6 - Pandas

Data Science Practicum 2021/22, Lesson 6

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Packages

- Packages enlarge the functionality of core Python
- PIP - Python Package Manager

```
pip install PackageName

pip uninstall PackageName

pip install --upgrade PackageName
```

- Anaconda Package Manager

```
conda list

conda search PackageName

conda install PackageName

conda update PackageName
```

Packages - Usage

```
import <module>
import <module> as <alt_name>

from <module> import <name>
from <module> import <name> as <alt_name>
```

```
import pandas
import pandas as pd

from pandas import DataFrame
from pandas import DataFrame as df
```

Table of Contents

Pandas

- Open source, BSD-licensed library
- high-performance, flexible , easy-to-use data structures
- tools for data analysis

Importing Data

- Format (e.g., CSV, HTML, JSON, SQL, MS Excel)
- File path (on your computer or online)

Importing Data

- Format (e.g., CSV, HTML, JSON, SQL, MS Excel)
- File path (on your computer or online)

```
import pandas as pd

path = "http://bit.do/car_data_csv"

data_frame = pd.read_csv(path, header = None)
```

Printing Data

```
data_frame
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

205 rows × 26 columns

Printing Data

```
data_frame
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

205 rows × 26 columns

```
data_frame.head(5)
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

Printing Data

```
data_frame.tail(3)
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

3 rows × 26 columns

Dataframe Basics

- Column labels of the DataFrame

```
data_frame.columns
```

- The index (i.e. row labels) of the DataFrame.

```
data_frame.index
```

- Values: Returns a Numpy representation of the DataFrame.

```
data_frame.values

or

data_frame.to_numpy()
```

- Dimensions

```
data_frame.ndim
```

Setting Columns

```
headers  =  ["symboling", "normalized-losses", "make", "fuel-type", "aspiration", "num-of-doors" , "body-style",
"drive-wheels", "engine-location", "wheel-base", "length", "width", "height" , "curb-weight", "engine-type",
"num-of-cylinders", "engine-size", "fuel-system", "bore", "stroke", "compression-ratio", "horsepower", "peak-rpm",
"city-mpg", "highway-mpg", "price"]

data_frame.columns = headers
```

Importing/Exporting

```
pd.read_csv()
pd.read_html()
pd.read_excel()
pd.read_json()
pd.read_sql()
```

```
data_frame.to_csv()
data_frame.to_html()
data_frame.to_excel()
data_frame.to_json()
data_frame.to_sql()
```

Importing/Exporting

```
pd.read_csv()
pd.read_html()
pd.read_excel()
pd.read_json()
pd.read_sql()
```

```
data_frame.to_csv()
data_frame.to_html()
data_frame.to_excel()
data_frame.to_json()
data_frame.to_sql()
```

Parameters:

- header: Write out the column names. If a list of string is given it is assumed to be aliases for the column names.
- index: Write row names (index)

```
file_name="car_data.csv"
data_frame.to_csv(file_name, header=True, index=True)

file_name="car_data.html"
data_frame.to_html(file_name, header=True,index=True)

file_name="car_data.xls"
data_frame.to_excel(file_name, header=True,index=True)
```

Describe

```
data_frame.describe()
```

	0	9	10	11	12	13	16	20	23	24
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

Accessing Data

- Accessing rows/columns by **integer** location

```
data_frame.iloc[ ]
```

- Accessing rows/columns by **labels** or **boolean** array

```
data_frame.loc[ ]
```

Accessing Data

- Accessing rows/columns by **integer** location

```
data_frame.iloc[ ]
```

- Accessing rows/columns by **labels** or **boolean** array

```
data_frame.loc[ ]
```

```
data_frame.iloc[0,:]  # first row

data_frame.iloc[:,0]  # first column

data_frame.iloc[0:5,:]  # first five columns

data_frame.iloc[:,0:5]  # first five rows
```

Accessing Data

- Accessing rows/columns by **integer** location

```
data_frame.iloc[ ]
```

- Accessing rows/columns by **labels** or **boolean** array

```
data_frame.loc[ ]
```

```
data_frame.iloc[0,:]  # first row

data_frame.iloc[:,0]  # first column

data_frame.iloc[0:5,:]  # first five columns

data_frame.iloc[:,0:5]  # first five rows
```

```
data_frame.loc[0,:]   # first row: "0" is the label of the row, not integer location!!!

data_frame.loc[:,"symboling"] # first column

data_frame.loc[0:4,:] # first 5 rows: "0" and "4" are labels

data_frame.loc[:,"symboling":"aspiration"]  # first 5 columns
```

Setting Index

- We can set the DataFrame index (i.e. set the row labels) using one or more existing columns

```
data_frame_by_make = data_frame.set_index("make")

data_frame_by_make.loc["audi"]
```

make	symboling	normalized-losses	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsep
audi	2	164	gas	std	four	sedan	fwd	front	99.8	176.6	...	109	mpfi	3.19	3.40	10.0	
audi	2	164	gas	std	four	sedan	4wd	front	99.4	176.6	...	136	mpfi	3.19	3.40	8.0	
audi	2	?	gas	std	two	sedan	fwd	front	99.8	177.3	...	136	mpfi	3.19	3.40	8.5	
audi	1	158	gas	std	four	sedan	fwd	front	105.8	192.7	...	136	mpfi	3.19	3.40	8.5	
audi	1	?	gas	std	four	wagon	fwd	front	105.8	192.7	...	136	mpfi	3.19	3.40	8.5	
audi	1	158	gas	turbo	four	sedan	fwd	front	105.8	192.7	...	131	mpfi	3.13	3.40	8.3	
audi	0	?	gas	turbo	two	hatchback	4wd	front	99.5	178.2	...	131	mpfi	3.13	3.40	7.0	

7 rows × 25 columns

Boolean Index

- We can create an index array with true/false values for the rows we want to extract

```
my_index = data_frame_by_make["num-of-doors"]=="two"
```

Boolean Index

- We can create an index array with true/false values for the rows we want to extract

```
my_index = data_frame_by_make["num-of-doors"]=="two"
```

```
make
alfa-romero     True
alfa-romero     True
audi           False
audi           False
audi            True
               ...
volvo          False
volvo          False
volvo          False
volvo          False
volvo          False
Name: num-of-doors, Length: 204, dtype: bool
```

Boolean Index

- We can create an index array with true/false values for the rows we want to extract

```
my_index = data_frame_by_make["num-of-doors"]=="two"
```

```
make
alfa-romero     True
alfa-romero     True
audi           False
audi           False
audi            True
               ...
volvo          False
volvo          False
volvo          False
volvo          False
volvo          False
Name: num-of-doors, Length: 204, dtype: bool
```

```
data_frame_by_make.loc[my_index]
```

Shape

- Getting the dimensions of a data frame

```
data_frame.shape
```

```
[output]: (205, 26)
```

Table of Contents

Pandas Exercises

Import

- use the following two functions from the object `DataFrame` from the pandas library in the correct way

```
l = [[1,2,3],[4,5,6],[7,8,9]]
df = pd.DataFrame(l)
df.head()
```

- given one of the importing approaches:

```
import <module>
import <module> as <alt_name>

from <module> import <name>
from <module> import <name> as <alt_name>
```

Import 1

```
import <module>
```

- Before starting: `Kernel.Restart & Clear Output`

Import 1

```
import <module>
```

- Before starting: `Kernel.Restart & Clear Output`

```
import pandas
l = [[1,2,3],[4,5,6],[7,8,9]]
df = pandas.DataFrame(l)
df.head()
```

Import 2

```
import <module> as <alt_name>
```

- Before starting: `Kernel.Restart & Clear Output`

Import 2

```
import <module> as <alt_name>
```

- Before starting: `Kernel.Restart & Clear Output`

```
import pandas as pd
l = [[1,2,3],[4,5,6],[7,8,9]]
df = pd.DataFrame(l)
df.head()
```

Import 3

```
from <module> import <name>
```

- Before starting: `Kernel.Restart & Clear Output`

Import 3

```
from <module> import <name>
```

- Before starting: `Kernel.Restart & Clear Output`

```
from pandas import DataFrame
l = [[1,2,3],[4,5,6],[7,8,9]]
df = DataFrame(l)
df.head()
```

Import 4

```
from <module> import <name> as <alt_name>
```

- Before starting: `Kernel.Restart & Clear Output`

Import 4

```
from <module> import <name> as <alt_name>
```

- Before starting: `Kernel.Restart & Clear Output`

```
from pandas import DataFrame as daf
l = [[1,2,3],[4,5,6],[7,8,9]]
df = daf(l)
df.head()
```

Importing

- most common way of importing the main data science libraries:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

read_csv()

- use the read_csv() method to read the file `delimiter.csv`
- use `head()` to display the top rows
- what do you observe?

read_csv()

- use the read_csv() method to read the file `delimiter.csv`
- use `head()` to display the top rows
- what do you observe?

```
import pandas as pd
df = pd.read_csv("delimiter.csv")
df.head()
```

```
  1.2   3.4   5.6   7.8
0   5.6   4.8   6.2   9.9
1   5.6   7.3   6.1   6.3
```

read_csv()

- use the read_csv() method to read the file `delimiter.csv`
- use head() to display the top rows
- what do you observe?

```
import pandas as pd
df = pd.read_csv("delimiter.csv")
df.head()
```

```
  1.2   3.4   5.6   7.8
0   5.6   4.8   6.2   9.9
1   5.6   7.3   6.1   6.3
```

- change the delimiter to dot

read_csv()

- use the read_csv() method to read the file `delimiter.csv`
- use head() to display the top rows
- what do you observe?

```
import pandas as pd
df = pd.read_csv("delimiter.csv")
df.head()
```

```
   1.2   3.4   5.6   7.8
0   5.6   4.8   6.2   9.9
1   5.6   7.3   6.1   6.3
```

- change the delimiter to dot

```
import pandas as pd
df = pd.read_csv("delimiter.csv", delimiter=".")
df.head()
```

```
   1   2,3   4,5   6,7   8
0   5   6,4   8,6   2,9   9
1   5   6,7   3,6   1,6   3
```

Decimal Point

- change the decimal point to comma

Decimal Point

- change the decimal point to comma

```
import pandas as pd
df = pd.read_csv("delimiter.csv", delimiter=".", decimal=",")
```

```
   1   2,3   4,5   6,7   8
0  5   6.4   8.6   2.9   9
1  5   6.7   3.6   1.6   3
```

About the data in the Dataframe

- use the following methods to get more info about the data:
  - head(n)
  - tail(n)
  - columns
  - index
  - values
  - ndim
  - describe()

Exercise

- make a 3x3 list with integer numbers
- create a dataframe out of it (pass the list as a parameter to the dataframe)
- print the dataframe
- what are the names of the columns?

Exercise

- make a 3x3 list with integer numbers
- create a dataframe out of it (pass the list as a parameter to the dataframe)
- print the dataframe
- what are the names of the columns?

```
l = [[1,2,3],[4,5,6],[7,8,9]]
df = pd.DataFrame(l)
df
```

```
   0   1   2
0  1   2   3
1  4   5   6
2  7   8   9
```

Exercise

- make a 3x3 list with integer numbers
- create a dataframe out of it (pass the list as a parameter to the dataframe)
- print the dataframe
- what are the names of the columns?

```
l = [[1,2,3],[4,5,6],[7,8,9]]
df = pd.DataFrame(l)
df
```

```
   0  1  2
0  1  2  3
1  4  5  6
2  7  8  9
```

- set the column names to A,B,C

Exercise

- make a 3x3 list with integer numbers
- create a dataframe out of it (pass the list as a parameter to the dataframe)
- print the dataframe
- what are the names of the columns?

```
l = [[1,2,3],[4,5,6],[7,8,9]]
df = pd.DataFrame(l)
df
```

```
   0   1   2
0  1   2   3
1  4   5   6
2  7   8   9
```

- set the column names to A,B,C

```
cols = ["A","B","C"]
df.columns = cols
df
```

```
   A   B   C
0  1   2   3
1  4   5   6
2  7   8   9
```

Accessing Data

- load the iris dataset
- print the values of the column `sepal_width` at the rows 5,8, and 50
- do not use loc/iloc

Accessing Data

- load the iris dataset
- print the values of the column sepal_width at the rows 5,8, and 50
- do not use loc/iloc

```
import pandas as pd
df = pd.read_csv("iris.data.head.csv")

print(df["sepal_width"][5])
print(df["sepal_width"][8])
print(df["sepal_width"][50])
```

```
3.9
2.9
3.2
```

1. column
2. row

Accessing data with loc

```
DataFrame.iloc[row][col]
```

- print the values of the column `sepal_width` at the rows 5,8, and 50

Accessing data with loc

```
DataFrame.iloc[row][col]
```

- print the values of the column sepal_width at the rows 5,8, and 50

```
print(df.iloc[5,1])
print(df.iloc[8,1])
print(df.iloc[50,1])
```

```
3.9
2.9
3.2
```

Accessing data with loc

```
DataFrame.iloc[row][col]
```

- print the values of the column sepal_width at the rows 5,8, and 50

```
print(df.iloc[5,1])
print(df.iloc[8,1])
print(df.iloc[50,1])
```

```
3.9
2.9
3.2
```

```
print(df.iloc[[5,8,50],1])
```

```
5     3.9
8     2.9
50    3.2
Name: sepal_width, dtype: float64
```

Exercise

- print out the rows 70 to 80 and all columns

Exercise

- print out the rows 70 to 80 and all columns

```
print(df.iloc[70:80,:])
```

```
    sepal_length  sepal_width  petal_length  petal_width           class
70           5.9          3.2           4.8          1.8  Iris-versicolor
71           6.1          2.8           4.0          1.3  Iris-versicolor
72           6.3          2.5           4.9          1.5  Iris-versicolor
73           6.1          2.8           4.7          1.2  Iris-versicolor
74           6.4          2.9           4.3          1.3  Iris-versicolor
75           6.6          3.0           4.4          1.4  Iris-versicolor
76           6.8          2.8           4.8          1.4  Iris-versicolor
77           6.7          3.0           5.0          1.7  Iris-versicolor
78           6.0          2.9           4.5          1.5  Iris-versicolor
79           5.7          2.6           3.5          1.0  Iris-versicolor
```

Exercise

- print the columns 2 to 4 and the rows 30 to 40

Exercise

- print the columns 2 to 4 and the rows 30 to 40

```
print(df.iloc[30:40,2:4])
```

```
    petal_length  petal_width
30           1.6          0.2
31           1.5          0.4
32           1.5          0.1
33           1.4          0.2
34           1.5          0.1
35           1.2          0.2
36           1.3          0.2
37           1.5          0.1
38           1.3          0.2
39           1.5          0.2
```

Exercise - Loc

- loc: labels for columns
- print the rows 10 to 20 for the labels:
  - `sepal_length`
  - `sepal_width`
  - `petal_length`

Exercise - Loc

- loc: labels for columns
- print the rows 10 to 20 for the labels:
  - sepal_length
  - sepal_width
  - petal_length

```
print(df.loc[10:20,"sepal_length","sepal_width","petal_length"])
```

```
print(df.loc[10:20,"sepal_length":"petal_length"])
```

Exercise

- make an index variable for when the `sepal_length` is bigger than 5

Exercise

- make an index variable for when the sepal_length is bigger than 5

```
i1 = df["sepal_length"] > 5
i1
```

- what is the type of the index variable?
- what can we do with it?

Exercise

- print the rows (all columns) where the `petal_length` is smaller than 1.5

Exercise

- print the rows (all columns) where the `petal_length` is smaller than 1.5

```
i1 = df["petal_length"] <1.5
df.loc[i1]
```

Exercise

- make a function that loads the iris dataset into a dataframe
- make a function that returns an index of the rows where
  - sepal_length is smaller than the average of sepal_length
- print the rows that correspond to the index
- make a function that returns the average value of the rows that correspond to the index

Exercise

- make a function that loads the iris dataset into a dataframe
- make a function that returns an index of the rows where
  - sepal_length is smaller than the average of sepal_length
- print the rows that correspond to the index
- make a function that returns the average value of the rows that correspond to the index

```
import pandas as pd

def load_iris():
    df = pd.read_csv("iris.data.head.csv")
    return df

def get_small_sepal_length(df):
    m = df.loc[:,"sepal_length"].mean()
    print(m)
    i = df.loc[:,"sepal_length"] < m
    return i

def get_new_mean(df,i):
        df2 = df.loc[i]
        m = df2.loc[:,"sepal_length"].mean()
        return m

d = load_iris()
i = get_small_sepal_length(d)
print(df.loc[i])
print(get_new_mean(d,i))
```

Exercise

- make the exercise above using a class

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

- https://realpython.com/python-namespaces-scope/
- https://www.tutorialspoint.com/python/python_functions.htm
- https://www.w3schools.com/python/python_functions.asp
- https://pynative.com/python-functions-exercise-with-solutions/
- https://www.w3schools.com/python/python_classes.asp
- https://stackoverflow.com/questions/625083/what-init-and-self-do-on-python