Python para el análisis de datos- Lectura 6

Ing. Pedro Rotta

Universidad de Piura - Vida Universitaria

Enero-2022

Pandas

Pandas es otra librería fundamental en python y que tiene muchos métodos para trabajar con archivos de excel y archivos .csv(valores separados por comas).

Su instalación se puede realizar similarmente con :

```
pip install pandas
```

Donde el comando se introduce en CMD.

Pandas

Pandas es otra librería fundamental en python y que tiene muchos métodos para trabajar con archivos de excel y archivos .csv(valores separados por comas).

Su instalación se puede realizar similarmente con :

```
pip install pandas
```

Donde el comando se introduce en CMD.

Para importar la librería escribimos:

```
import pandas as pd
```

DataFrame

El objeto principal de pandas es el DataFrame. El dataframe es una colección de datos de doble entrada que permite ordenar a las variables en filas y columnas. Es en términos prácticos una tabla digital.

DataFrame

El objeto principal de pandas es el DataFrame. El dataframe es una colección de datos de doble entrada que permite ordenar a las variables en filas y columnas. Es en términos prácticos una tabla digital.

Un DataFrame se puede crear a partir de colecciones primarias como diccionarios o listas y al mismo tiempo, se puede crear a partir de la importación de archivos con datos como archivos CSV(Valores separados por coma) y archivos excel(xlsx).

Sintaxis de creación de un DataFrame

Para crear un DataFrame podemos crearlo por medio de un diccionario
Sintaxis:

```
df = pd.DataFrame(df_dicc)
```

Donde cada columna es el nombre del diccionario y los items se guardan en listas.

También otra forma de crear un dataframe es por medio de una lista. Como sigue la siguiente sintaxis:

```
df_list = [[items_fila1],[items_fila2],...]
colums = [column1,column2,column3]
df = pd.DataFrame(df_list, columns)
```

Ver problema 1.

Sintaxis de archivos

Para abrir archivos de excel:

```
import pandas as pd
archivo = 'archivo.xlsx'
df = pd.read_excel(archivo)
```

Para abrir archivos de csv:

```
import pandas as pd
archivo = 'archivo.csv'
df = pd.read_csv(archivo)
```

Ver problema 2

Métodos para archivos en pandas

Para añadir una fila, el comando que usaremos es **append**.
La sintaxis en pandas para añadir una fila a una tabla es:

```
df.append({fila},ignore_index = True)
```

Como sigue el problema 3.

Métodos para archivos en pandas

Para añadir una fila, el comando que usaremos es **append**.
La sintaxis en pandas para añadir una fila a una tabla es:

```
df.append({fila},ignore_index = True)
```

Como sigue el problema 3.

Para obtener las columnas:

```
columnas = df.columns()
```

Ver problema 4.

Métodos para archivos en pandas

Para añadir una fila, el comando que usaremos es **append**.
La sintaxis en pandas para añadir una fila a una tabla es:

```
df.append({fila},ignore_index = True)
```

Como sigue el problema 3.

Para obtener las columnas:

```
columnas = df.columns()
```

Ver problema 4.

En el problema 5 vemos un programa con estos comandos.

Head Tail

Para mostrar los primeros n elementos de una tabla, se usa el método que pertenece al dataframe **head(n)**. Se muestra la sintaxis:

```
df.head(n)
```

Para mostrar los primeros n elementos de una tabla, se usa el método que pertenece al dataframe **tail(n)**. Se muestra la sintaxis:

```
df.tail(n)
```

Podemos ver el siguiente ejemplo en el problema 6.

Indexación

Para nosotros un dataframe será sinónimo de tabla.

Un dataframe tiene indexación por columna y por fila. Para llamar a todas las columnas aplicamos el método `columns()`. Para llamar a una columna, sabiendo su nombre, aplicamos la sintaxis siguiente:

```
columna_name = df['name']
```

Indexación

Luego los elementos de la columna tendran una indexación de elementos similar a las listas, empezando por el índice 0. Por ejemplo, si queremos llamar a los primeros 5 elementos de la columna guardada. Podemos escribir la sintaxis:

```
columna_items = columna_name[0:5]
```

Donde 5 define hasta el 5to elemento, o del 0 al 4.

Esto es lo mismo a escribir:

```
columna_items = df['name'][0:5]
```

Indexación

Luego para indexar por filas, usamos el método iloc, que también pertence al objeto del dataframe. Su sintaxis es:

```
fila_n = df.iloc[n]
```

Donde n hace referencia al valor de la fila que llamo.

En caso quiero llamar a varias filas, uso la indexación clásica, donde 0 es el primer valor, como:

```
fila_n = df.iloc[0:n]
```

Estos comandos se exploran en el problema 7.

Loc y describe

El comando Loc sirve para filtrar valores de un dataframe de manera que se puedan obtener datos según se necesiten. La sintaxis es:

```
filtro = df.loc[prop_log]
```

Donde dentro del corchete van las proposiciones lógicas que hacen que se cumpa el filtro.

Loc y describe

El comando Loc sirve para filtrar valores de un dataframe de manera que se puedan obtener datos según se necesiten. La sintaxis es:

```
filtro = df.loc[prop_log]
```

Donde dentro del corchete van las proposiciones lógicas que hacen que se cumpa el filtro.

El método describe que pertenece al objeto dataframe, sirve para proporcionar algunos datos estadísticos simples para el conjunto de datos.

La sintaxis es :

```
df.describe()
```

Ver problema 8.

Nuevas columnas y ordenamiento

Para añadir nuevas columnas a un dataframe, solo basta con llamar a esta nueva columna como una indexación del dataframe. Como se muestra en la siguiente sintaxis:

```
df['col_name'] = Relación
```

También se puede seleccionar, la posibilidad de ordenar una columna con el comando **sort_values**. Se usa la siguiente sintaxis:

```
df_ordenado = df.sort_values('Col_elegida',
ascending = True)
```

Donde el True significa que está ordenado ascendentemente. Veremos esto aplicado en el problema 9.

Guardar archivos de dataframes

Para guardar un archivo de excel en un dataframe aplicamos la sintaxis siguiente:

```
df.to_excel("path_to_file.xlsx")
```

Veremos una aplicación en el problema 10.