Data manipulation

Reading and manipulating the data

Learning goals for this session

1. Learn the basics of examining, verifying and manipulation of data programmatically.

2. Learn to manipulate data with Python fluently.
   - Focus on **pandas** library

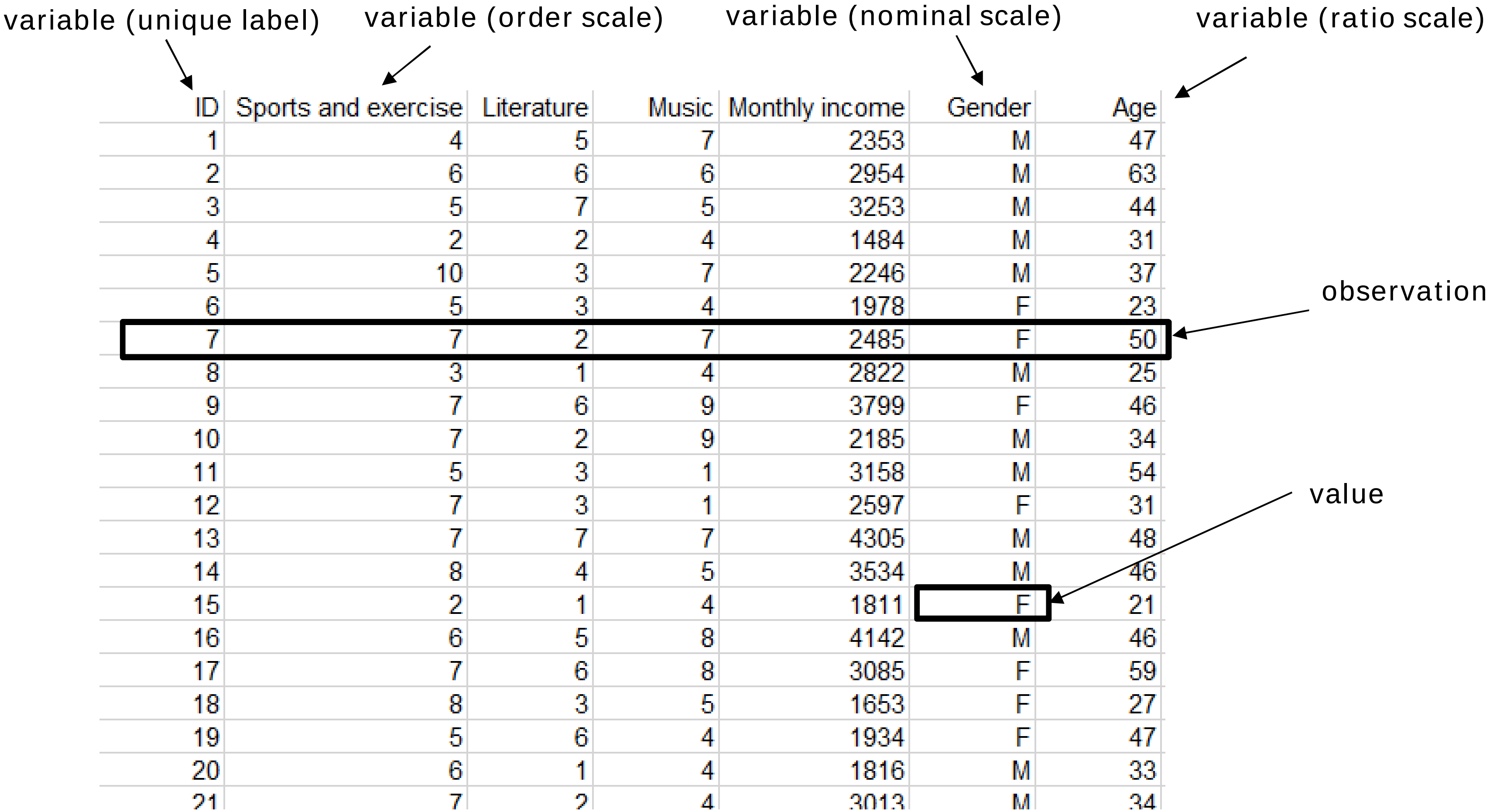
Python in machine learning

- Python is becoming increasingly popular in machine learning.
- Reasons:
  1. Easy-to-learn scripting language
  2. Extensive, well-documented ML libraries
  3. Intuitive analysis environments (specifically JuPyTer Notebook)
  4. Large user base

Data preparation

- Data preparation is the most important phase in a machine learning project.
- It is also the most time-consuming.
  - Takes about 90% of the time.
- Idea: convert the data into a single table.

Concepts of tabular data model



variable (unique label)

variable (order scale)

variable (nominal scale)

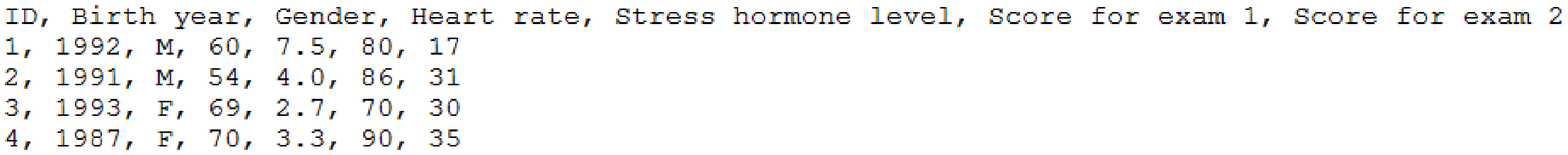
variable (ratio scale)

observation

value

ID	Sports and exercise	Literature	Music	Monthly income	Gender	Age
1	4	5	7	2353	M	47
2	6	6	6	2954	M	63
3	5	7	5	3253	M	44
4	2	2	4	1484	M	31
5	10	3	7	2246	M	37
6	5	3	4	1978	F	23
7	7	2	7	2485	F	50
8	3	1	4	2822	M	25
9	7	6	9	3799	F	46
10	7	2	9	2185	M	34
11	5	3	1	3158	M	54
12	7	3	1	2597	F	31
13	7	7	7	4305	M	48
14	8	4	5	3534	M	46
15	2	1	4	1811	F	21
16	6	5	8	4142	M	46
17	7	6	8	3085	F	59
18	8	3	5	1653	F	27
19	5	6	4	1934	F	47
20	6	1	4	1816	M	33
21	7	2	4	3013	M	34

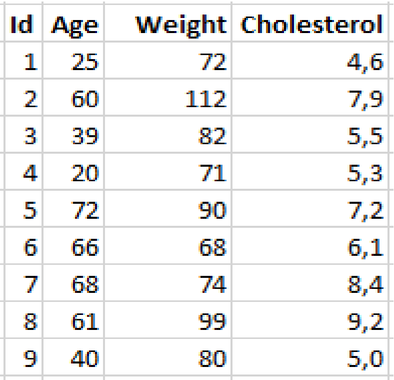
CSV format



```
ID, Birth year, Gender, Heart rate, Stress hormone level, Score for exam 1, Score for exam 2
1, 1992, M, 60, 7.5, 80, 17
2, 1991, M, 54, 4.0, 86, 31
3, 1993, F, 69, 2.7, 70, 30
4, 1987, F, 70, 3.3, 90, 35
```

- CSV is a comma (or semicolon) separated line-oriented data format.
  - It is supported in most machine learning environments.
- A CSV file is a human-readable ASCII file that can be edited with a common text editor.
- The first row contains variable names.
- The remaining rows contain observations.
  - The number of observations in data mining can vary from just a few to several billions.

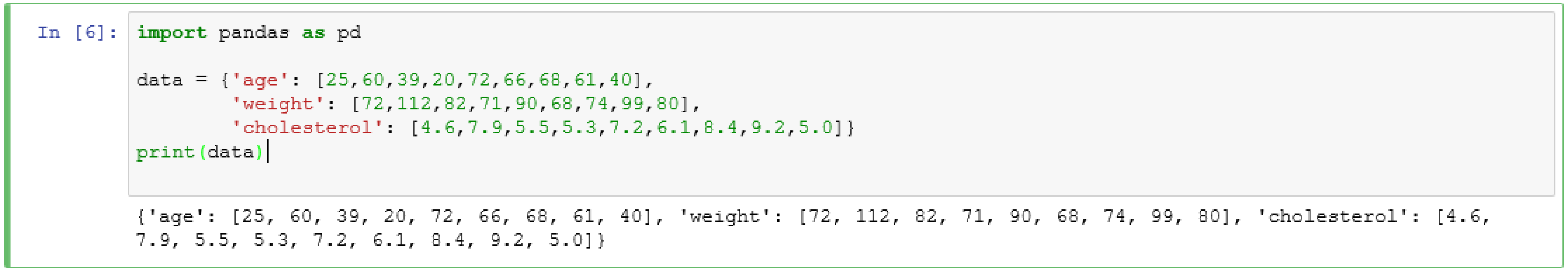
Python data manipulation example



Id	Age	Weight	Cholesterol
1	25	72	4,6
2	60	112	7,9
3	39	82	5,5
4	20	71	5,3
5	72	90	7,2
6	66	68	6,1
7	68	74	8,4
8	61	99	9,2
9	40	80	5,0

- Consider the data set above.
- The aim is to read the data set into Python and compute the basic statistics (minimum, maximum, average) for each of the variables.
- Verifying data quality is a vital steps for any inference and/or learning from the data.
  - Analyzing minimum, maximum, and mean values.
  - Inspecting outliers, coding missing data.
  - Bad quality data is useless: GIGO principle (“garbage in, garbage out”).

Step 1: coding tabular data in python

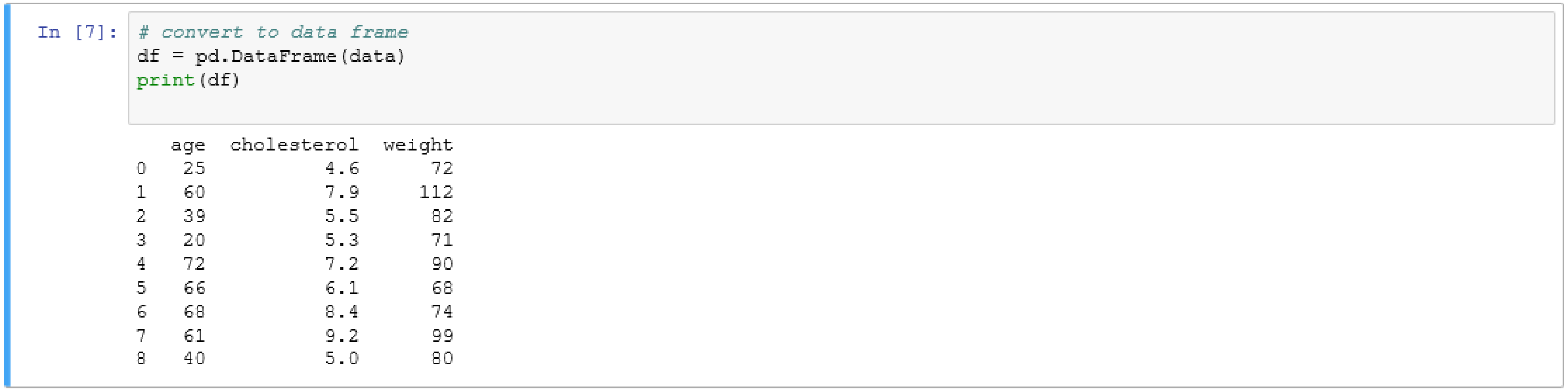


```
In [6]: import pandas as pd

        data = {'age': [25,60,39,20,72,66,68,61,40],
                'weight': [72,112,82,71,90,68,74,99,80],
                'cholesterol': [4.6,7.9,5.5,5.3,7.2,6.1,8.4,9.2,5.0]}
        print(data)
```

```
{'age': [25, 60, 39, 20, 72, 66, 68, 61, 40], 'weight': [72, 112, 82, 71, 90, 68, 74, 99, 80], 'cholesterol': [4.6,
7.9, 5.5, 5.3, 7.2, 6.1, 8.4, 9.2, 5.0]}
```

- A data set can be constructed from the scratch by generating an associative array of lists.
  - An associative array named **`data`** holds three (key, value) pairs in this example.
  - Each key is a string that holds the variable name.
  - Each value is a list of values in the observations.

Step 2: making a **pandas** data frame



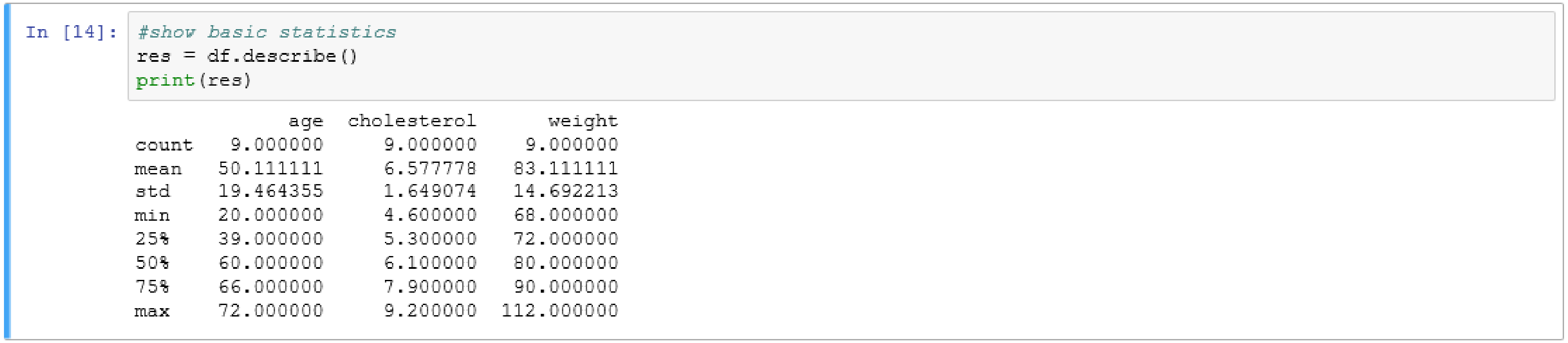
In [7]:

```
# convert to data frame
df = pd.DataFrame(data)
print(df)
```

```
   age  cholesterol  weight
0   25          4.6      72
1   60          7.9     112
2   39          5.5      82
3   20          5.3      71
4   72          7.2      90
5   66          6.1      68
6   68          8.4      74
7   61          9.2      99
8   40          5.0      80
```

- Many machine learning methods implemented in the **`scikit-learn`** assume **pandas** data frames as the input format.
- Above, the associative array is converted into a data frame and printed.
- **pd** refers to the alias name for the imported **pandas** library.
- From now on, use **pandas** documention at https://pandas.pydata.org/ → Documentation.

Step 3: printing basic statistics



```
In [14]: #show basic statistics
         res = df.describe()
         print(res)
```

```
             age  cholesterol      weight
count   9.000000     9.000000    9.000000
mean   50.111111     6.577778   83.111111
std    19.464355     1.649074   14.692213
min    20.000000     4.600000   68.000000
25%    39.000000     5.300000   72.000000
50%    60.000000     6.100000   80.000000
75%    66.000000     7.900000   90.000000
max    72.000000     9.200000  112.000000
```

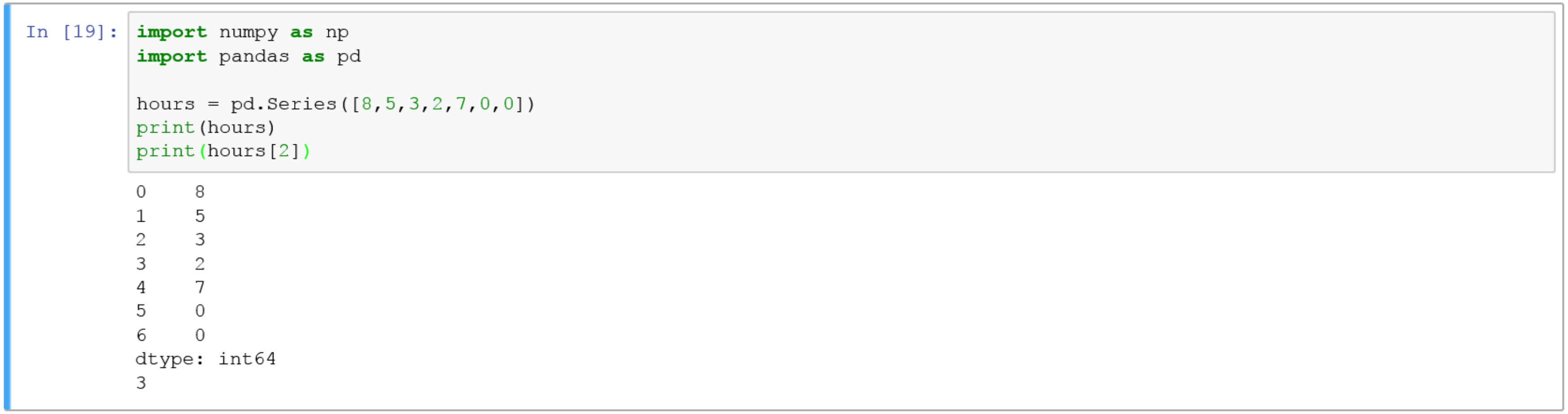
- By default, mean, standard deviation, minimum and maximum values as well as 25%, 50% and 75% percentiles are displayed.
  - E.g. the 75% percentile for cholesterol shows that 75% of the cholesterol values in the data set are smaller than or equal to 7.9.

Data structures

- Pandas supports two data structures:
  1. A series
  2. A data frame

- In addition, there is a deprecated Panel.
  - Deprecated functionality should not be used.

Series



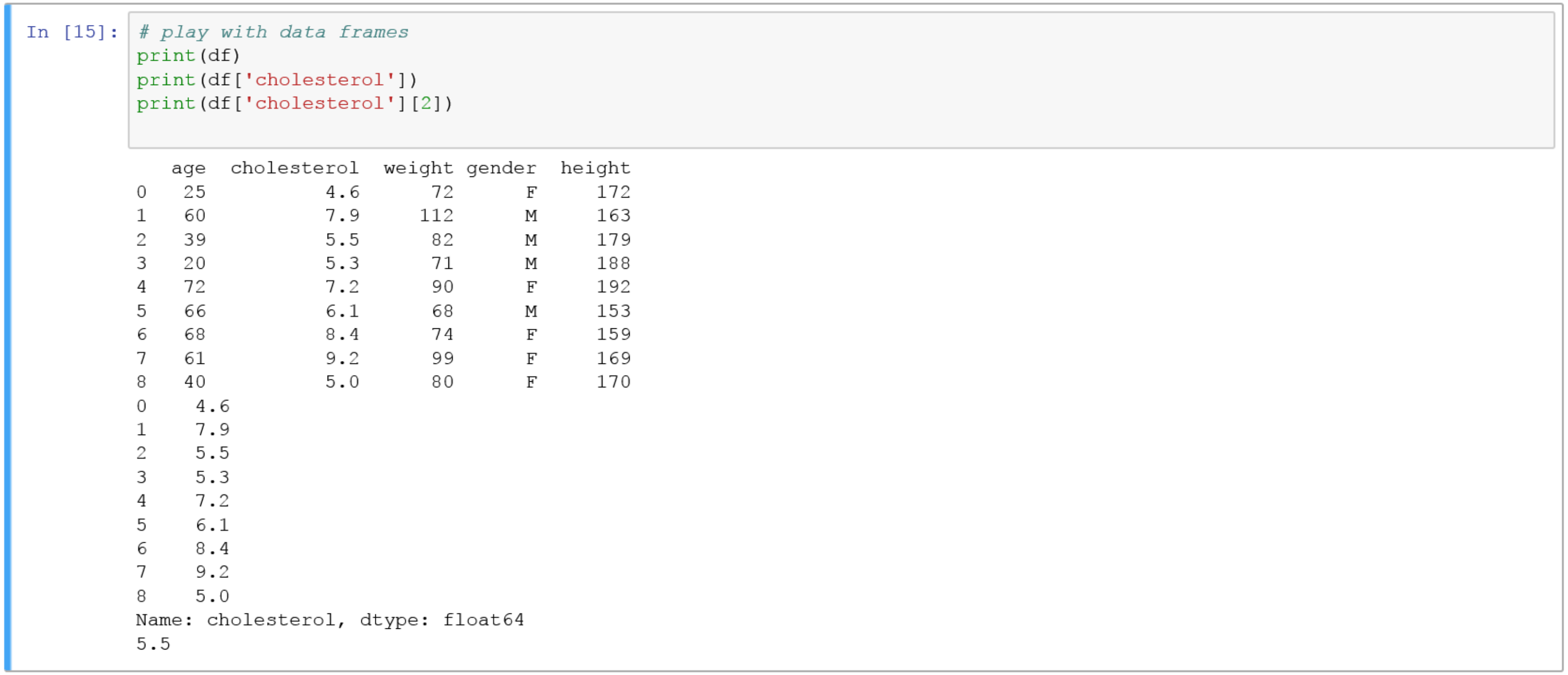
```
In [19]: import numpy as np
         import pandas as pd

         hours = pd.Series([8,5,3,2,7,0,0])
         print(hours)
         print(hours[2])
```

```
0    8
1    5
2    3
3    2
4    7
5    0
6    0
dtype: int64
3
```

- A series represents a vector of values.
  - One-dimensional
  - Labeled
  - Values can be of any data types

- Above, a series is constructed from a Python list structure.

Data frames



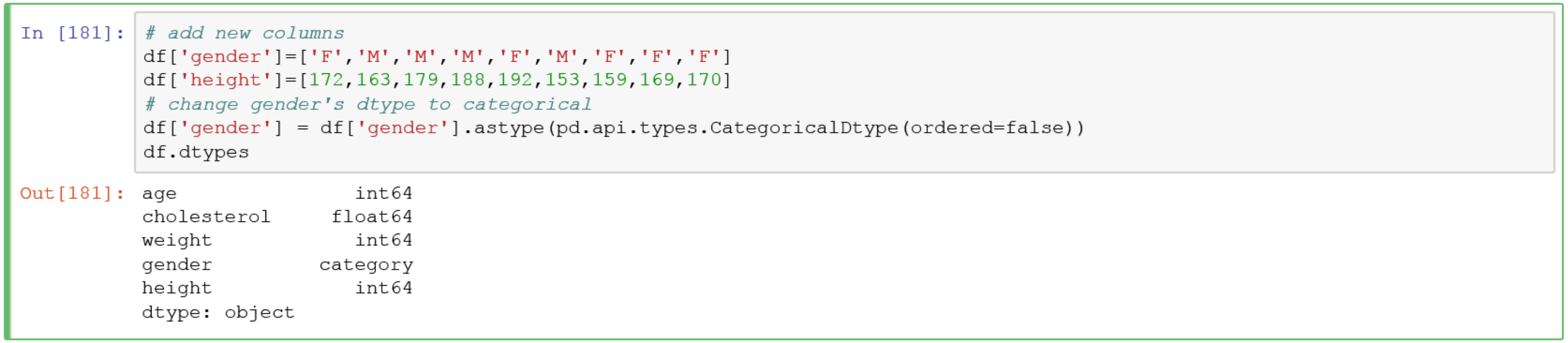
In [15]:

```
# play with data frames
print(df)
print(df['cholesterol'])
print(df['cholesterol'][2])
```

```
   age  cholesterol  weight gender  height
0   25          4.6      72      F     172
1   60          7.9     112      M     163
2   39          5.5      82      M     179
3   20          5.3      71      M     188
4   72          7.2      90      F     192
5   66          6.1      68      M     153
6   68          8.4      74      F     159
7   61          9.2      99      F     169
8   40          5.0      80      F     170
0    4.6
1    7.9
2    5.5
3    5.3
4    7.2
5    6.1
6    8.4
7    9.2
8    5.0
Name: cholesterol, dtype: float64
5.5
```

- A data frame stores an array of values.
- Columns can be accessed by labels.
- A data frame can be generated from various structures.

Adding columns, changing data types



```
In [181]: # add new columns
          df['gender']=['F','M','M','M','F','M','F','F','F']
          df['height']=[172,163,179,188,192,153,159,169,170]
          # change gender's dtype to categorical
          df['gender'] = df['gender'].astype(pd.api.types.CategoricalDtype(ordered=false))
          df.dtypes

Out[181]: age              int64
          cholesterol    float64
          weight           int64
          gender        category
          height           int64
          dtype: object
```

- The data types of the columns should reflect the scales of the variables.
- This is vital for some of the machine learning algorithms to work correctly.
- Commonly used data types:
  - For ratio scale: int64 or float64
  - For interval scale: int64 or float64
  - For ordinal scale: CategoricalDtype(ordered=True)
  - For nominal scale: CategoricalDtype(ordered=False)