dtype: object

>>> s.dropna()
0   a
1   b
3   c
5   d
dtype: object
```

La clase de objetos DataFrame

Un objeto del tipo DataFrame define un conjunto de datos estructurado en forma de tabla donde cada columna es un objeto de tipo Series, es decir, todos los datos de una misma columna son del mismo tipo, y las filas son registros que pueden contender datos de distintos tipos.

Un DataFrame contiene dos índices, uno para las filas y otro para las columnas, y se puede acceder a sus elementos mediante los nombres de las filas y las columnas.

Ejemplo. El siguiente DataFrame contiene información sobre los alumnos de un curso. Cada fila corresponde a un alumno y cada columna a una variable.
Creación de un DataFrame a partir de un diccionario de listas

Para crear un DataFrame a partir de un diccionario cuyas claves son los nombres de las columnas y los valores son listas con los datos de las columnas se utiliza el método:

DataFrame(data=diccionario, index=filas, columns=columnas, dtype=tipos) : Devuelve un objeto del tipo DataFrame cuyas columnas son las listas contenidas en los valores del diccionario diccionario, los nombres de filas indicados en la lista filas, los nombres de columnas indicados en la lista columnas y los tipos indicados en la lista tipos. La lista filas tiene que tener el mismo tamaño que las listas del diccionario, mientras que las listas columnas y tipos tienen que tener el mismo tamaño que el diccionario. Si no se pasa la lista de filas se utilizan como nombres los enteros empezando en 0. Si no se pasa la lista de columnas se utilizan como nombres las claves del diccionario. Si no se pasa la lista de tipos, se infiere.

Los valores asociados a las claves del diccionario deben ser listas del mismo tamaño.

```
>>> import pandas as pd
>>> datos = {'nombre':['María', 'Luis', 'Carmen', 'Antonio'],
... 'edad':[18, 22, 20, 21],
... 'grado':['Economía', 'Medicina', 'Arquitectura', 'Economía'],
... 'correo':['maria@gmail.com', 'luis@yahoo.es', 'carmen@gmail.com',
'antonio@gmail.com']
... }
>>> df = pd.DataFrame(datos)
>>> print(df)
   nombre  edad        grado            correo
0    María    18     Economía    maria@gmail.com
1     Luis    22     Medicina      luis@yahoo.es
2   Carmen    20  Arquitectura   carmen@gmail.com
3  Antonio    21     Economía  antonio@gmail.com
```

Creación de un DataFrame a partir de una lista de listas
Para crear un DataFrame a partir de una lista de listas con los datos de las columnas se utiliza el siguiente método:

DataFrame(data=listas, index=filas, columns=columnas, dtype=tipos) : Devuelve un objeto del tipo DataFrame cuyas columnas son los valores de las listas de la lista listas, los nombres de filas indicados en la lista filas, los nombres de columnas indicados en la lista columnas y los tipos indicados en la lista tipos. La lista filas, tiene que tener el mismo tamaño que la lista listas mientras que las listas columnas y tipos tienen que tener el mismo tamaño que las listas anidadas en listas. Si no se pasa la lista de filas o de columnas se utilizan enteros empezando en 0. Si no se pasa la lista de tipos, se infiere.

Si las listas anidadas en listas no tienen el mismo tamaño, las listas menores se rellenan con valores NaN.

```
>>> import pandas as pd
>>> df = pd.DataFrame([['María', 18], ['Luis', 22], ['Carmen', 20]], columns=['Nombre',
'Edad'])
>>> print(df)
  Nombre   Edad
0   María     18
1    Luis     22
2  Carmen     20
```

Creación de un DataFrame a partir de una lista de diccionarios

Para crear un DataFrame a partir de una lista de diccionarios con los datos de las filas, se utiliza el siguiente método:

DataFrame(data=diccionarios, index=filas, columns=columnas, dtype=tipos) : Devuelve un objeto del tipo DataFrame cuyas filas contienen los valores de los diccionarios de la lista diccionarios, los nombres de filas indicados en la lista filas, los nombres de columnas indicados en la lista columnas y los tipos indicados en la lista tipos. La lista filas tiene que tener el mismo tamaño que la lista lista. Si no se pasa la lista de filas se utilizan enteros empezando en 0. Si no se pasa la lista de columnas se utilizan las claves de los diccionarios. Si no se pasa la lista de tipos, se infiere.
Si los diccionarios no tienen las mismas claves, las claves que no aparecen en el diccionario se rellenan con valores NaN.

```
>>> import pandas as pd
>>> df = pd.DataFrame([{'Nombre':'María', 'Edad':18}, {'Nombre':'Luis', 'Edad':22},
{'Nombre':'Carmen'}])
>>> print(df)
0   María  18.0
1    Luis  22.0
2  Carmen   NaN
```

Creación de un DataFrame a partir de un array

Para crear un DataFrame a partir de un array de NumPy se utiliza el siguiente método:

DataFrame(data=array, index=filas, columns=columnas, dtype=tipo) : Devuelde un objeto del tipo DataFrame cuyas filas y columnas son las del array array, los nombres de filas indicados en la lista filas, los nombres de columnas indicados en la lista columnas y el tipo indicado en tipo. La lista filas tiene que tener el mismo tamaño que el número de filas del array y la lista columnas el mismo tamaño que el número de columnas del array. Si no se pasa la lista de filas se utilizan enteros empezando en 0. Si no se pasa la lista de columnas se utilizan las claves de los diccionarios. Si no se pasa la lista de tipos, se infiere.

```
>>> import pandas as pd
>>> df = pd.DataFrame(np.random.randn(4, 3), columns=['a', 'b', 'c'])
>>> print(df)
          a         b         c
0 -1.408238  0.644706  1.077434
1 -0.279264 -0.249229  1.019137
2 -0.805470 -0.629498  0.935066
3  0.236936 -0.431673 -0.177379
```

Creación de un DataFrame a partir de un fichero CSV o Excel

Dependiendo del tipo de fichero, existen distintas funciones para importar un DataFrame desde un fichero.

read_csv(fichero.csv, sep=separador, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal) : Devuelve un objeto del tipo DataFrame con los datos del fichero CSV fichero.csv usando como separador de los datos la cadena separador. Como nombres de columnas se utiliza los valores de la fila n y como nombres de filas los valores de la columna m. Si no se indica m se utilizan como nombres de filas los enteros empezando en 0. Los valores incluídos en la lista no-validos se convierten en NaN. Para los datos numéricos se utiliza como separador de decimales el carácter indicado en separador-decimal.

read_excel(fichero.xlsx, sheet_name=hoja, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal) : Devuelve un objeto del tipo DataFrame con los datos de la hoja de cálculo hoja del fichero Excel fichero.xlsx. Como nombres de columnas se utiliza los valores de la fila n y como nombres de filas los valores de la columna m. Si no se indica m se utilizan como nombres de filas los enteros empezando en 0. Los valores incluídos en la lista no-validos se convierten en NaN. Para los datos numéricos se utiliza como separador de decimales el carácter indicado en separador-decimal.

```
>>> import pandas as pd
>>> # Importación del fichero datos-colesteroles.csv
>>> df = pd.read_csv(
'ruta al archio', sep=';', decimal=',')
>>> print(df.head())
                      nombre  edad sexo    peso    altura  colesterol
0     José Luis Martínez Izquierdo    18    H    85.0    1.79        182.0
1                Rosa Díaz Díaz    32    M    65.0    1.73        232.0
2           Javier García Sánchez    24    H     NaN    1.81        191.0
3            Carmen López Pinzón    35    M    65.0    1.70        200.0
4            Marisa López Collado    46    M    51.0    1.58        148.0
```

Exportación de ficheros

También existen funciones para exportar un DataFrame a un fichero con diferentes formatos.

df.to_csv(fichero.csv, sep=separador, columns=booleano, index=booleano) : Exporta el DataFrame df al fichero fichero.csv en formato CSV usando como separador de los datos la cadena separador. Si se pasa True al parámetro columns se exporta también la fila con

los nombres de columnas y si se pasa True al parámetro index se exporta también la columna con los nombres de las filas.

df.to_excel(fichero.xlsx, sheet_name = hoja, columns=booleano, index=booleano) : Exporta el DataFrame df a la hoja de cálculo hoja del fichero fichero.xlsx en formato Excel. Si se pasa True al parámetro columns se exporta también la fila con los nombres de columnas y si se pasa True al parámetro index se exporta también la columna con los nombres de las filas.

Atributos de un DataFrame

Existen varias propiedades o métodos para ver las características de un DataFrame.

df.info() : Devuelve información (número de filas, número de columnas, índices, tipo de las columnas y memoria usado) sobre el DataFrame df.

df.shape : Devuelve una tupla con el número de filas y columnas del DataFrame df.

df.size : Devuelve el número de elementos del DataFrame.

df.columns : Devuelve una lista con los nombres de las columnas del DataFrame df.

df.index : Devuelve una lista con los nombres de las filas del DataFrame df.

df.dtypes : Devuelve una serie con los tipos de datos de las columnas del DataFrame df.

df.head(n) : Devuelve las n primeras filas del DataFrame df.

df.tail(n) : Devuelve las n últimas filas del DataFrame df.

```
>>> import pandas as pd
>>> df = pd.read_csv(
‘ruta del archivo’)

>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   nombre      14 non-null     object
 1   edad        14 non-null     int64
 2   sexo        14 non-null     object
 3   peso        13 non-null     float64
```

```
 4   altura      14 non-null     float64
 5   colesterol  13 non-null     float64
dtypes: float64(3), int64(1), object(2)
memory usage: 800.0+ bytes

>>> df.shape
(14, 6)
>>> df.size
84
>>> df.columns
Index(['nombre', 'edad', 'sexo', 'peso', 'altura', 'colesterol'], dtype='object')
>>> df.index
RangeIndex(start=0, stop=14, step=1)
>>> df.dtypes
nombre        object
edad           int64
sexo          object
peso         float64
altura       float64
colesterol    float64
dtype: object
```

Renombrar los nombres de las filas y columnas

Para cambiar el nombre de las filas y las columnas de un DataFrame se utiliza el siguiente método:

df.rename(columns=columnas, index=filas): Devuelve el DataFrame que resulta de renombrar las columnas indicadas en las claves del diccionario columnas con sus valores y las filas indicadas en las claves del diccionario filas con sus valores en el DataFrame df.

```
>>> import pandas as pd
>>> df = pd.read_csv(
‘ruta del archivo’)

>>> print(df.rename(columns={'nombre':'nombre y apellidos', 'altura':'estatura'},
index={0:1000, 1:1001, 2:1002}))
              nombre y apellidos  edad sexo    peso  estatura    colesterol
1000     José Luis Martínez Izquierdo    18    H    85.0      1.79        182.0
1001                Rosa Díaz Díaz    32    M    65.0      1.73        232.0
1002           Javier García Sánchez    24    H     NaN      1.81        191.0
3              Carmen López Pinzón    35    M    65.0      1.70        200.0
4             Marisa López Collado    46    M    51.0      1.58        148.0
...
```

Cambiar el índice de un DataFrame

Aunque el índice de un DataFrame suele fijarse en la creación del mismo, en ocasiones puede ser necesario cambiar el índice una vez creado el DataFrame. Para ello se utiliza el siguiente método:

df.set_index(keys = columnas, verify_integrity = bool): Devuelve el DataFrame que resulta de eliminar las columnas de la lista columnas y convertirlas en el nuevo índice. El parámetro verify_integrity recibe un booleano (False por defecto) y realiza una comprobación para evitar duplicados en la clave cuando recibe True.

```
>>> import pandas as pd
>>> df = pd.read_csv(
'ruta del archivo')
>>> print(df.set_index("nombre").head())
                        edad sexo  peso  altura  colesterol
nombre
José Luis Martínez Izquierdo    18    H  85.0    1.79       182.0
Rosa Díaz Díaz                  32    M  65.0    1.73       232.0
Javier García Sánchez           24    H   NaN    1.81       191.0
Carmen López Pinzón             35    M  65.0    1.70       200.0
Marisa López Collado            46    M  51.0    1.58       148.0
>>>
```