Pandas – gym example

Import Pandas module as pd

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

Reads the .csv file 'gym.csv' using ';' as separator

In [2]:
```
df = pd.read_csv('gym.csv', sep = ';', parse_dates=['date'])
```

Shows the first n rows from the DataFrame (5 rows if the argument is ommited)

In [3]:
```
df.head(3)
```

Out[3]:

	id	name	date	height	weight	age	hours	status	children	sex
0	1373900	Marisa Martins	2013-02-05	155	48	45	3	married	2	F
1	1109891	Rita Fonseca	2018-08-28	166	54	45	3	married	3	F
2	1158895	Joana Freitas	2013-10-21	150	42	52	3	widow	1	F

Shows the last n rows from the DataFrame (5 rows if the argument is ommited)

In [4]:
```
df.tail(3)
```

Out[4]:

	id	name	date	height	weight	age	hours	status	children	sex
57	1150196	Antonio Goncalves	2014-11-22	158	49	34	3	single	0	M
58	1658802	Manuel Freitas	2015-11-06	170	51	57	1	widow	2	M
59	1769504	Joao Tavares	2004-06-08	177	85	32	3	married	2	M

How many lines and columns has the DataFrame

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

Out[5]:
```
(60, 10)
```

General info about the DataFrame

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        60 non-null     int64
 1   name      60 non-null     object
 2   date      60 non-null     datetime64[ns]
 3   height    60 non-null     int64
 4   weight    60 non-null     int64
 5   age       60 non-null     int64
 6   hours     60 non-null     int64
 7   status    60 non-null     object
 8   children  60 non-null     int64
 9   sex       60 non-null     object
dtypes: datetime64[ns](1), int64(6), object(3)
memory usage: 4.8+ KB
```

Shows the type of data in each DataFrame column. Type object is Python string

In [7]:
```
df.dtypes
```

Out[7]:
```
id                  int64
name               object
date       datetime64[ns]
height              int64
weight              int64
age                 int64
hours               int64
status             object
children            int64
sex                object
dtype: object
```

Pandas – indexing and selecting data

Getting values from an object with multi-axes selection uses the following notation (using .loc as an example, but the following applies to .iloc as well). Any of the axes accessors may be the null slice :. Axes left out of the specification are assumed to be :, e.g. p.loc['a'] is equivalent to p.loc['a', :, :].

Object Type	Indexers
Series	s.loc[indexer]
DataFrame	df.loc[row_indexer,column_indexer]

The primary function of indexing with [] is selecting out lower-dimensional slices. The following table shows return type values when indexing pandas objects with []:

Object Type	Selection	Return Value Type
Series	series[label]	scalar value
DataFrame	frame[colname]	Series corresponding to colname

Pandas – indexing and selecting data

.loc

is primarily label based, but may also be used with a boolean array

- A single label, e.g. 5 or 'a' (Note that 5 is interpreted as a label of the index. This use is not an integer position along the index.).
- A list or array of labels ['a', 'b', 'c'].
- A slice object with labels 'a':'f' (Note that contrary to usual Python slices, both the start and the stop are included, when present in the index.)
- A boolean array (any NA values will be treated as False).
- A callable function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above).

Pandas – gym example - loc

Access the row with label 0. Returns a series

```
In [8]: df.loc[0]
```

```
Out[8]: id                    1373900
        name            Marisa Martins
        date       2013-02-05 00:00:00
        height                    155
        weight                     48
        age                        45
        hours                       3
        status                married
        children                    2
        sex                         F
        Name: 0, dtype: object
```

```
In [9]: type(df.loc[0])
```

```
Out[9]: pandas.core.series.Series
```

Access all the rows from column 'age'

```
In [10]: s = df.loc[:,'age']
```

```
In [11]: s.head()
```

```
Out[11]: 0    45
         1    45
         2    52
         3    59
         4    43
         Name: age, dtype: int64
```

```
In [12]: type(s)
```

```
Out[12]: pandas.core.series.Series
```

Access value in row with label 0 in column 'age'

```
In [13]: df.loc[0,'age']
```

```
Out[13]: 45
```

Access the rows with labels 0, 2 and 4

```
In [14]: df.loc[[0,2,4]]
```

Out[14]:

	id	name	date	height	weight	age	hours	status	children	sex
0	1373900	Marisa Martins	2013-02-05	155	48	45	3	married	2	F
2	1158895	Joana Freitas	2013-10-21	150	42	52	3	widow	1	F
4	1974598	Francisco Fonseca	2009-08-22	162	52	43	2	married	1	M

Access values in rows with labels 0 and 2 and columns 'name' and 'age'

```
In [15]: df.loc[[0,2],['name', 'age']]
```

Out[15]:

	name	age
0	Marisa Martins	45
2	Joana Freitas	52

Access the values between rows with labels 0 and 2 and between columns 'name' and 'age'

```
In [16]: df.loc[0:2,'name':'age']
```

Out[16]:

	name	date	height	weight	age
0	Marisa Martins	2013-02-05	155	48	45
1	Rita Fonseca	2018-08-28	166	54	45
2	Joana Freitas	2013-10-21	150	42	52

Pandas – indexing and selecting data

.iloc

is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array.

- An integer e.g. 5.
- A list or array of integers [4, 3, 0].
- A slice object with ints 1:7.
- A boolean array (any NA values will be treated as False).
- A callable function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above).

Pandas – gym example - iloc

Access the value in row 1 and column 1

```
In [17]: df.iloc[1,1]
Out[17]: 'Rita Fonseca'
```

Access to all the values in row 1

```
In [18]: df.iloc[1,:]
Out[18]: id                    1109891
         name              Rita Fonseca
         date       2018-08-28 00:00:00
         height                     166
         weight                      54
         age                         45
         hours                        3
         status                 married
         children                     3
         sex                          F
         Name: 1, dtype: object
```

Access to rows between 0 and 2 and columns between 2 and 4

```
In [19]: df.iloc[0:2,2:4]
Out[19]:
```

	date	height
0	2013-02-05	155
1	2018-08-28	166

Pandas – gym example - indexes

Row index, by default an integer row number

```
In [20]: df.index
```

```
Out[20]: RangeIndex(start=0, stop=60, step=1)
```

Column index - column names

```
In [21]: df.columns
```

```
Out[21]: Index(['id', 'name', 'date', 'height', 'weight', 'age', 'hours', 'status',
                'children', 'sex'],
               dtype='object')
```

Set column 'name' as the new index

```
In [22]: df.set_index('name', inplace=True)
```

The values in a row can be accessed using the index of the row

```
In [23]: df.loc['Jose Carvalho']
```

```
Out[23]: id                       1871098
         date         2004-09-17 00:00:00
         height                        153
         weight                         67
         age                            26
         hours                           7
         status                    married
         children                        1
         sex                             M
         Name: Jose Carvalho, dtype: object
```

The DataFrame index can be sorted

```
In [24]: df.sort_index(ascending=False)
```

Out[24]:

name	id	date	height	weight	age	hours	status	children	sex
Susana Marinho	1521104	2012-08-18	174	60	35	5	single	0	F
Susana Madeira	1436901	2008-09-05	160	49	56	2	divorced	2	F
Susana Goncalves	1170490	2013-07-03	157	65	40	2	married	5	F

The DataFrame can be sorted by the values in another column

```
In [25]: df.sort_values('age').head(5)
```

Out[25]:

name	id	date	height	weight	age	hours	status	children	sex
Francisco Pinho	1294205	2015-08-25	154	58	23	6	single	3	M
Francisco Carvalho	1653399	2002-05-09	150	66	23	7	single	0	M
Francisco Madeira	1692591	2012-09-24	154	49	23	8	single	0	M
Antonio Carvalho	1856504	2001-02-07	159	73	24	6	single	0	M
Manuel Martins	1767791	2003-01-25	179	85	24	6	single	0	M

Reset the index to the default

```
In [26]: df.reset_index(inplace=True)
```