



Módulo 2 – Obtención y Preparación de Datos

Librería Pandas

Ciencia de Datos

Contenidos



- Introducción
- Estructuras Básicas: Series y DataFrames
- Selección de un subset en un DataFrame
- Haciendo cálculos estadísticos
- Creando columnas en un DataFrame
- Creando una Serie o DataFrame desde cero
- Eliminar filas o columnas del DataFrame



KIBERNUM

Librería Pandas

1. Librería de software libre, construida sobre la base de NumPy.
2. Se utiliza para análisis, limpieza, preparación y visualización de datos.
3. Muy popular en proyectos de ciencia de datos.
4. Posee buen performance en su ejecución.
5. Permite alta productividad al desarrollar los algoritmos.
6. Permite trabajar con una amplia variedad de fuentes de datos.



Características

- Define nuevas estructuras de datos basadas en los **arrays de la librería NumPy**, pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente ficheros en **formato CSV, Excel y bases de datos SQL**.
- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para **reordenar, dividir y combinar conjuntos** de datos.
- Permite trabajar con **series temporales**.
- Realiza todas estas operaciones de manera muy eficiente.





About us ▾ Getting started Documentation Community ▾ Contribute

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

Getting started

- [Install pandas](#)
- [Getting started](#)

Documentation

- [User guide](#)
- [API reference](#)
- [Contributing to pandas](#)
- [Release notes](#)

With the support of:



<https://pandas.pydata.org/>

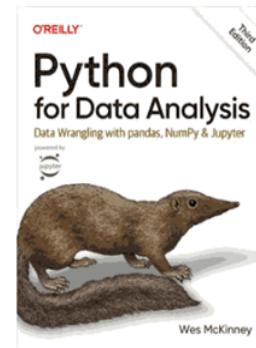
Latest version: 1.5.3

- [What's new in 1.5.3](#)
- Release date:
Jan 19, 2023
- [Documentation \(web\)](#)
- [Download source code](#)

Follow us



Get the book



Previous versions

- 1.5.2 (Nov 22, 2022)



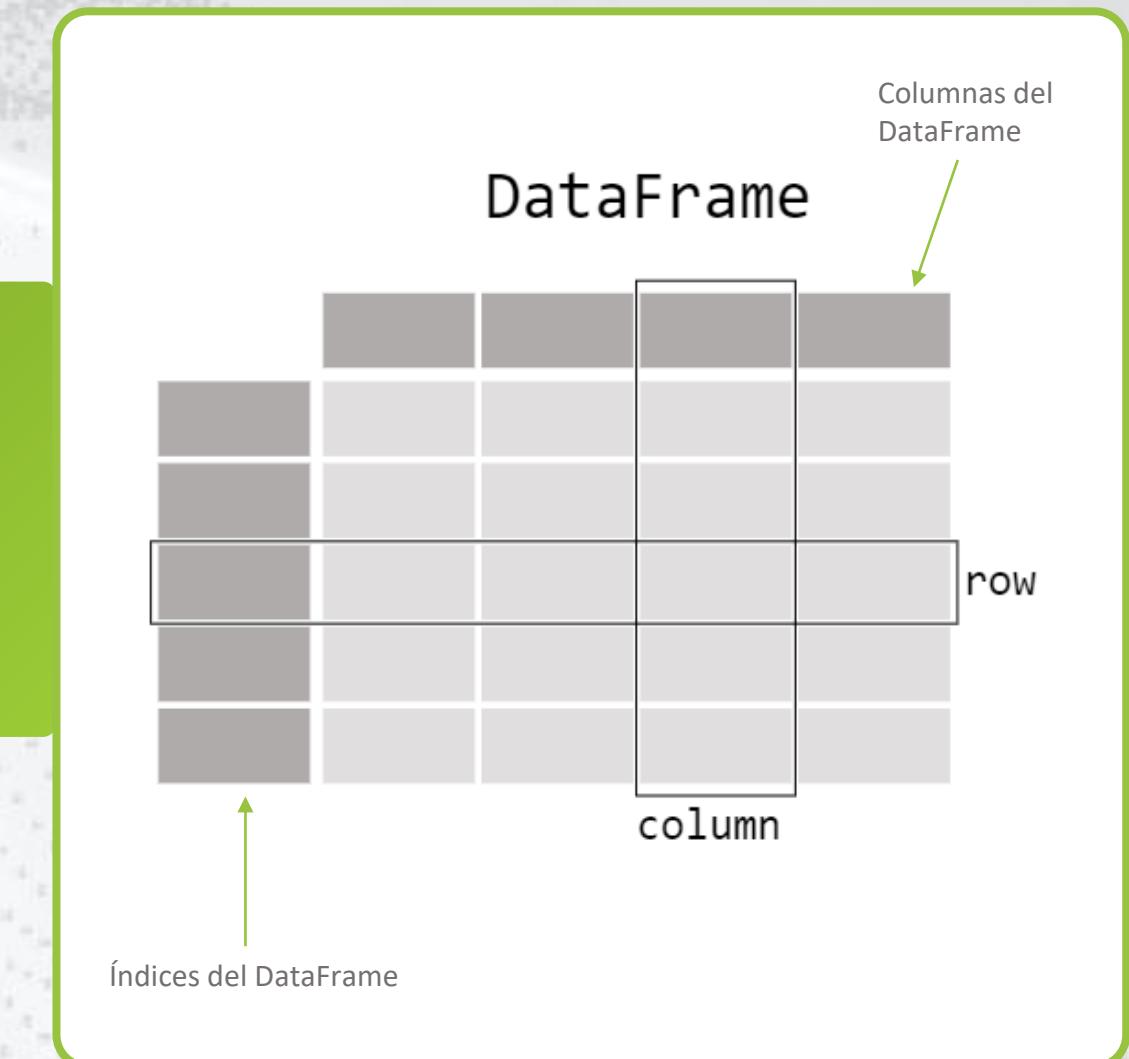
Estructuras Básicas: Series y DataFrames



KIBERNUM

DataFrame

- ➊ El **DataFrame** es un **objeto pandas** más común y una estructura de datos tabulares potencialmente heterogénea de tamaño mutable bidimensional con ejes etiquetados (filas y columnas).



Métodos de Exploración

El DataFrame tiene métodos que permiten una rápida exploración de la información:

El método head() muestra por defecto los primeros 5 elementos del DataFrame, pero se puede indicar el parámetro cantidad de registros.

```
: df.head(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nan	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	female	38.0	1	0	PC 17599	71.2833	C85	C

El método tail() muestra por defecto los último 5 elementos del DataFrame, pero se puede indicar el parámetro cantidad de registros.

```
: df.tail(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	Nan	Q

Métodos de Exploración

El método `describe()` permite obtener un sumario de estadísticas de las variables numéricas.

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Métodos de Exploración

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          -----          ----  
 0   PassengerId 891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64 
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare          891 non-null    float64 
 10  Cabin         204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

El método `info()` despliega un sumario técnico del DataFrame, en donde se listan las columnas, cantidad de registros no nulos y el tipo de datos de cada uno de ellos.

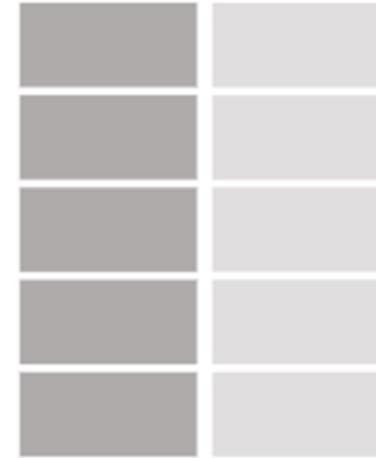


KIBERNUM

Series Pandas

- ➊ Son estructuras similares a los **arrays** de una dimensión. Son homogéneas, es decir, sus elementos tienen que ser del mismo tipo, y su tamaño es inmutable, es decir, no se puede cambiar, aunque sí su contenido.
- ➋ Dispone de un índice que asocia un nombre a cada elemento de la serie, a través de la cual se accede al elemento.

Series



↑
Índices de la Serie Valores de la Serie

Series Pandas

Ejemplo. La siguiente serie contiene las asignaturas de un curso.

Índice	→	A1	A2	A3	A4
Valores	→	Matemáticas	Economía	Programación	Inglés

Series Pandas

Ejemplo:

Los valores vienen en la columna Frecuencia y la columna Año corresponde al índice.

Año	Frecuencia (f)
1998	1066
1999	924
2000	1147
2001	945
2002	962
2003	895
2004	822
2005	717
2006	691



KIBERNUM

Seleccionando un Subset de un Dataframe



KIBERNUM

Seleccionando Columnas



Primero realizamos las importaciones.



KIBERNUM

Seleccionando Columnas

Como vimos anteriormente, para seleccionar una columna de un DataFrame debemos indicar de la siguiente manera. El resultado será una serie.

```
df['Name']
```

0	Braund, Mr. Owen Harris
1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina
2	Futrelle, Mrs. Jacques Heath (Lily May Peel)
3	Allen, Mr. William Henry
4	...
886	Montvila, Rev. Juozas
887	Graham, Miss. Margaret Edith
888	Johnston, Miss. Catherine Helen "Carrie"
889	Behr, Mr. Karl Howell
890	Dooley, Mr. Patrick

Name: Name, Length: 891, dtype: object

Si buscamos seleccionar varias columnas, debemos indicarlo mediante una lista con los nombres de las columnas que necesitamos. El resultado será otro DataFrame.

```
df[ ['Name', 'Age'] ]
```

	Name	Age
0	Braund, Mr. Owen Harris	22.0
1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	38.0
2	Futrelle, Mrs. Jacques Heath (Lily May Peel)	26.0
3	Allen, Mr. William Henry	35.0
4
886	Montvila, Rev. Juozas	27.0
887	Graham, Miss. Margaret Edith	19.0
888	Johnston, Miss. Catherine Helen "Carrie"	NaN
889	Behr, Mr. Karl Howell	26.0
890	Dooley, Mr. Patrick	32.0

Seleccionando Filas

Para seleccionar filas específicas de un DataFrame, se puede hacer de distintas formas. Una de ellas es indicando el índice que nos interesa rescatar.

En este caso, estamos rescatando en rango 5:8, es decir, los elementos 5, 6 y 7.

```
df.iloc[5:8]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S

En este caso, estamos indicando un listado con los índices específicos que deseamos rescatar

```
df.iloc[ [5,7,9] ]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Seleccionando Columnas



Podemos seleccionar celdas específicas de un dataframe indicando los índices y las columnas requeridas.



KIBERNUM

Seleccionando Celdas

Podemos seleccionar celdas específicas de un dataframe indicando los índices y las columnas requeridas.

```
df.loc[2:5, 'Name']
```

```
2          Heikkinen, Miss. Laina  
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)  
4          Allen, Mr. William Henry  
5          Moran, Mr. James  
Name: Name, dtype: object
```

En este caso se han seleccionado el rango de índices 2, 3 y 4 en la columna EmployeeName. El resultado en este caso es una Serie.

```
df.loc[2:5, ['Name', 'Age']]
```

	Name	Age
2	Heikkinen, Miss. Laina	26.0
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0
4	Allen, Mr. William Henry	35.0
5	Moran, Mr. James	NaN

En este caso, se han seleccionado dos columnas por lo tanto el resultado es un DataFrame.

Filtrando Filas del DataFrame

Se puede realizar una selección de datos condicional de datos, especificando la condición que deben cumplir las filas de la siguiente forma:

Condición de filtro

```
df[ df['Fare'] > 500 ]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
258	259	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	NaN	C
679	680	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	PC 17755	512.3292	B51 B53 B55	C
737	738	1	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	C

Filtrando Filas del DataFrame

Se puede hacer una selección de filas mediante expresiones con varias condiciones.

Los operadores son:
& : operador “and”
| : operador “or”

Las condiciones deben ser
encerradas entre paréntesis.

```
df[ (df['Fare'] > 500) & (df['Sex'] == 'female') ]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
258	259	1	1 Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	Nan	C



Realizando Cálculos Estadísticos



KIBERNUM

Valores Máximos y Mínimos

El método **max()** retorna el valor máximo de un dataframe (retorna una serie) o bien seleccionar el valor específico de una columna.

```
df.max()
```

```
PassengerId      100
Survived          1
Pclass             3
Name    Woolner, Mr. Hugh
Sex                male
Age                71
SibSp              5
Parch              5
Ticket           W.E.P. 5734
Fare                263
dtype: object
```

```
df['Fare'].max()
```

```
512.3292
```

El método **min()** retorna el valor mínimo de un dataframe (en forma de serie) o bien selecciona el valor específico de una columna.

```
df.min()
```

```
PassengerId      1
Survived          0
Pclass             1
Name    Abbing, Mr. Anthony
Sex                female
Age                0.42
SibSp              0
Parch              0
Ticket           110152
Fare                0
dtype: object
```

```
df['Fare'].min()
```

```
0.0
```

Contar los Valores

```
df.count()
```

```
PassengerId    100
Survived       100
Pclass          100
Name            100
Sex             100
Age             78
SibSp           100
Parch           100
Ticket          100
Fare            100
Cabin           20
Embarked        99
dtype: int64
```

El método **count()** retorna la cantidad de valores no nulos (NA) de un DataFrame.

```
df[ 'Fare' ].count()
```

```
891
```



KIBERNUM

Media y Mediana

El método **median()** retorna el valor que corresponde al 50% de la muestra.

```
df.median()
```

```
PassengerId    446.0000
Survived       0.0000
Pclass          3.0000
Age            28.0000
SibSp          0.0000
Parch          0.0000
Fare           14.4542
dtype: float64
```

```
df['Fare'].median()
```

```
15.675
```

El método **mean()** corresponde al promedio.

```
df.mean()
```

```
PassengerId    50.500000
Survived       0.410000
Pclass          2.400000
Age            27.465769
SibSp          0.730000
Parch          0.440000
Fare           29.517625
dtype: float64
```

```
df['Fare'].mean()
```

```
29.517625
```



KIBERNUM

Cuantil

El método **quantile()** retorna el valor que corresponde al **q%** de la muestra.

Cuantil (10%)

```
df.quantile(q=0.1)
```

PassengerId	10.90000
Survived	0.00000
Pclass	1.00000
Age	7.70000
SibSp	0.00000
Parch	0.00000
Fare	7.78375
Name:	0.1, dtype: float64

```
df['Fare'].quantile(q=0.1)
```

7.783749999999995

Cuantil (50%)

```
df.quantile(q=0.5)
```

PassengerId	50.500
Survived	0.000
Pclass	3.000
Age	26.000
SibSp	0.000
Parch	0.000
Fare	15.675
Name:	0.5, dtype: float64

```
df['Fare'].quantile(q=0.5)
```

15.675

Cuantil (90%)

```
df.quantile(q=0.9)
```

PassengerId	90.10000
Survived	1.00000
Pclass	3.00000
Age	46.90000
SibSp	3.00000
Parch	2.00000
Fare	62.11711
Name:	0.9, dtype: float64

```
df['Fare'].quantile(q=0.9)
```

62.11711000000001

Creación de Columnas en un DataFrame



KIBERNUM

Agregando Nuevas Columnas



A un dataframe podemos agregarle
columnas a conveniencia.

Columna con Valor Calculado

También, se puede definir una columna calculada en función de otras columnas.

```
df['Taxes'] = df['Fare'] * 0.05 + 1
```

```
df.head(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Taxes
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nan	S	1.362500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	female	38.0	1	0	PC 17599	71.2833	C85	C	4.564165

```
df['Total'] = df['Fare'] + df['Taxes']
```

```
df.head(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Taxes	Total
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nan	S	1.362500	8.612500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	female	38.0	1	0	PC 17599	71.2833	C85	C	4.564165	75.847465

Creación de Estructuras desde Cero



KIBERNUM

Creando una Serie

Para crear una serie desde cero, debemos proporcionar un listado con los datos y un listado con los índices.

```
Arreglo de  
datos  
↓  
mi_serie = pd.Series(data=[17.9, 43.8, 207.7, 31.77], index=['CL','AR','BR','PE'])  
Arreglo de  
etiquetas  
↓
```

```
mi_serie  
  
CL    17.90  
AR    43.80  
BR    207.70  
PE    31.77  
dtype: float64
```

```
type(mi_serie)
```

```
pandas.core.series.Series
```

Creación de Series

A partir de una lista:

- Sintaxis: Series(data=list, index=índices, dtype=tipo)

Devuelve un objeto de tipo Series. Si no se pasa la lista de índices se utilizan como índices los enteros del 0 al n-1, donde n es el tamaño de la serie. Si no se pasa el tipo de dato, éste se infiere.

```
import pandas as pd
lista= ['Matemáticas', 'Economía', 'Programación', 'Inglés']
s = pd.Series(data=lista, dtype='string')
print(s)

0    Matemáticas
1      Economía
2  Programación
3       Inglés
dtype: string
```

Creación de Series

Creación de una serie y asignación de índices:

```
lista= ['Matemáticas', 'Economía', 'Programación', 'Inglés']
índices= ['a', 'b', 'c', 'd']
s = pd.Series(data=lista, index=índices, dtype='string')
print(s)
```

```
a      Matemáticas
b      Economía
c  Programación
d      Inglés
dtype: string
```



KIBERNUM

Creación de Series

A partir de un diccionario. Sintaxis:

```
Series(data=diccionario, index=indices)
```

Devuelve un objeto de tipo Series. Si no se pasa la lista de índices, se utilizan como índices las claves del diccionario.

```
: 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
```

Creando un DataFrame

Creación de DataFrames:

- Desde listas.
- Desde diccionarios.
- Desde arreglos (arrays).
- Cargando conjuntos de datos desde repositorios (SQL Database, Excel, CSV and texto).

```
datos = [['Juan', 25], ['Pedro', 28], ['Miguel', 30]]  
columnas = ['Nombre', 'Edad']
```

```
mi_dataframe = pd.DataFrame(data=datos, columns=columnas)
```

```
mi_dataframe
```

	Nombre	Edad
0	Juan	25
1	Pedro	28
2	Miguel	30

```
type(mi_dataframe)
```

```
pandas.core.frame.DataFrame
```



KIBERNUM

Creación de DataFrame



Creando un DataFrame desde un array:

DataFrame(data=array, index=filas, columns=columnas, dtype=tipo)

Devuelve un objeto del tipo DataFrame - 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.

```
>>> 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 DataFrame

Creando un DataFrame desde un diccionario:

DataFrame(data=diccionario, index=filas, columns=columnas, dtype=tipos)

```
>>> 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',
... ]}
>>> 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 DataFrame

```
>>> 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',
... ]
>>> 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
```

Devuelve un objeto del tipo DataFrame. 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.

Eliminación de Filas y Columnas



KIBERNUM

Agregar o Eliminar Columnas

Eliminar una columna del DataFrame:

Nombre de columna Axis=1 columna
 Axis=0 fila

Inplace = true modifica el dataframe original
Inplace=false solo devuelve una copia

```
df.drop('Total', axis=1, inplace=True)
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Taxes
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nan	S	1.362500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599	71.2833	C85	C	4.564165
2	3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	STON/O2. 3101282	7.9250	Nan	S	1.396250
3	4	1	1	Allen, Mr. William Henry	male	35.0	0	0	113803	53.1000	C123	S	3.655000
4	5	0	3						373450	8.0500	Nan	S	1.402500

Agregar o Eliminar Columnas

Eliminar una fila del DataFrame:

Índice de Fila Axis=1 columna
 Axis=0 fila Inplace = true modifica el dataframe original
 Inplace=false solo devuelve una copia

```
df.drop(0, axis=0, inplace=True)
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Taxes
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	38.0	1	0	PC 17599 STON/O2. 3101282	71.2833 7.9250 53.1000	C85 NaN C123	C	4.564165 1.396250 3.655000
2	3	1	3	Allen, Mr. William Henry Moran, Mr. James	male	35.0	0	0	373450 330877	8.0500 8.4583	NaN NaN	S	1.402500 1.422915
3	4	1	1										
4	5	0	3										
5	6	0	3										

Dudas y consultas



KIBERNUM



Fin Presentación



KIBERNUM