

# python para Ciencia de Datos: Hoja de Referencia



Aprende Python para Ciencia de Datos en www.datademia.es



### Pandas

La biblioteca Pandas está construida sobre NumPy y proporciona un uso fácil para estructuras de datos y herramientas de análisis de datos con Python.

### Usa la siguiente convención:

>>> import pandas as pd







### Estructuras de Datos Pandas

# Serie (Series)

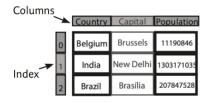
Una matriz unidimensional etiquetada capaz de contener cualquier tipo de dato.



>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])

### **DataFrame**

Una estructura de datos bidimensional etiquetada con columnas de tipos potencialmente diferentes



```
>>> df = pd.DataFrame(data,
           columns=['Country','Capital','Population'])
```

# Leer v escribir csv

```
>>> pd.read csv('archivo.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

# Leer y escribir archivos Excel

```
>>> pd.read excel('archivo.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

# Leer varias hojas del mismo archivo

```
>>> xlsx = pd.ExcelFile('archivo.xlsx')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

# Leer y escribir en consulta SQL o tabla de base de datos

```
>>> from sqlalchemy import create_engine
>>> engine = create engine('sglite:///:memorv:')
>>> engine - Create_engine( sqrite:///.memory. )
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
\verb|read_sql()| \textbf{ es un contenedor alrededor de } \verb|read_sql_table()| \textbf{ y} \verb|read_sql_query()|
>>> df.to_sql('myDf', engine)
```

>>> help(pd.Series.loc)

### Selección básica

Por posición

>>> df.iloc[[0],[0]]

```
>>> s['b']
                                                  Seleccionar un elemento
>>> df[1:]
                                                   Seleccionar parte de un DataFrame
 Country Capital Population
1 India New Delhi 1303171035
 2 Brazil Brasília 207847528
```

Selecciona un valor por fila y columna

Configura el índice a de la Serie s a 6

# Selección, Indexamiento Booleano y Configuración

'Belgium'	
>>> df.iat([0],[0])	
'Belgium'	
Por etiqueta	Selecciona un valor por etiquetas de fila
>>> df.loc[[0], ['Country']]	y columna
'Belgium'	y columna
>>> df.at([0], ['Country'])	
'Belgium'	
Por etiqueta/posición	
>>> df.ix[2]	Selecciona una sola fila del
Country Brazil	subconjunto de filas
Capital Brasília	, ,
Population 207847528	
>>> df.ix[:,'Capital']	Selecciona una sola columna del
0 Brussels	subconjunto de columnas
1 New Delhi	•
2 Brasília	
>>> df.ix[1,'Capital']	Selecciona filas y colunas
'New Delhi'	
Indexamiento Booleano	
>>> s[~(s > 1)]	Serie s donde el valor no es > 1
>>> s[(s < -1)   (s > 2)]	s donde el valor es < -1 o > 2
>>> df[df['Population']>1200000000]	Usa el filtro para ajustar el DataFrame
Configuración	

#### Descartar

>>> s['a'] = 6

>>> s.drop(['a', 'c'])	Descarta valores de filas (eje = 0)
>>> df.drop('Country', axis=1)	Descarta valores de columnas (eje = 1)

# Ordenar v Clasificar

>>> df.sort_index()	Ordenar por etiquetas por el eje
>>> df.sort_values(by='Country')	Ordenar por valores por el eje
>>> df.rank()	Asignar rangos a los valores

# Información Básica

>>> df.shape	(filas, columnas)
>>> df.index	Describe el índice
>>> df.columns	Describe las columnas del DataFrame
>>> df.info()	Información del DataFrame
>>> df.count()	Número de valores no NA

### Resumen

>>> df.sum()	Suma de valores
>>> df.cumsum()	Suma cumulativa de valores
>>> df.min()/df.max()	Valor mínimo/maximo
>>> df.idxmin()/df.idxmax()	Valor del índice mínimo/máximo
>>> df.describe()	Resumen de datos
>>> df.mean()	Media de valores
>>> df.median()	Mediana de valores

## **Aplicando funciones**

>>> f = lambda x: x*2	
>>> df.apply(f)	Aplica la función
>>> df.applymap(f)	Aplica la función por elemento

# Alineamiento de datos

## Alineamiento de datos interno

Los valores NA se introducen en los índices que no se superponen-

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
 a 10.0
 b NaN
 c 5.0
 d 7.0
```

## Operaciones aritméticas con métodos de relleno

También puedes hacer alineamiento interno de los datos tu mismo con la ayuda de los métodos de relleno:

```
>>> s.add(s3, fill_value = 0)
 a 10.0
 b -5.0
 d 7.0
>>> s.sub(s3, fill_value = 2)
>>> s.div(s3, fill_value = 4)
>>> s.mul(s3, fill_value = 3)
```



Aprende Python para Ciencia de Datos en www.datademia.es