Pandas

Pandas (acrónimo para Panel Data) es un módulo orientado a la manipulación y limpieza de estructuras de datos.

Se suele utilizar en conjunto a otros módulos como numpy , scipy y matplotlib para el análisis de datos.

¿Por qué pandas ?

1. Es una colección de funciones y algoritmos que implementan las principales convenciones sobre el análisis de datos.
2. Permite importar distintos archivos de datos con procedimientos robustos permiten centrar al investigador en lo substancial y no en procesar archivos.
3. Genera un objeto DataFrame con una estructura de matriz (Filas y Columnas) que resulta intuitiva para desarrollar el análisis orientado en variables y segmentado por casos.
4. Presenta una amplia gama de funciones y procedimientos comunes generadas en los objetos.

Importando archivos con pandas

Usualmente trabajaremos con datos en un formato de texto plano, llámese csv o xlsx .

Ahora generaremos nuestra primera interacción con pandas :

- La convención indica que pandas se abrevia de forma pd .

In [1]:
```
import pandas as pd
```

- Dentro de nuestro directorio de trabajo encontraremos el archivo alumnos.csv .
- Un archivo csv (comma separated value) presenta el nombre de los atributos en la primera fila. Después de ella, le siguen las observaciones ingresadas.
- Cada observación está separada mediante comas (de ahí el nombre archivo separado por comas).

In [2]:
```
# solicitemos las primeras filas del archivo alumnos (desde bash)
!head alumnos.csv
```

```
nombre,altura,peso,edad,sexo
Hugo,1.67,60,23,h
Paco,1.73,83,25,h
Luis,1.62,70,28,h
Diana,1.58,58,21,m
Francisco,1.86,98,28,h
Felipe,1.79,100,26,h
Jacinta,1.69,62,20,m
Bernardo,1.6,83,31,h
Marisol,1.6,56,30,m
```

Para ingresar éste archivo a nuestro notebook, utilizaremos la función read_csv de pandas.

Digresión: ¿Cómo llamar funciones dentro de un módulo?

Para llamar una función y/u objeto específico dentro de nuestro módulo importado, utilizamos la siguiente sintáxis:

$$modulo.funcion$$

Vamos a generar un objeto llamado df (contracción de dataframe) mediante read_csv

In [3]:
```
df = pd.read_csv('alumnos.csv')
```

Si todo resulta bien, podemos ver las primeras 5 observaciones de nuestro nuevo objeto con df.head().

In [4]:
```
df.head()
```

Out[4]:

	nombre	altura	peso	edad	sexo
0	Hugo	1.67	60	23	h
1	Paco	1.73	83	25	h
2	Luis	1.62	70	28	h
3	Diana	1.58	58	21	m
4	Francisco	1.86	98	28	h

Los resultados son idénticos que el archivo alumnos.csv , con la salvedad que están mejor presentados.

El resultado de head es la representación en filas y columnas.

Hay algunas salvedades a destacar:

1. En Python, los índices comienzan en 0.
2. La primera columna de nuestra tabla corresponde a la posición de la fila respecto al DataFrame. Esta información nos facilitará segmentación de archivos.

DataFrame

El objeto df que creamos recientemente es un objeto DataFrame, una de las estructuras elementales de pandas .

Un objeto DataFrame representa una tabla rectangular de datos compuestas por filas (observaciones registradas en el archivo) y una serie de columnas (atributos medibles que pueden ser integer, float, string, boolean, etc...).

Las observaciones registradas son insertadas en bloques bidimensionales que responden a la notación de matrices.

Podemos inspeccionar las dimensiones de la tabla mediante .shape. Éste nos informará de la cantidad de filas y columnas.

```
In [5]: df.shape
```

```
Out[5]: (21, 5)
```

Esta tabla es manipulable y segmentable. Imaginemos que ahora desamos extraer sólo las primeras 3 observaciones de nuestra tabla. La operación se realiza de la siguiente manera

```
In [7]: df[:3]
```

Out[7]:

	nombre	altura	peso	edad	sexo
0	Hugo	1.67	60	23	h
1	Paco	1.73	83	25	h
2	Luis	1.62	70	28	h

Python interpretó esta instrucción como "dentro de la tabla df , muéstrame las observaciones hasta la 5".

Eso se generó dentro de los brackets [] , donde pasamos un operador: que se llama slice y permite instruír hasta dónde se puede cortar un elemento.

En el caso anterior utilizamos slice para generar una submuestra hasta cierta condición (que se evalúa por el índice de la tabla; la primera columna).

¿Qué pasa si deseamos generar una segmentación desde un valor en específico? Utilizamos la siguiente sintáxis:

In [8]:
```
df[13:]
```

Out[8]:

	nombre	altura	peso	edad	sexo
13	Diego	1.62	78	23	h
14	Gonzalo	1.58	67	22	h
15	Alejandra	1.86	74	21	m
16	Fernando	1.79	93	27	h
17	Carolina	1.60	63	28	m
18	Vicente	1.98	102	31	h
19	Benjamín	1.72	78	36	h
20	Gloria	1.58	65	23	m

Acá instruímos a la tabla que entregue los resultados desde la fila 13 hasta el final de las observaciones.

¿Y si queremos seleccionar entre dos valores? Utilizamos la siguiente sintáxis:

In [9]:
```
df[3:13]
```

Out[9]:

	nombre	altura	peso	edad	sexo
3	Diana	1.58	58	21	m
4	Francisco	1.86	98	28	h
5	Felipe	1.79	100	26	h
6	Jacinta	1.69	62	20	m
7	Bernardo	1.60	83	31	h
8	Marisol	1.60	56	30	m
9	Facundo	1.98	112	36	h
10	Trinidad	1.72	72	21	m
11	Camila	1.63	57	26	m
12	Macarena	1.73	68	27	m

Series

Las segmentaciones realizadas anteriormiente fueron orientadas a las filas de una tabla. Esto también se puede realizar a las columnas de la tabla.

Para ello utilizamos una forma similar. Lo que vamos a separar la columna peso.

In [12]:
```
df['peso']
```

Out[12]:
```
0      60
1      83
2      70
3      58
4      98
5     100
6      62
7      83
8      56
9     112
10     72
11     57
12     68
13     78
14     67
15     74
16     93
17     63
18    102
19     78
20     65
Name: peso, dtype: int64
```

Entre los brackets pasamos el nombre exacto de la columna que deseamos analizar. Ya que trabajaremos con ésta columna, guardémosla en un nuevo objeto

In [13]:
```
peso = df['peso']
type(peso)
```

Out[13]:
```
pandas.core.series.Series
```

Cuando separamos ésta columna y preguntamos por su tipo, Python nos entrega que es un objeto pandas.core.series.Series , este elemento que separamos se conoce como Series.

En pandas , las series son listas unidimensionales que contienen una secuencia de valores.

Todo objeto pd.Series tiene asociado una lista de etiquetas de datos denominada index. De manera similar a su comportamiento en DataFrame , nos permite realizar segmentaciones.

In [14]:
```
peso[15:]
```

Out[14]:
```
15     74
16     93
17     63
18    102
19     78
20     65
Name: peso, dtype: int64
```

In [ ]:
```

```