## Abrir archivos con pandas

nombre edad ciudad  
0 Juan 28 Madrid  
1 Ana 22 Barcelona

para saltar tabulaciones agrega el parametro delimiter al pd.read asi:  
 pd.read\_csv(source, delimiter='\t')

Para archivos con comentarios con el signo #,   
 se debe saltar las filas con el parametro skiprows = n

nombre edad ciudad  
0 Juan 28 Madrid  
1 Ana 22 Barcelona  
2 Luis 35 Valencia

Dependiendo el caso agregar el parametro: engine='python'

## Descripcion preliminar de los datos

Para tener una descripcion de los datos utlizar el metodo describe de pandas

edad  
count 14.000000  
mean 25.000000  
std 3.113247  
min 22.000000  
25% 22.000000  
50% 25.000000  
75% 28.000000  
max 28.000000

Si quiero saber los tipos utilizo la propiedad dtypes:

nombre object  
edad int64  
ciudad object

Si quero obtener los valores unicos de una columna utilizo el metodo unique, e  
 specificando la columna, ejemplo: df["nombre"].unique():

['Juan' 'Ana' 'Pablo' 'Carlos' 'Maria' 'Carla' 'Marcos' 'Jhon' 'Camila'  
 'Enzo' 'Julian' 'Tita' 'Pepe']

Si quiero contarlos utilizo el metodo value\_counts:

nombre  
Juan 2  
Ana 1  
Pablo 1  
Carlos 1  
Maria 1  
Carla 1  
Marcos 1  
Jhon 1  
Camila 1  
Enzo 1  
Julian 1  
Tita 1  
Pepe 1

## Manejo de datos faltantes

Para contar la cantidad de datos faltantes por columna   
 utilizar los metodos .isnull().sum():

ID 0  
Nombre 1  
Edad 2  
Ciudad 1  
Salario 2

Para eliminar los datos faltantes se utiliza el metodo .dropna()

ID Nombre Edad Ciudad Salario  
0 1 Juan 30.0 Madrid 3000.0  
1 2 Ana 22.0 Barcelona 3200.0  
2 3 Luis 35.0 Valencia 3500.0  
3 4 María 29.0 Sevilla 4000.0  
6 7 Pedro 20.0 Valencia 3700.0

Si quiero eliminar las columnas con datos faltantes se utiliza el parametro axis=1  
 en el metodo .dropna()

ID  
0 1  
1 2  
2 3  
3 4  
4 5  
5 6  
6 7  
7 8  
8 9  
9 10

Se puede rellenar los datos faltantes utilizando el metodo fillna()  
 y especificar que deseas rellenar

ID Nombre Edad Ciudad Salario  
0 1 Juan 30.0 Madrid 3000.0  
1 2 Ana 22.0 Barcelona 3200.0  
2 3 Luis 35.0 Valencia 3500.0  
3 4 María 29.0 Sevilla 4000.0  
4 5 Carlos NaN Madrid 3200.0  
5 6 Laura 30.0 Madrid NaN  
6 7 Pedro 20.0 Valencia 3700.0  
7 8 Desconocido 22.0 Valencia 3400.0  
8 9 Lucía NaN NaN 3600.0  
9 10 Hugo 25.0 Sevilla NaN

Para rellenar datos numericos como el salario se puede rellenar con la media

ID Nombre Edad Ciudad Salario  
0 1 Juan 30.0 Madrid 3000.0  
1 2 Ana 22.0 Barcelona 3200.0  
2 3 Luis 35.0 Valencia 3500.0  
3 4 María 29.0 Sevilla 4000.0  
4 5 Carlos NaN Madrid 3200.0  
5 6 Laura 30.0 Madrid 3450.0  
6 7 Pedro 20.0 Valencia 3700.0  
7 8 Desconocido 22.0 Valencia 3400.0  
8 9 Lucía NaN NaN 3600.0  
9 10 Hugo 25.0 Sevilla 3450.0

Para datos numericos como la edad, es mas util la mediana:

ID Nombre Edad Ciudad Salario  
0 1 Juan 30.0 Madrid 3000.0  
1 2 Ana 22.0 Barcelona 3200.0  
2 3 Luis 35.0 Valencia 3500.0  
3 4 María 29.0 Sevilla 4000.0  
4 5 Carlos 27.0 Madrid 3200.0  
5 6 Laura 30.0 Madrid 3450.0  
6 7 Pedro 20.0 Valencia 3700.0  
7 8 Desconocido 22.0 Valencia 3400.0  
8 9 Lucía 27.0 NaN 3600.0  
9 10 Hugo 25.0 Sevilla 3450.0

Para rellenar datos como la cuidad, seria util el uso de la moda

ID Nombre Edad Ciudad Salario  
0 1 Juan 30.0 Madrid 3000.0  
1 2 Ana 22.0 Barcelona 3200.0  
2 3 Luis 35.0 Valencia 3500.0  
3 4 María 29.0 Sevilla 4000.0  
4 5 Carlos 27.0 Madrid 3200.0  
5 6 Laura 30.0 Madrid 3450.0  
6 7 Pedro 20.0 Valencia 3700.0  
7 8 Desconocido 22.0 Valencia 3400.0  
8 9 Lucía 27.0 Madrid 3600.0  
9 10 Hugo 25.0 Sevilla 3450.0

Para especificar los datos en donde se deban eliminar las columnas  
 se utiliza la propiedad subset, ejemplo de subset= [Nombre]:

ID Nombre Edad Ciudad Salario  
0 1 Juan 30.0 Madrid 3000.0  
1 2 Ana 22.0 Barcelona 3200.0  
2 3 Luis 35.0 Valencia 3500.0  
3 4 María 29.0 Sevilla 4000.0  
4 5 Carlos NaN Madrid 3200.0  
5 6 Laura 30.0 Madrid NaN  
6 7 Pedro 20.0 Valencia 3700.0  
8 9 Lucía NaN NaN 3600.0  
9 10 Hugo 25.0 Sevilla NaN

## Reestructurar datos

Para segmentar y clasificar datos continuos en grupos o intevalos se utiliza pd.cut():

ID Nombre Edad Ciudad Salario Rango\_Edad  
0 1 Juan 30.0 Madrid 3000.0 25-30  
1 2 Ana 22.0 Barcelona 3200.0 20-25  
2 3 Luis 35.0 Valencia 3500.0 30-35  
3 4 María 29.0 Sevilla 4000.0 25-30  
4 5 Carlos 27.0 Madrid 3200.0 25-30  
5 6 Laura 30.0 Madrid 3450.0 25-30  
6 7 Pedro 20.0 Valencia 3700.0 20-25  
7 8 Desconocido 22.0 Valencia 3400.0 20-25  
8 9 Lucía 27.0 Madrid 3600.0 25-30  
9 10 Hugo 25.0 Sevilla 3450.0 20-25

Para obtener el salario promedio por edad utilizo groupby()

Rango\_Edad  
20-25 3437.5  
25-30 3450.0  
30-35 3500.0

## Manejo de duplicados

Al usar el metodo .duplicated() me indica por medio de booleans que filas son duplicadas, sin necesidad de que estos sean consecutivos

0 False  
1 False  
2 False  
3 False  
4 False  
5 True  
6 False  
7 False  
8 False  
9 False  
10 True

Se puede usar el parametro subset para especificar las columnas y saber si contiene duplicados

0 False  
1 False  
2 False  
3 False  
4 False  
5 True  
6 False  
7 False  
8 False  
9 True  
10 True

Obteniendo las filas de los duplicados con: duplicate[duplicate.duplicated()

ID Nombre Edad Salario Fecha\_Ingreso  
5 3 Carlos 30 58000 2020-03-25  
10 2 Luis 35 60000 2019-06-15

Para eliminar los duplicados se utiliza el metodo .drop\_duplicates()

ID Nombre Edad Salario Fecha\_Ingreso  
0 1 Ana 23 50000 2020-01-10  
1 2 Luis 35 60000 2019-06-15  
2 3 Carlos 30 58000 2020-03-25  
3 4 María 22 58000 2018-07-30  
4 5 Elena 28 49000 2018-07-30  
6 7 Sofía 34 72000 2016-11-22  
7 8 Andrés 45 61000 2021-04-01  
8 9 Laura 29 52000 2015-05-19  
9 10 Ana 41 67000 2014-08-25

Para crear una columna que indique los valores duplicados se hace de este metodo: duplicate['Es duplicado?'] = duplicate.duplicated()

ID Nombre Edad Salario Fecha\_Ingreso Es duplicado?  
0 1 Ana 23 50000 2020-01-10 False  
1 2 Luis 35 60000 2019-06-15 False  
2 3 Carlos 30 58000 2020-03-25 False  
3 4 María 22 58000 2018-07-30 False  
4 5 Elena 28 49000 2018-07-30 False  
5 3 Carlos 30 58000 2020-03-25 True  
6 7 Sofía 34 72000 2016-11-22 False  
7 8 Andrés 45 61000 2021-04-01 False  
8 9 Laura 29 52000 2015-05-19 False  
9 10 Ana 41 67000 2014-08-25 False  
10 2 Luis 35 60000 2019-06-15 True

Usando .map puedo cambiar los valores de la columna 'Es duplicado' de un booleano a Si 0 No

ID Nombre Edad Salario Fecha\_Ingreso Es duplicado?  
0 1 Ana 23 50000 2020-01-10 No  
1 2 Luis 35 60000 2019-06-15 No  
2 3 Carlos 30 58000 2020-03-25 No  
3 4 María 22 58000 2018-07-30 No  
4 5 Elena 28 49000 2018-07-30 No  
5 3 Carlos 30 58000 2020-03-25 Si  
6 7 Sofía 34 72000 2016-11-22 No  
7 8 Andrés 45 61000 2021-04-01 No  
8 9 Laura 29 52000 2015-05-19 No  
9 10 Ana 41 67000 2014-08-25 No  
10 2 Luis 35 60000 2019-06-15 Si

Para reordenar las columnas de un dataframe se hace de esta forma:duplicate[['ID','Nombre','Edad', 'Salario','Es duplicado?','Fecha\_Ingreso']]

ID Nombre Edad Salario Es duplicado? Fecha\_Ingreso  
0 1 Ana 23 50000 No 2020-01-10  
1 2 Luis 35 60000 No 2019-06-15  
2 3 Carlos 30 58000 No 2020-03-25  
3 4 María 22 58000 No 2018-07-30  
4 5 Elena 28 49000 No 2018-07-30  
5 3 Carlos 30 58000 Si 2020-03-25  
6 7 Sofía 34 72000 No 2016-11-22  
7 8 Andrés 45 61000 No 2021-04-01  
8 9 Laura 29 52000 No 2015-05-19  
9 10 Ana 41 67000 No 2014-08-25  
10 2 Luis 35 60000 Si 2019-06-15

Para elegir columnas especificas de un dataframe se puede usa la propiedad .loc de esta manera: duplicate.loc[:,['ID','Nombre']]

ID Nombre  
0 1 Ana  
1 2 Luis  
2 3 Carlos  
3 4 María  
4 5 Elena  
5 3 Carlos  
6 7 Sofía  
7 8 Andrés  
8 9 Laura  
9 10 Ana  
10 2 Luis

El primer parametro de la propiedad .loc se utiliza para especificar el rango de las filas que se selecionaran, ejemplo de filas del 1 al 9 / 1:9

ID Nombre  
1 2 Luis  
2 3 Carlos  
3 4 María  
4 5 Elena  
5 3 Carlos  
6 7 Sofía  
7 8 Andrés  
8 9 Laura

Para eliminar una columna, ejemplo la duplicado, se utiliza el metodo ..drop(columns=['Es duplicado?'])

ID Nombre Edad Salario Fecha\_Ingreso  
0 1 Ana 23 50000 2020-01-10  
1 2 Luis 35 60000 2019-06-15  
2 3 Carlos 30 58000 2020-03-25  
3 4 María 22 58000 2018-07-30  
4 5 Elena 28 49000 2018-07-30  
5 3 Carlos 30 58000 2020-03-25  
6 7 Sofía 34 72000 2016-11-22  
7 8 Andrés 45 61000 2021-04-01  
8 9 Laura 29 52000 2015-05-19  
9 10 Ana 41 67000 2014-08-25  
10 2 Luis 35 60000 2019-06-15

Si quiero filtra los salarios superiores a 50,000 se hace de la siguiente manera: duplicate.loc[duplicate['Salario' ]> 50000]

ID Nombre Edad Salario Fecha\_Ingreso  
1 2 Luis 35 60000 2019-06-15  
2 3 Carlos 30 58000 2020-03-25  
3 4 María 22 58000 2018-07-30  
5 3 Carlos 30 58000 2020-03-25  
6 7 Sofía 34 72000 2016-11-22  
7 8 Andrés 45 61000 2021-04-01  
8 9 Laura 29 52000 2015-05-19  
9 10 Ana 41 67000 2014-08-25  
10 2 Luis 35 60000 2019-06-15

Para agregar una columna nueva a un dataframe se hace de esta manera:duplicate['Posición'] agregando los valores que tendra la columna:

ID Nombre Edad Salario Fecha\_Ingreso Posición  
0 1 Ana 23 50000 2020-01-10 junior  
1 2 Luis 35 60000 2019-06-15 junior  
2 3 Carlos 30 58000 2020-03-25 junior  
3 4 María 22 58000 2018-07-30 junior  
4 5 Elena 28 49000 2018-07-30 junior  
5 3 Carlos 30 58000 2020-03-25 junior  
6 7 Sofía 34 72000 2016-11-22 Senior  
7 8 Andrés 45 61000 2021-04-01 mid  
8 9 Laura 29 52000 2015-05-19 junior  
9 10 Ana 41 67000 2014-08-25 Senior  
10 2 Luis 35 60000 2019-06-15 junior

Esta nueva columna se calcula cuanto se le descuenta de afp + ars:

ID Nombre Edad Salario Fecha\_Ingreso Posición AFP + ARS Salario\_Neto  
0 1 Ana 23 50000 2020-01-10 junior 2955.0 47045.0  
1 2 Luis 35 60000 2019-06-15 junior 3546.0 56454.0  
2 3 Carlos 30 58000 2020-03-25 junior 3427.8 54572.2  
3 4 María 22 58000 2018-07-30 junior 3427.8 54572.2  
4 5 Elena 28 49000 2018-07-30 junior 2895.9 46104.1  
5 3 Carlos 30 58000 2020-03-25 junior 3427.8 54572.2  
6 7 Sofía 34 72000 2016-11-22 Senior 4255.2 67744.8  
7 8 Andrés 45 61000 2021-04-01 mid 3605.1 57394.9  
8 9 Laura 29 52000 2015-05-19 junior 3073.2 48926.8  
9 10 Ana 41 67000 2014-08-25 Senior 3959.7 63040.3  
10 2 Luis 35 60000 2019-06-15 junior 3546.0 56454.0

## Concatenacion y combinacion

Para concatenar dos o mas dataframes se utiliza el metodo concat de esta manera: pd.concat([df\_lima, df\_bogota]), este es el resultado:

Producto Ventas Ciudad  
0 A 250 Lima  
1 B 150 Lima  
0 A 100 Bogotá  
1 C 200 Bogotá

Al concatenar datframes es recomendable reiniciar indices usando la propiedad: reset\_index(drop=True)

Producto Ventas Ciudad Producto Inventario  
0 A 250 Lima A 30.0  
1 B 150 Lima B 45.0  
2 A 100 Bogotá NaN NaN  
3 C 200 Bogotá NaN NaN