Import and store data

## Import pandas

* To load the pandas package and start working with it, import the package.
* The community agreed alias for pandas is pd, so loading pandas as pd is assumed standard practice for all of the pandas documentation:

import pandas as pd

## Data creation

* To manually store data in a table, create a DataFrame:
* create the DataFrame and name it my\_df
* add the variable “gender” with the following observations: male, female and male

Hint:

my\_df = pd.DataFrame(  
 {   
 'name': [ "Tom", "Lisa", "Peter"],  
 'height': [1.68, 1.93, 1.72],  
 'weight': [48.4, 89.8, 84.2]  
 \_\_\_\_\_\_ \_\_\_\_\_\_\_\_  
 }  
)

### BEGIN SOLUTION  
my\_df = pd.DataFrame(  
 {   
 'name': [ "Tom", "Lisa", "Peter"],  
 'height': [1.68, 1.93, 1.72],  
 'weight': [48.4, 89.8, 84.2],  
 'gender': ['male', 'female', 'male']   
 }  
)  
### END SOLUTION

"""Check that your code returns the correct output"""  
assert my\_df.iloc[0, 0] == "Tom"  
assert my\_df.iloc[0, 3] == "male"  
assert my\_df.iloc[1, 3] == "female"

## Import data

Pandas supports many different file formats or data sources out of the box (csv, excel, sql, json, parquet, …). Each of them import data with the prefix read\_\*

* Next, we import data, available as a CSV file in a GitHub repo

Important notes: The CSV file has a typical european format with

* ; as seperators between cells (instead of the Pandas default ,)
* , as number decimal seperator (instead of the Pandas default .)

We need to take care of this issues.

Hint

df = pd.\_\_\_(sep=';', decimal=',')

# URL of data  
URL = "https://raw.githubusercontent.com/kirenz/datasets/master/height\_unclean.csv"

### BEGIN SOLUTION  
df = pd.read\_csv(URL, sep=";", decimal=',')  
### END SOLUTION

"""Check that your code returns the correct output"""  
assert df.iloc[0, 0] == 'Stefanie'  
assert df.iloc[0, 3] == 161.5

Now we do the same import with a different approach. Use the name df2

Hint:

ROOT = "https://raw.githubusercontent.com/kirenz/datasets/master/"  
DATA = "\_\_\_"  
  
df2 = pd.read\_csv(\_\_\_ + \_\_\_, sep=";", decimal=',' )  
  
# show head of the data  
df2.head()

### BEGIN SOLUTION  
ROOT = "https://raw.githubusercontent.com/kirenz/datasets/master/"  
DATA = "height\_unclean.csv"  
  
df2 = pd.read\_csv(ROOT + DATA, sep=";", decimal=',' )  
  
df2.head()  
### END SOLUTION

|  | Name | ID% | Height | Average Height Parents | Gender |
| --- | --- | --- | --- | --- | --- |
| 0 | Stefanie | 1 | 162 | 161.5 | female |
| 1 | Peter | 2 | 163 | 163.5 | male |
| 2 | Stefanie | 3 | 163 | 163.2 | female |
| 3 | Manuela | 4 | 164 | 165.1 | female |
| 4 | Simon | 5 | 164 | 163.2 | male |

"""Check that your code returns the correct output"""  
assert df2.iloc[0, 0] == 'Stefanie'  
assert df2.iloc[0, 3] == 161.5

## Store data

Pandas supports many different file formats (csv, excel, sql, json, parquet, …). Each of them stores data with the prefix to\_\*

* The following code should save the data df as an Excel file in your current directory.
* In the example here, the sheet\_name is named people\_height instead of the default Sheet1.
* By setting index=False the row index labels are not saved in the spreadsheet:

df.to\_excel("height.xlsx", sheet\_name="people\_height", index=False)

* The equivalent read function read\_excel() would reload the data to a DataFrame:

# load excel file  
df\_xlsx = pd.read\_excel("height.xlsx", sheet\_name="people\_height")

## Viewing data

### Overview

# show df  
df

|  | Name | ID% | Height | Average Height Parents | Gender |
| --- | --- | --- | --- | --- | --- |
| 0 | Stefanie | 1 | 162 | 161.5 | female |
| 1 | Peter | 2 | 163 | 163.5 | male |
| 2 | Stefanie | 3 | 163 | 163.2 | female |
| 3 | Manuela | 4 | 164 | 165.1 | female |
| 4 | Simon | 5 | 164 | 163.2 | male |
| 5 | Sophia | 6 | 164 | 164.4 | female |
| 6 | Ellen | 7 | 164 | 164.0 | female |
| 7 | Emilia | 8 | 165 | 165.2 | female |
| 8 | Lina | 9 | 165 | 165.2 | female |
| 9 | Marie | 10 | 165 | 165.1 | female |
| 10 | Lena | 11 | 165 | 166.3 | female |
| 11 | Mila | 12 | 165 | 167.4 | female |
| 12 | Fin | 13 | 165 | 165.5 | male |
| 13 | Eric | 14 | 166 | 166.2 | male |
| 14 | Pia | 15 | 166 | 166.1 | female |
| 15 | Marc | 16 | 166 | 166.5 | male |
| 16 | Ralph | 17 | 166 | 166.6 | male |
| 17 | Tom | 18 | 167 | 166.2 | male |
| 18 | Steven | 19 | 167 | 167.3 | male |
| 19 | Emanuel | 20 | 168 | 168.5 | male |

# show first 2 rows  
df.head(2)

|  | Name | ID% | Height | Average Height Parents | Gender |
| --- | --- | --- | --- | --- | --- |
| 0 | Stefanie | 1 | 162 | 161.5 | female |
| 1 | Peter | 2 | 163 | 163.5 | male |

# show last 2 rows  
df.tail(2)

|  | Name | ID% | Height | Average Height Parents | Gender |
| --- | --- | --- | --- | --- | --- |
| 18 | Steven | 19 | 167 | 167.3 | male |
| 19 | Emanuel | 20 | 168 | 168.5 | male |

* The info() method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage:

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 20 entries, 0 to 19  
Data columns (total 5 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Name 20 non-null object   
 1 ID% 20 non-null int64   
 2 Height 20 non-null int64   
 3 Average Height Parents 20 non-null float64  
 4 Gender 20 non-null object   
dtypes: float64(1), int64(2), object(2)  
memory usage: 928.0+ bytes

### Column names

# Show columns  
df.columns

Index(['Name', 'ID%', 'Height', 'Average Height Parents', ' Gender'], dtype='object')

### Data type

* Show data types ([dtypes](https://pandas.pydata.org/docs/user_guide/basics.html#basics-dtypes)).

df.dtypes

Name object  
ID% int64  
Height int64  
Average Height Parents float64  
 Gender object  
dtype: object

* The data types in this DataFrame are integers (int64), floats (float64) and strings (object).

### Index

# Only show index  
df.index

RangeIndex(start=0, stop=20, step=1)