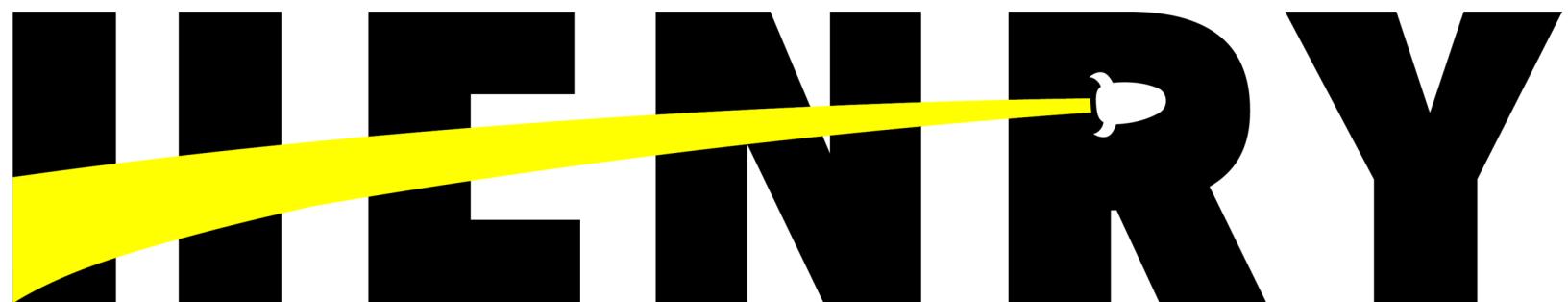
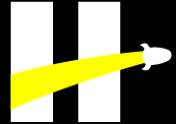


HENRY

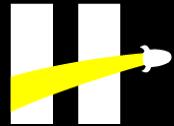
A thick, yellow, curved line starts from the bottom left, passing through the vertical stroke of the 'H', then curving upwards and to the right, ending at the top of the 'Y'. A small white rocket ship is positioned at the end of the line, pointing towards the upper right.

Pandas



Pandas

- Estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente archivos 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.



Pandas

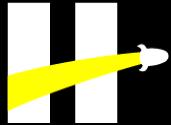
Pandas dispone de estructuras de datos diferentes:

- Series: Estructura de una dimensión.
- DataFrame: Estructura de dos dimensiones (tablas)

Series		Series		DataFrame	
	apples		oranges		
0	3	0	0	0	0
1	2	1	3	1	3
2	0	2	7	2	7
3	1	3	2	3	2

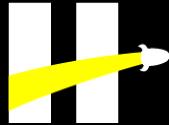
+

=



Series

- Estructuras similares a los arrays de una dimensión.
- Sus elementos tienen que ser del mismo tipo, y su tamaño es inmutable, es decir, no se puede cambiar, aunque si su contenido.
- Dispone de un índice que asocia un nombre a cada elemento del la serie, a través de la cuál se accede al elemento.



Crear Series

A partir de una lista:

```
>>> import pandas as pd  
>>> s = pd.Series(['Matemáticas', 'Historia', 'Economía', 'Programación', 'Inglés'], dtype='string')  
>>> print(s)  
0    Matemáticas  
1        Historia  
2        Economía  
3   Programación  
4        Inglés  
dtype: string
```

A partir de un diccionario:

```
>>> import pandas as pd  
>>> 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
```



Atributos de una Serie

```
>>> import pandas as pd  
>>> s = pd.Series([1, 2, 2, 3, 3, 3, 4, 4, 4, 4])  
>>> s.size  
10  
>>> s.index  
RangeIndex(start=0, stop=10, step=1)  
>>> s.dtype  
dtype('int64')
```

II → Acceso a elementos de una Serie

```
>>> s[1:3]
Economía      4.5
Programación   8.5
dtype: float64
>>> s['Economía']
4.5
>>> s[['Programación', 'Matemáticas']]
Programación   8.5
Matemáticas    6.0
dtype: float64
```

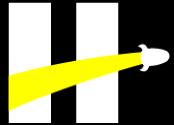
Resumen descriptivo

de una Serie

- `s.count()` : Devuelve el número de elementos que no son nulos ni NaN en la serie s.
- `s.sum()` : Devuelve la suma de los datos de la serie s cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str.
- `s.cumsum()` : Devuelve una serie con la suma acumulada de los datos de la serie s cuando los datos son de un tipo numérico.
- `s.value_counts()` : Devuelve una serie con la frecuencia (número de repeticiones) de cada valor de la serie s.
- `s.min()` : Devuelve el menor de los datos de la serie s.
- `s.max()` : Devuelve el mayor de los datos de la serie s.

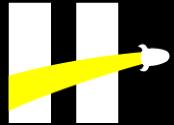
Resumen descriptivo de una Serie

- `s.mean()` : Devuelve la media de los datos de la serie s cuando los datos son de un tipo numérico.
- `s.std()` : Devuelve la desviación típica de los datos de la serie s cuando los datos son de un tipo numérico.
- `s.describe()`: Devuelve una serie con un resumen descriptivo que incluye el número de datos, su suma, el mínimo, el máximo, la media, la desviación típica y los cuartiles.



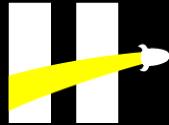
Operaciones con Series

- Los operadores binarios (+, *, /, etc.) pueden utilizarse con una serie, y devuelven otra serie con el resultado de aplicar la operación a cada elemento de la serie.
- `s.apply(f)` : Devuelve una serie con el resultado de aplicar la función `f` a cada uno de los elementos de la serie `s`.
- `s[condicion]` : Devuelve una serie con los elementos de la serie `s` que se corresponden con el valor `True` de la lista booleana `condición`. `condición` debe ser una lista de valores booleanos de la misma longitud que la serie.



Operaciones con Series

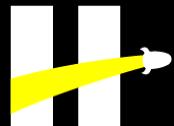
- `s.sort_values(ascending=boolano)` : Ordena los valores la serie s.
- `s.sort_index(ascending=boolano)` : Ordena el índice de la serie s.
- `s.dropna()` : Elimina los datos desconocidos o nulos de la serie s.



Dataframe

Define un conjunto de datos estructurado en forma de tabla donde cada columna es un objeto de tipo Series, y las filas son registros que pueden contener datos de distintos tipos. Un DataFrame contiene dos índices, uno para las filas y otro para las columnas, y se puede acceder a sus elementos mediante los nombres de las filas y las columnas.

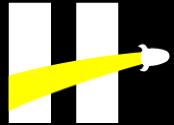
	Nombre	Edad	Grado	Correo
1	María	18	Economía	maria@gmail.com
2	Luis	22	Medicina	luis@yahoo.es
3	Carmen	20	Arquitectura	carmen@gmail.com
4	Antonio	21	Economía	antonio@gmail.com



Creación de Dataframes

A partir de un diccionario:

```
>>> import pandas as pd
>>> 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', 'antonio@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
```



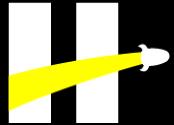
Creación de Dataframes

A partir de una lista de listas:

```
>>> import pandas as pd  
>>> df = pd.DataFrame([['María', 18], ['Luis', 22], ['Carmen', 20]], columns=['Nombre', 'Edad'])  
>>> print(df)  
    Nombre   Edad  
0   María     18  
1   Luis      22  
2 Carmen     20
```

A partir de una lista de diccionarios:

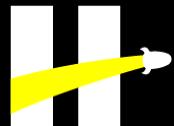
```
>>> import pandas as pd  
>>> df = pd.DataFrame([{'Nombre':'María', 'Edad':18}, {'Nombre':'Luis', 'Edad':22}, {'Nombre':'Carmen'}])  
>>> print(df)  
0   María  18.0  
1   Luis   22.0  
2 Carmen  NaN
```



Creación de Dataframes

A partir de un array:

```
>>> import pandas as pd  
>>> 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
```

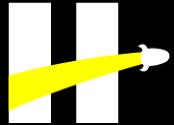


Creación de Dataframes

A partir de CSV o Excel:

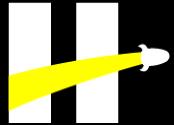
- `read_csv(fichero.csv, sep=separador, header=n, index_col=m, na_values=no-
validos, decimal=separador-decimal)`
- `read_excel(fichero.xlsx, sheet_name=hoja, header=n, index_col=m, na_values=no-
validos, decimal=separador-decimal)`

```
>>> df = pd.read_csv('colesterol.csv', sep=';', decimal=',')
>>> print(df.head())
      nombre  edad  sexo    peso   altura  colesterol
0  José Luis Martínez Izquierdo    18     H    85.0    1.79      182.0
1            Rosa Díaz Díaz    32     M    65.0    1.73      232.0
2  Javier García Sánchez    24     H    NaN    1.81      191.0
3  Carmen López Pinzón    35     M    65.0    1.70      200.0
4  Marisa López Collado    46     M    51.0    1.58      148.0
```



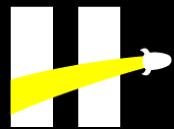
Exportación de Dataframes

- `df.to_csv(fichero.csv, sep=separador, columns=booleano, index=booleano)` :
Exporta el DataFrame df al archivo fichero.csv en formato CSV usando como separador de los datos la cadena separador. Si se pasa True al parámetro columns se exporta también la fila con los nombres de columnas y si se pasa True al parámetro index se exporta también la columna con los nombres de las filas.
- `df.to_excel(fichero.xlsx, sheet_name = hoja, columns=booleano, index=booleano)` :
Exporta el DataFrame df a la hoja de cálculo hoja del archivo fichero.xlsx en formato Excel. Si se pasa True al parámetro columns se exporta también la fila con los nombres de columnas y si se pasa True al parámetro index se exporta también la columna con los nombres de las filas.



Atributos de Dataframes

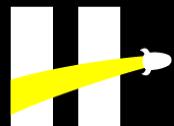
- df.info() : Devuelve información (número de filas, número de columnas, índices, tipo de las columnas y memoria usado).
- df.shape : Devuelve una tupla con el número de filas y columnas.
- df.size : Devuelve el número de elementos.
- df.columns : Devuelve una lista con los nombres de las columnas.
- df.index : Devuelve una lista con los nombres de las filas.
- df.dtypes : Devuelve una serie con los tipos de datos de las columnas.
- df.head(n) : Devuelve las n primeras filas.
- df.tail(n) : Devuelve las n últimas filas.



Operaciones sobre Dataframes

Agregar columnas:

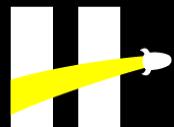
- `d[nombre] = lista`: Añade al DataFrame `df` una nueva columna con el nombre `nombre` y los valores de la lista `lista`. La lista debe tener el mismo tamaño que el número de filas de `df`.
- `d[nombre] = serie`: Añade al DataFrame `df` una nueva columna con el nombre `nombre` y los valores de la serie `serie`. Si el tamaño de la serie es menor que el número de filas de `df` se rellena con valores `NaN` mientras que si es mayor se recorta.



Operaciones sobre Dataframes

```
>>> import pandas as pd  
>>> df = pd.read_csv('colesterol.csv')  
>>> print(df['altura']*100)  
0    179  
1    173  
2    181  
...  
>>> print(df['sexo']=='M')  
0    False  
1     True  
2    False  
...
```

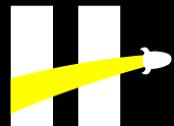
```
>>> import pandas as pd  
>>> from math import log  
>>> df = pd.read_csv('colesterol.csv')  
>>> print(df['altura'].apply(log))  
0    0.582216  
1    0.548121  
2    0.593327  
...
```



Operaciones sobre Dataframes

Convertir una columna, al tipo **datetime**

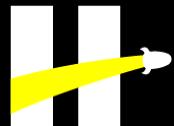
```
>>> import pandas as pd  
>>> df = pd.DataFrame({'Name': ['María', 'Carlos', 'Carmen'], 'Nacimiento':['05-03-2000', '20-05-2001', '10-12-1999']})  
>>> print(pd.to_datetime(df.Nacimiento, format = '%d-%m-%Y'))  
0    2000-03-05  
1    2001-05-20  
2    1999-12-10  
Name: Nacimiento, dtype: datetime64[ns]
```



Operaciones con Dataframes

- Renombrar filas y columnas:

```
>>> import pandas as pd
>>> df = pd.read_csv('colesterol.csv')
>>> print(df.loc[2, 'colesterol'])
191
>>> print(df.rename(columns={'nombre':'nombre y apellidos', 'altura':'estatura'}, index={0:1000, 1:1001, 2:1002}))
      nombre y apellidos  edad sexo    peso  estatura  colesterol
1000  José Luis Martínez Izquierdo   18    H    85.0     1.79      182.0
1001            Rosa Díaz Díaz   32    M    65.0     1.73      232.0
1002        Javier García Sánchez   24    H      NaN     1.81      191.0
3                Carmen López Pinzón   35    M    65.0     1.70      200.0
4            Marisa López Collado   46    M    51.0     1.58      148.0
...
...
```



Operaciones con Dataframes

- Reindexar:

```
>>> import pandas as pd
>>> df = pd.read_csv('colesterol.csv')
>>> print(df.reindex(index=[4, 3, 1], columns=['nombre', 'tensión', 'colesterol']))
      nombre   tensión   colesterol
4  Marisa López Collado     NaN     148.0
3  Carmen López Pinzón     NaN     200.0
1    Rosa Díaz Díaz     NaN     232.0
```

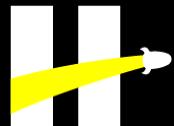


Operaciones con Dataframes

Eliminar columnas:

- del d[nombre] : Elimina la columna con nombre 'nombre'.
- df.pop(nombre) : Elimina la columna con nombre 'nombre' y la devuelve como una serie.

```
>>> import pandas as pd
>>> df = pd.read_csv('colesterol.csv')
>>> edad = df.pop('edad')
>>> print(df)
                nombre    sexo   peso   altura  colesterol
0  José Luis Martínez Izquierdo    H   85.0    1.79      182.0
1                  Rosa Díaz Díaz    M   65.0    1.73      232.0
2  Javier García Sánchez    H
NaN      1.81      191.0
...
print(edad)
0     18
1     32
2     24
...
```



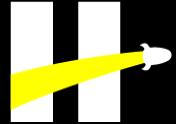
Operaciones con Dataframes

Eliminar filas:

- df.drop(filas) : Devuelve el DataFrame que resulta de eliminar las filas con los nombres indicados en la lista 'filas'.

```
>>> import pandas as pd
>>> df = pd.read_csv('colesterol.csv')
>>> print(df.drop([1, 3]))
```

	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
2	Javier García Sánchez	24	H	NaN	1.81	191.0
4	Marisa López Collado	46	M	51.0	1.58	148.0
...						

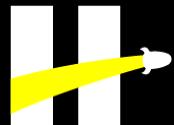


Operaciones con Dataframes

Eliminar filas con datos desconocidos:

- `s.dropna(subset=columnas)` : Devuelve el DataFrame que resulta de eliminar las filas que contienen algún dato desconocido o nulo en las columnas de la lista 'columna'. Si no se pasa un argumento al parámetro subset se aplica a todas las columnas.

```
>>> import pandas as pd
>>> df = pd.read_csv('colesterol1.csv')
>>> print(df.dropna())
          nombre  edad  sexo   peso  altura  colesterol
0  José Luis Martínez Izquierdo    18     H   85.0    1.79      182.0
1                  Rosa Díaz Díaz    32     M   65.0    1.73      232.0
3                  Carmen López Pinzón    35     M   65.0    1.70      200.0
4                  Marisa López Collado    46     M   51.0    1.58      148.0
...
...
```

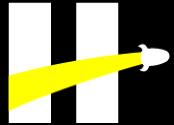


Operaciones con Dataframes

Filtrar filas:

- `df[condicion]` : Devuelve un DataFrame con las filas del DataFrame `df` que se corresponden con el valor `True` de la lista booleana '`condicion`', el cual debe ser una lista de valores booleanos de la misma longitud que el número de filas del DataFrame.

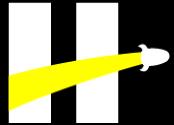
```
>>> import pandas as pd
>>> df = pd.read_csv('colesterol.csv')
>>> print(df[(df['sexo']=='H') & (df['colesterol'] > 260)])
      nombre  edad  sexo     peso    altura    colesterol
6  Antonio Fernández Ocaña    51     H    62.0    1.72      276.0
9   Santiago Reillo Manzano    46     H    75.0    1.85      280.0
```



Operaciones con Dataframes

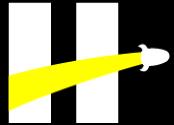
Ordenar:

- `df.sort_values(columna, ascending=booleano)` : Devuelve el DataFrame que resulta de ordenar las filas del DataFrame df según los valores de la columna con nombre 'columna'. Si argumento del parámetro ascending es True el orden es creciente y si es False decreciente.
- `df.sort_index(ascending=booleano)` : Devuelve el DataFrame que resulta de ordenar las filas del DataFrame df según los nombres de las 'filas'.



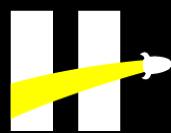
Acceso sobre el Dataframe

- `df.iloc[i, j]` : Devuelve el elemento que se encuentra en la fila `i` y la columna `j`. Pueden indicarse secuencias de índices para obtener partes del DataFrame.
- `df.iloc[filas, columnas]` : Devuelve un DataFrame con los elementos de las filas de la lista 'filas' y de las columnas de la lista 'columnas'.
- `df.iloc[i]` : Devuelve una serie con los elementos de la fila `i`.



Acceso sobre el Dataframe

- `df.loc[fila, columna]` : Devuelve el elemento que se encuentra en la fila con nombre fila y la columna de con nombre columna.
- `df.loc[filas, columnas]` : Devuelve un DataFrame con los elementos que se encuentra en las filas con los nombres de la lista filas y las columnas con los nombres de la lista columnas.
- `df[columna]` : Devuelve una serie con los elementos de la columna de nombre columna.
- `df.columna` : Devuelve una serie con los elementos de la columna de nombre columna. Es similar al método anterior pero solo funciona cuando el nombre de la columna no tiene espacios en blanco.



Resumen descriptivo del Dataframe

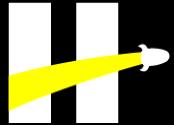
- df.count() : Devuelve una serie número de elementos que no son nulos ni NaN en cada columna.
- df.sum() : Devuelve una serie con la suma de los datos de las columnas cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str.
- df.cumsum() : Devuelve un DataFrame con la suma acumulada de los datos de las columnas cuando los datos son de un tipo numérico.



Resumen descriptivo

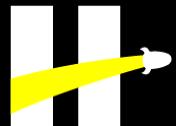
del Dataframe

- `df.mean()` : Devuelve una serie con las media de los datos de las columnas cuando los datos son de un tipo numérico.
- `df.std()` : Devuelve una serie con las desviaciones típicas de los datos de las columnas cuando los datos son de un tipo numérico.
- `df.describe(include = tipo)` : Devuelve un DataFrame con un resumen estadístico de las columnas del tipo 'tipo'. Para los datos numéricos (number) se calcula la media, la desviación típica, el mínimo, el máximo y los cuartiles de las columnas numéricas. Para los datos no numéricos (object) se calcula el número de valores, el número de valores distintos, la moda y su frecuencia. Si no se indica el tipo solo se consideran las columnas numéricas.



Reagrupar un Dataframe

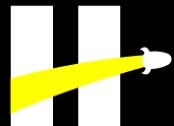
- `df.groupby(columnas).groups` : Devuelve un diccionario con cuyas claves son las tuplas que resultan de todas las combinaciones de los valores de las columnas con nombres en la lista 'columnas', y valores las listas de los nombres de las filas que contienen esos valores en las correspondientes columnas del DataFrame df.



Reagrupar un Dataframe

Aplicar una función de agregación por grupos:

- `df.groupby(columnas).agg(funciones)` : Devuelve un DataFrame con el resultado de aplicar las funciones de agregación de la lista funciones a cada uno de los DataFrames que resultan de dividir el DataFrame según las columnas de la lista columnas.
- Una **función de agregación** toma como argumento una lista y devuelve una único valor. Algunas de las funciones de agregación más comunes son: `np.min`, `np.max`, `np.count_nonzero`, `np.sum`, `np.mean`, `np.std`.



Reagrupar un Dataframe

Pivot:

- df.pivot(index=filas, columns=columna, values=valores) : Devuelve el DataFrame que resulta de convertir el DataFrame df de formato largo a formato ancho. Se crean tantas columnas nuevas como valores distintos haya en la columna 'columna'. Los nombres de estas nuevas columnas son los valores de la columna 'columna' mientras que sus valores se toman de la columna 'valores'. Los nombres del índice del nuevo DataFrame se toman de los valores de la columna 'filas'.