Fachhochschule Südwestfalen
University of Applied Sciences

Exercises

Advanced Analytics

Prof. Dr. Christian Leubner

Wintersemester 2025/26

1 Introduction to Python (Lambert book)

Exercise 1:

a) Write a Python program that prints (displays) your name, address, and telephone number.

b) Write and test a program that computes the area of a circle. This program should request a number representing a radius as input from the user. It should use the formula

```
3.14 * radius ** 2
```

to compute the area and then output this result suitably labeled.

Exercise 2:

a) You can calculate the surface area of a cube if you know the length of an edge. Write a program that takes the length of an edge (an integer) as input and prints the cube's surface area as output.

b) Write a program that takes the radius of a sphere (a floating-point number) as input and then outputs the sphere's diameter, circumference, surface area, and volume.

c) An employee's total weekly pay equals the hourly wage multiplied by the total number of regular hours plus any overtime pay. Overtime pay equals the total overtime hours multiplied by 1.5 times the hourly wage. Write a program that takes as inputs the hourly wage, total regular hours, and total overtime hours and displays an employee's total weekly pay.

Exercise 3:

a) A local biologist needs a program to predict population growth. The inputs would be the initial number of organisms, the rate of growth (a real number greater than 0), the number of hours it takes to achieve this rate, and a number of hours during which the population grows. For example, one might start with a population of 500 organisms, a growth rate of 2, and a growth period to achieve this rate of 6 hours. Assuming that none of the organisms die, this would imply that this population would double in size every 6 hours. Thus, after allowing 6 hours for growth, we would have 1000 organisms, and after 12 hours, we would have 2000 organisms. Write a program that takes these inputs and displays a prediction of the total population.

b) The greatest common divisor of two positive integers, A and B, is the largest number that can be evenly divided into both of them. Euclid's algorithm can be used to find the greatest common divisor (GCD) of two positive integers. You can use this algorithm in the following manner:

   a) Compute the remainder of dividing the larger number by the smaller number.
   b) Replace the larger number with the smaller number and the smaller number with the remainder.
   c) Repeat this process until the smaller number is zero.

   The larger number at this point is the GCD of A and B. Write a program that lets the user enter two integers and then prints each step in the process of using the Euclidean algorithm to find their GCD.

c) Write a program that receives a series of numbers from the user and allows the user to press the enter key to indicate that he or she is finished providing inputs. After the user presses the enter key, the program should print the sum of the numbers and their average.

d) Write a function `exponential` that calculates and returns the exponential value for a given number `number` (base) and an (integer) exponent `exponent`. Use a loop for this. Then determine the result for 2^{16} (gives 65,536).

e) In mathematics, the **factorial** is a function that assigns to a natural number the product of all natural numbers less than or equal to this number. The sign for the factorial is an exclamation mark (!).

Example:

- $1! = 1$
- $2! = 2 \cdot 1 = 2$
- $3! = 3 \cdot 2 \cdot 1 = 6$
- $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$
- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$
- etc.

Write a function `factorial` that calculates and returns the factorial for a given number (integer). Use a loop for this. Then call the `factorial` function in the main programme and determine the factorials for the values 1 to 5.

f) You want to find out the youngest student in a lecture. On closer inspection, you realise that only the age is of interest, not the name.

Assume that you have the age information in the form of a Python list `age` (think of some arbitrary age information). Use variables, loops and conditional structures to determine the smallest number (age) in the list.

Exercise 4:

– intentionally left blank –

Exercise 5:

a) Write a loop that accumulates the sum of the numbers in a list named `data`.

b) Write a loop that replaces each number in a list named `data` with its absolute value.

c) Define a function named `even`. This function expects a number as an argument and returns `True` if the number is divisible by 2, or it returns `False` otherwise. (Hint: A number is evenly divisible by 2 if the remainder is 0.)

d) Define a function named `summation`. This function expects two numbers, named `low` and `high`, as arguments. The function computes and returns the sum of the numbers between `low` and `high`, inclusive.

e) A group of statisticians at a local college has asked you to create a function that computes the mean of a set of numbers, Define this function in a module named `stats.py`. The function should expect a list of numbers as an argument and return a single number. The function should return 0 if the list is empty. Include a main function that tests the function with a given list `[3, 1, 7, 1, 4, 10]`.

2 NumPy and Pandas

Exercise 6: Import data

a) The following exercises will work with demand series of materials (approx. 1.000 materials). The demand series consist of 32 periods each. Please find this data in your Moodle forum as Excel sheet "material.xlsx" and download it to your VM.

Get ready to work with demand data:

- Find and read about the `read_excel` method on the internet and use it to import the Excel sheet "material.xlsx" into a dataframe.
- Create a new function `load`. The functions transfers the demand data to a Pandas dataframe object with the period (month) as index and the material number as column headers. The dataframe will be returned. *Pay attention to column and index orientation.*
- Transform the row index from string to a date/period format. Look up the `to_period` method to do so.
- Print out the resulting dataframe with the standard print command.

Result:

	1001	1002	1003	1004	1005	1006	...	1993	1994	1995	1996	1997	1998
2017-01	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2017-02	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0
2017-03	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	2.0	0.0	0.0	0.0
2017-04	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2017-05	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2017-06	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2017-07	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	1.0	0.0	0.0
2017-08	1.0	6.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0
2017-09	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

b) Examine the given dataset with the Pandas `describe` method.

Result:

```
            1001       1002       1003  ...       1996       1997       1998
count  32.000000  32.00000  32.000000  ...  32.000000  32.000000  32.000000
mean    0.125000   0.53125   0.062500  ...   0.093750   0.062500   0.125000
std     0.336011   1.50235   0.245935  ...   0.296145   0.245935   0.336011
min     0.000000   0.00000   0.000000  ...   0.000000   0.000000   0.000000
25%     0.000000   0.00000   0.000000  ...   0.000000   0.000000   0.000000
50%     0.000000   0.00000   0.000000  ...   0.000000   0.000000   0.000000
75%     0.000000   0.00000   0.000000  ...   0.000000   0.000000   0.000000
max     1.000000   6.00000   1.000000  ...   1.000000   1.000000   1.000000

[8 rows x 998 columns]
```

c) Extract a Pandas Series object for material ’1500’ and print it to the console. Calculate the sum of the demand series for material ’1057’.

Result:

```
[8 rows x 998 columns]
2017-01    0.0
2017-02    0.0
2017-03    0.0
2017-04    0.0
2017-05    0.0
2017-06    0.0
2017-07    0.0
2017-08    0.0
2017-09    0.0
2017-10    1.0
2017-11    0.0
2017-12    0.0
2018-01    1.0
2018-02    0.0
2018-03    0.0
2018-04    1.0
2018-05    0.0
2018-06    0.0
2018-07    0.0
2018-08    0.0
2018-09    0.0
2018-10    0.0
2018-11    0.0
2018-12    0.0
2019-01    0.0
2019-02    0.0
2019-03    0.0
2019-04    1.0
2019-05    0.0
2019-06    0.0
2019-07    0.0
2019-08    0.0
Name: 1500, dtype: float64
```

d) Extract a Pandas Series object for period '2018-12' and print it to the console. Also try out the "View as DataFrame" and "View as Series"function in PyCharm.

Result:

```
[8 rows x 998 columns]
1001    0.0
1002    0.0
1003    0.0
1004    0.0
1005    0.0
       ...
1994    0.0
1995    1.0
1996    1.0
1997    0.0
1998    0.0
Name: 2018-12, Length: 998, dtype: float64
```

Exercise 7: Moving Average

a) Continue with the demand data as imported before. In order to work with time series data it is helpful to work with the Pandas tools for this purpose. Check out the docs at `https://pandasguide.readthedocs.io/en/latest/Pandas/timeseries.html`. Extend the `load` method from exercise 2 and convert the string index to a date/time format using `pd.to_datetime` method.

b) Implement a `moving_average` method that takes the following parameters:

- the demand dataframe `d`,
- the number `extra_periods` of periods to forecast and,
- the number `n` of periods which are considered in the average calculation

The method shall return:

- The original dataframe `d` with `extra_periods` additional rows with "nan" content and correct index (subsequent periods in YYYY-MM notation, see Pandas tooling for this purpose).
- A forecast dataframe `f` with `extra_periods` rows (same as new `d`) that contains the forecasts as moving average, beginning with period `n+1` until the given number `extra_periods` of forecast periods.
- Only the first extra period is calculated as moving average, further extra periods are just copied from that.
- Pay attention to create and return Dataframes of dtype "float64".

Example: with `extra_periods=3` and `n=4`, the first moving average is calculated for period 5 ("2017-05"), the last for period "2019-09". These forecasts are stored in the dataframe `f`. The new demand dataframe `d` is extended with 3 rows which all contain "nan" and have the indices "2019-09", "2019-10" and "2019-11". Forecasts are only calculated until period "2019-09", for the additional periods "2019-10" and "2019-11" the forecasts will be carried forward unchanged.

Results:

demand

	1001	1002	1003	1004	1005	1006	1007	1008
2018-06	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	3.00000	0.00000
2018-07	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2018-08	0.00000	4.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2018-09	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2018-10	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000
2018-11	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000
2018-12	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2019-01	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2019-02	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2019-03	0.00000	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2019-04	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000
2019-05	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2019-06	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2019-07	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2019-08	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
2019-09	nan	nan	nan	nan	nan	nan	nan	nan
2019-10	nan	nan	nan	nan	nan	nan	nan	nan
2019-11	nan	nan	nan	nan	nan	nan	nan	nan

forecast

	1001	1002	1003	1004	1005	1006	1007	1008
2017-01	nan	nan	nan	nan	nan	nan	nan	nan
2017-02	nan	nan	nan	nan	nan	nan	nan	nan
2017-03	nan	nan	nan	nan	nan	nan	nan	nan
2017-04	nan	nan	nan	nan	nan	nan	nan	nan
2017-05	0.25	0.0	0.0	0.0	0.0	0.0	0.5	0.0
2017-06	0.25	0.0	0.0	0.0	0.0	0.0	0.5	0.0
2017-07	0.25	0.0	0.0	0.0	0.0	0.0	0.5	0.0
2017-08	0.25	0.25	0.0	0.0	0.0	0.0	0.0	0.0
2017-09	0.25	1.75	0.0	0.0	0.0	0.0	0.25	0.0
2017-10	0.25	1.75	0.0	0.0	0.0	0.0	0.25	0.0
2017-11	0.25	1.75	0.0	0.0	0.0	0.0	0.25	0.0
2017-12	0.25	1.5	0.0	0.0	0.0	0.0	1.0	0.0
2018-01	0.0	0.0	0.0	0.25	0.0	0.0	0.75	0.0
2018-02	0.0	0.0	0.0	0.25	0.0	0.0	2.0	0.0
2018-03	0.0	0.0	0.0	0.25	0.0	0.0	2.5	0.0
2018-04	0.0	0.0	0.0	0.25	0.0	0.0	1.75	0.0
2018-05	0.0	0.0	0.0	0.0	0.0	0.0	2.25	0.0
2018-06	0.25	1.25	0.25	0.5	0.25	0.5	2.0	0.25

c) Use the Dataframe `plot()` method to get a visual representation of demand and forecast for material 1057.

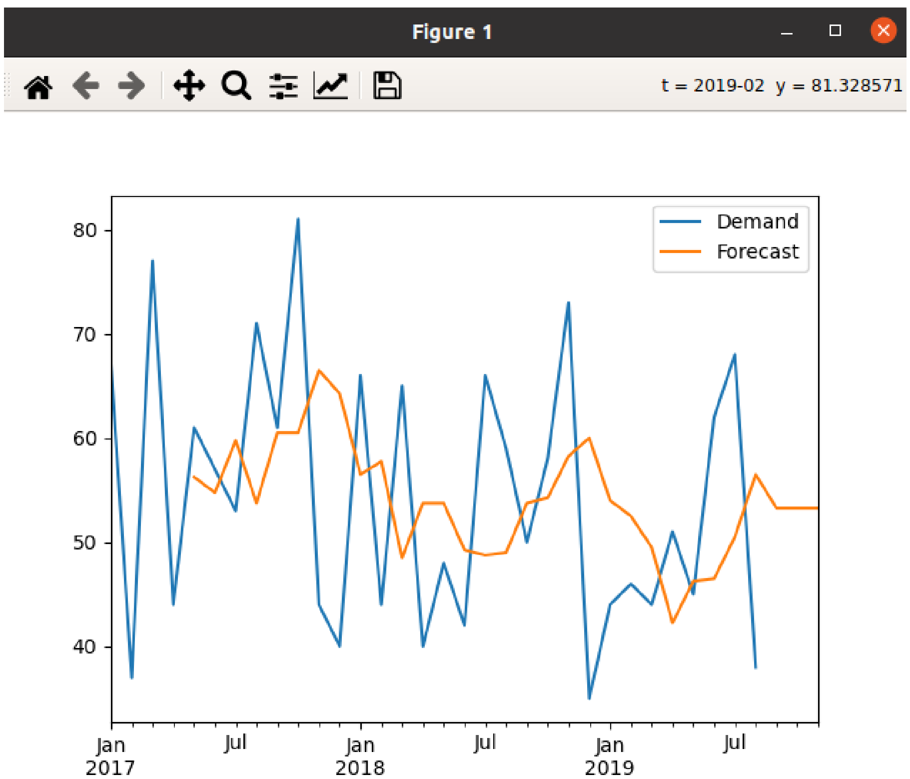
The `plot` method will automatically use the index as x-axis and column values on the y-axis. Just copy the columns for material 1057 to a new dataframe and invoke the `plot` method. Within the PyCharm IDE you need to enable plotting (i. e. opening the plot in a separate view client) by adding to the import statements in the header

```
import matplotlib.pyplot as plt
```

and after calling the `plot()` method:

```
plt.show()
```

Result:



Exercise 8: Forecast Error

a) Implement a method `calculate_error` that takes a demand Dataframe and a forecast Dataframe as input and returns an error Dataframe:

$$e_t = f_t - d_t$$

where

$$e_t \quad : \quad \text{error for period } t$$

$$d_t \quad : \quad \text{demand during period } t$$

$$f_t \quad : \quad \text{forecast for period } t$$

error

	1001	1002	1003	1004	1005	1006	1007	1008
2018-06	0.25	1.25	0.25	0.5	0.25	0.5	-1.0	0.25
2018-07	0.25	1.25	0.25	0.5	0.25	0.5	2.25	0.25
2018-08	0.25	-2.75	0.25	0.5	0.25	0.5	2.25	0.25
2018-09	0.25	2.25	0.25	0.5	0.25	0.5	1.75	0.25
2018-10	0.0	1.0	0.0	0.0	-1.0	0.0	0.75	0.0
2018-11	0.0	1.0	0.0	-1.0	0.25	0.0	0.0	0.0
2018-12	0.0	1.0	0.0	0.25	0.25	0.0	0.0	0.0
2019-01	0.0	0.0	0.0	0.25	0.25	0.0	0.0	0.0
2019-02	0.0	0.0	0.0	0.25	0.25	0.0	0.0	0.0
2019-03	0.0	-1.0	-1.0	0.25	0.0	0.0	0.0	0.0
2019-04	0.0	0.25	0.25	0.0	0.0	-1.0	0.0	0.0
2019-05	0.0	0.25	0.25	0.0	0.0	0.25	0.0	0.0
2019-06	0.0	0.25	0.25	0.0	0.0	0.25	0.0	0.0
2019-07	0.0	0.25	0.25	0.0	0.0	0.25	0.0	0.0
2019-08	-1.0	0.0	0.0	0.0	0.0	0.25	-1.0	-1.0
2019-09	nan	nan	nan	nan	nan	nan	nan	nan
2019-10	nan	nan	nan	nan	nan	nan	nan	nan
2019-11	nan	nan	nan	nan	nan	nan	nan	nan

b) Implement a method `calculate_bias` that takes a demand Dataframe and a forecast Dataframe as input and returns a bias Series:

$$\text{bias} = \frac{1}{n}\sum_{n} e_t$$

where

e_t : error for period t

n : number of historical periods with both forecast and demand

f_t : forecast for period t

Result:

bias

	0
1001	0.00000
1002	0.00000
1003	0.00000
1004	0.00000
1005	0.00000
1006	0.00000
1007	0.01786
1008	-0.03571
1009	-0.01786
1010	-0.02679
1011	0.00000
1012	-0.00893
1013	0.00000
1014	0.00000
1015	0.00000
1016	0.00000
1017	-0.01786
1018	0.00000
1019	-0.03571

c) Implement a method `calculate_mape` that takes a demand Dataframe and a forecast Dataframe as input and returns a MAPE Series:

$$\text{MAPE} = \frac{1}{n}\sum \frac{|e_t|}{d_t}$$

where

e_t : error for period t

n : number of historical periods with both forecast and demand

d_t : demand during period t

Find out the MAPE for material 1133, 1136 and 1139.

Result:

mape

	0
1130	inf
1131	inf
1132	inf
1133	1.02100
1134	inf
1135	inf
1136	0.46598
1137	0.67587
1138	inf
1139	inf
1140	0.68936
1141	inf
1142	inf
1143	inf
1144	inf
1145	0.82961
1146	inf
1147	inf
1148	inf

d) Implement a method `calculate_mae` that takes a demand Dataframe and a forecast Dataframe as input and returns a MAE Series:

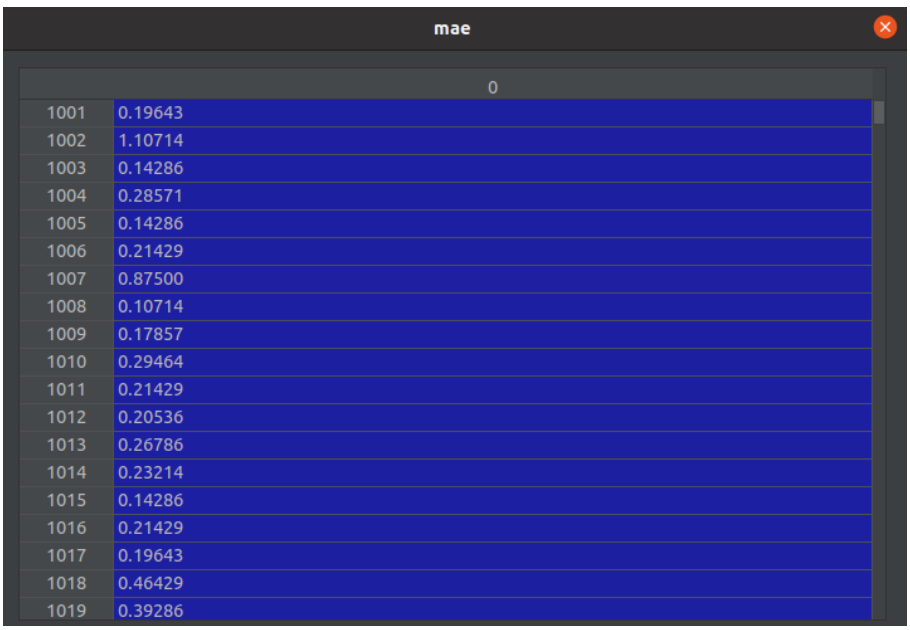
$$\text{MAE} = \frac{1}{n} \sum |e_t|$$

where

e_t : error for period t

n : number of historical periods with both forecast and demand

Result:


mae

	0
1001	0.19643
1002	1.10714
1003	0.14286
1004	0.28571
1005	0.14286
1006	0.21429
1007	0.87500
1008	0.10714
1009	0.17857
1010	0.29464
1011	0.21429
1012	0.20536
1013	0.26786
1014	0.23214
1015	0.14286
1016	0.21429
1017	0.19643
1018	0.46429
1019	0.39286

e) Implement a method `calculate_mse` that takes a demand Dataframe and a forecast Dataframe as input and returns a MSE Series:

$$\text{MSE} = \frac{1}{n}\sum e_t^2$$

where

e_t : error for period t

n : number of historical periods with both forecast and demand

Result:

mse

	0
1001	0.11607
1002	3.23214
1003	0.08929
1004	0.26786
1005	0.08929
1006	0.22321
1007	1.91518
1008	0.08036
1009	0.12500
1010	0.26562
1011	0.22321
1012	0.13170
1013	0.17411
1014	0.16071
1015	0.08929
1016	0.13393
1017	0.21875
1018	0.69643
1019	0.87500

f) Implement a method `calculate_rmse` that takes a demand Dataframe and a forecast Dataframe as input and returns a RMSE Series:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum e_t^2}$$

where

e_t : error for period t

n : number of historical periods with both forecast and demand

Result:

rmse

	0
1001	0.34069
1002	1.79782
1003	0.29881
1004	0.51755
1005	0.29881
1006	0.47246
1007	1.38390
1008	0.28347
1009	0.35355
1010	0.51539
1011	0.47246
1012	0.36290
1013	0.41726
1014	0.40089
1015	0.29881
1016	0.36596
1017	0.46771
1018	0.83452
1019	0.93541

g) Examine bias, MAPE, MAE and RMSE for material 1999 starting with period "2017-08". Assume three forecasts where the first is always 2, the second always 4 and the third always 6 ("always" in terms of "for each period"):

- Implement a method `forecast_error_exercise` that takes the demand Dataframe as input.
- Cut out material 1999 for the mentioned periods.
- Calculate the forecast indicators and store the results in a Dataframe that consists of the three forecasts as columns and the forecast indicators as indices.
- Write down and interpret the overall results. How do them relate to the given forecasts values?

	Forecast 2	Forecast 4	Forecast 6
Bias	-3.92000	-1.92000	0.08000
MAPE	0.64447	1.08893	1.79740
MAE	4.40000	4.08000	4.80000
RMSE	7.11618	6.24179	5.93970

Exercise 9: Exponential Smoothing

a) Implement a `simple_exp_smoothing` method that takes the following parameters:

- the demand dataframe `d`,
- the number `extra_periods` of periods to forecast and,
- the smoothing factor `alpha`

The method shall return:

- The original dataframe `d` with `extra_periods` additional rows with "nan" content and correct index (subsequent periods in YYYY-MM notation, see Pandas tooling for this purpose).
- A forecast dataframe `f` with `extra_periods` rows (same as new `d`) that contains the forecasts as simple exponential smoothing, beginning with period `n+1` until the given number `extra_periods` of forecast periods.
- Only the first extra period is calculated as simple exponential smoothing, further extra periods are just copied from that.
- Pay attention to create and return Dataframes of dtype "float64".

b) Determine the simple exponential smoothing forecasts for materials 1140, 1142 and 1144 in period "2019-09" with $\alpha = 0.1, \alpha = 0.2$ and $\alpha = 0.3$. Which model (i. e. which α) returns the best forecast in terms of RMSE and MAE?

Results:

forecast

	1140	1141	1142	1143	1144	1145	1146	1147
2018-06	2.82689	1.87655	7.82405	1.44916	2.09672	3.49518	3.58956	2.41365
2018-07	2.84420	1.88889	7.64165	1.70424	2.08705	3.44566	3.63060	2.37228
2018-08	2.65978	1.80000	8.47748	1.53382	1.87834	3.50109	3.26754	2.53505
2018-09	2.69380	2.12000	8.62973	1.38044	2.39051	3.25098	3.14079	2.48155
2018-10	2.82442	1.90800	8.86676	1.24239	2.25146	3.22588	3.12671	2.43339
2018-11	2.94198	2.01720	9.18008	1.31815	2.12631	3.10330	3.21404	2.39005
2018-12	3.14778	2.01548	9.16208	1.58634	2.21368	2.99297	3.29263	2.15105
2019-01	3.03300	1.91393	8.74587	1.82770	2.49231	2.79367	3.16337	1.93594
2019-02	3.12970	1.82254	8.37128	1.64493	2.64308	2.71430	3.54703	1.74235
2019-03	3.11673	2.04029	8.33415	1.68044	2.57877	2.64287	3.59233	1.96811
2019-04	3.60506	1.93626	8.90074	1.91240	2.62090	2.77859	3.33310	1.87130
2019-05	3.44455	1.94263	8.71066	2.12116	2.65881	2.70073	3.59979	2.08417
2019-06	3.70010	2.04837	9.03960	2.30904	3.19293	2.63065	3.33981	2.17576
2019-07	3.73009	2.14353	8.83564	2.07814	3.17363	2.56759	3.10583	2.25818
2019-08	3.95708	1.92918	8.45207	2.27032	3.05627	2.51083	3.29524	2.23236
2019-09	3.86137	1.83626	8.30687	2.04329	3.15064	2.35975	2.96572	2.20913
2019-10	3.86137	1.83626	8.30687	2.04329	3.15064	2.35975	2.96572	2.20913
2019-11	3.86137	1.83626	8.30687	2.04329	3.15064	2.35975	2.96572	2.20913

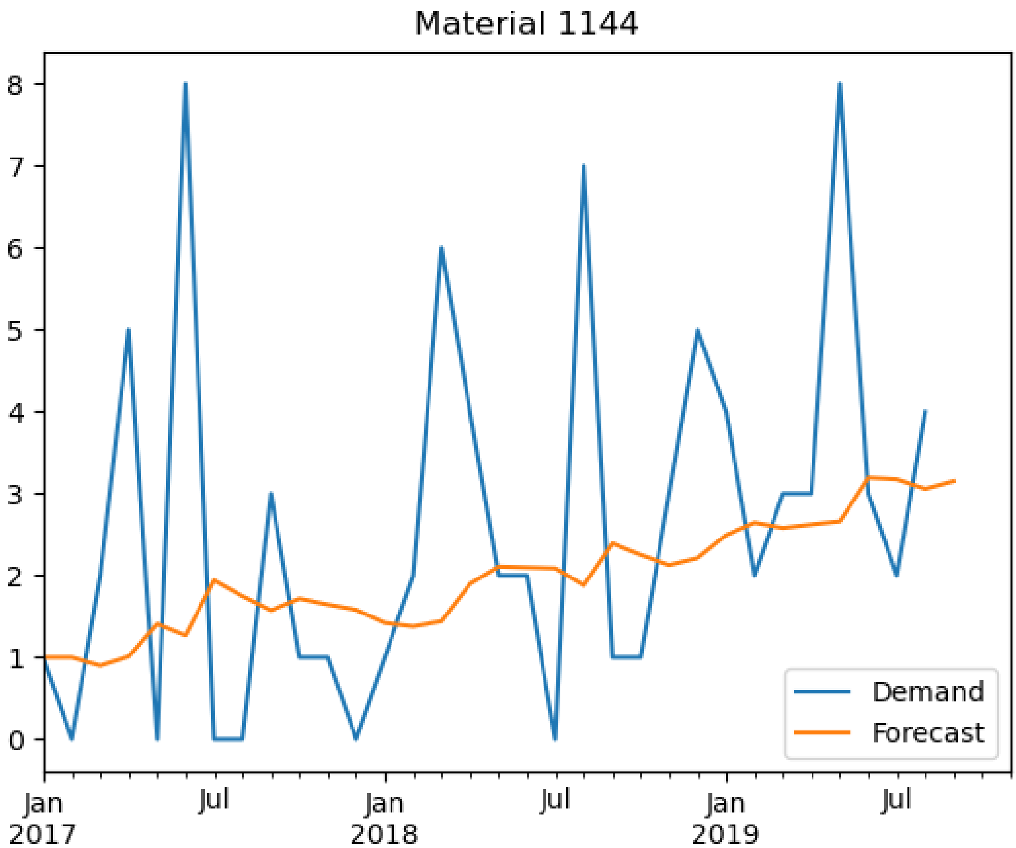
rmse

	1140	1142	1144
0.1	1.92298	4.97130	2.37141
0.2	1.96410	4.95692	2.40759
0.3	2.03269	5.07876	2.48979

mae

	1140	1142	1144
0.1	1.50536	3.76282	1.69707
0.2	1.56962	3.77935	1.80767
0.3	1.63934	3.86146	1.90488

c) Plot the demand and the forecast with simple exponential smoothing for material 1144 with $\alpha = 0.1$ into a graph.



Exercise 10: Double Exponential Smoothing

a) Implement a `double_exp_smoothing` method that takes the following parameters:

- the demand dataframe `d`,
- the number `extra_periods` of periods to forecast,
- the smoothing factor `alpha` for the level and,
- the smoothing factor `beta` for the trend.

The method shall return:

- The original dataframe `d` with `extra_periods` additional rows with "nan" content and correct index (subsequent periods in YYYY-MM notation, see Pandas tooling for this purpose).
- A forecast dataframe `f` with `extra_periods` rows (same as new `d`) that contains the forecasts as double exponential smoothing, beginning with period `n+1` until the given number `extra_periods` of forecast periods.
- Use $a_0 = d_0$ and $b_0 = d_1 - d_0$ as initialization values. *Note: that way f_1 will be perfect; effect decreases for bigger datasets.*
- Pay attention to create and return Dataframes of dtype "float64".

b) Determine the RMSE for materials 1374, 1388 and 1444 for

- double exponential smoothing with $\alpha = 0.1$ and $\beta = 0.1$,
- double exponential smoothing with $\alpha = 0.1$ and $\beta = 0.2$,
- double exponential smoothing with $\alpha = 0.1$ and $\beta = 0.3$,
- simple exponential smoothing with $\alpha = 0.1$.

Split the dataset into a training and a test dataset:

- training data: periods '2017-01' to '2019-05' (29 periods)
- test data: periods '2019-06' to '2019-08' (3 periods)

Calculate the forecasts for three extra periods on the training dataset. Determine the RMSE only for the forecasts of three test data periods ("ex-post").

Discuss and interpret the results.

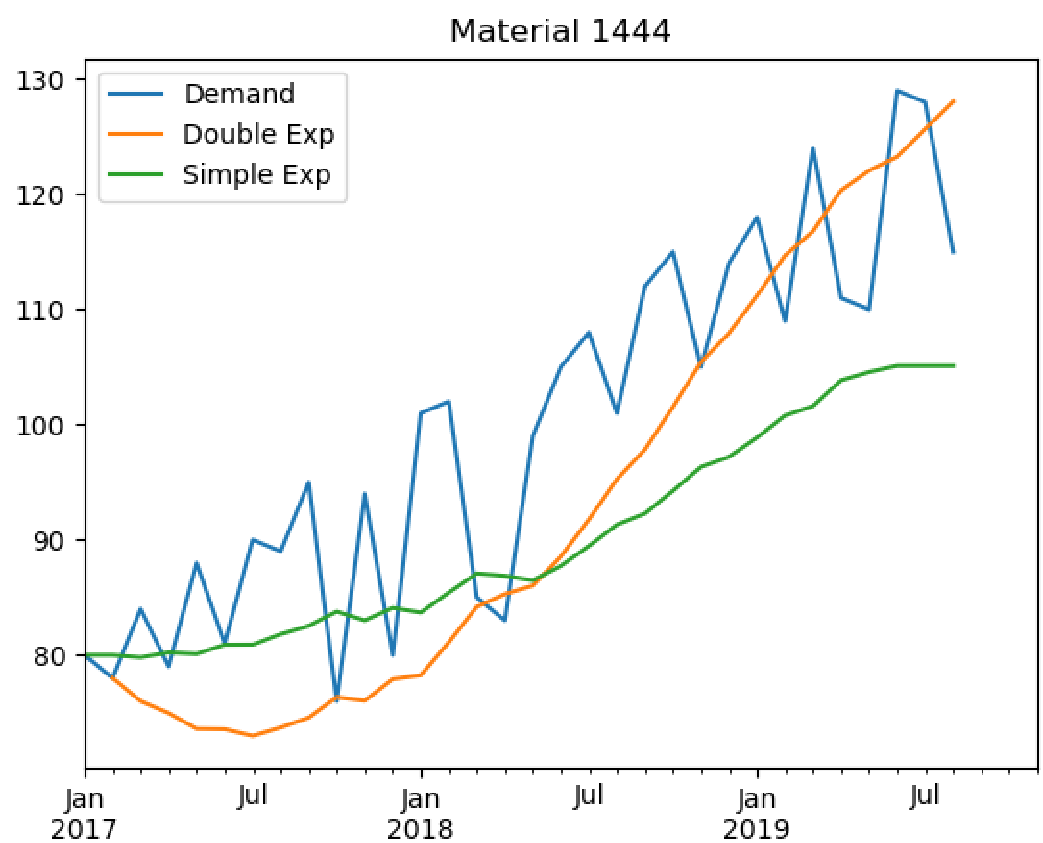
	D0.1/0.1	D0.1/0.2	D0.1/0.3	S0.1
1374	49.26938	30.97138	17.57466	23.34354
1388	196.56269	128.36964	128.76851	47.69437
1444	11.94435	8.34583	8.45498	19.93929

c) Plot for material 1444 the

- demand,
- the forecast with double exponential smoothing with $\alpha = 0.1$ and $\beta = 0.2$ and
- the forecast with (simple) exponential smooting with $\alpha = 0.1$

into a graph.

Discuss and interpret the results.


Material 1444
Demand
Double Exp
Simple Exp
130
120
110
100
90
80
Jan
2017
Jul
Jan
2018
Jul
Jan
2019
Jul

Exercise 11: Model Optimization

a) Implement a function that determines for each material the best `alpha` (Simple Exponential Smoothing) and the best `alpha, beta` combination (Double Exponential Smoothing). Use the MAE as performance indicator.

Use `0.05, 0.1, 0.2, 0.3, 0.4, 0.5` as parameter range for both `alpha` and `beta`. The function shall take an import parameter `expost_periods` that defines the size of the test dataset. Keep in mind not to use the test data for training purposes.

As output the function returns a Dataframe objects with materials as columns and the following indices:

- SimExp_alpha: best alpha value for Simple Exponential Smoothing
- SimExp_MAE: MAE for best alpha value
- DoubleExp_alpha/beta: best alpha, beta value for Double Exponential Smoothing
- DoubleExp_MAE: MAE for best alpha, beta value

Hint: A Dataframe index can be constructed as `MultiIndex` consisting of tuples (here: alpha and beta). Look up Pandas docs.

b) Use again materials 1374, 1388 and 1444. Find out the best parameters using your new function and setting the `expost_periods` to 3 again. Discuss the results.

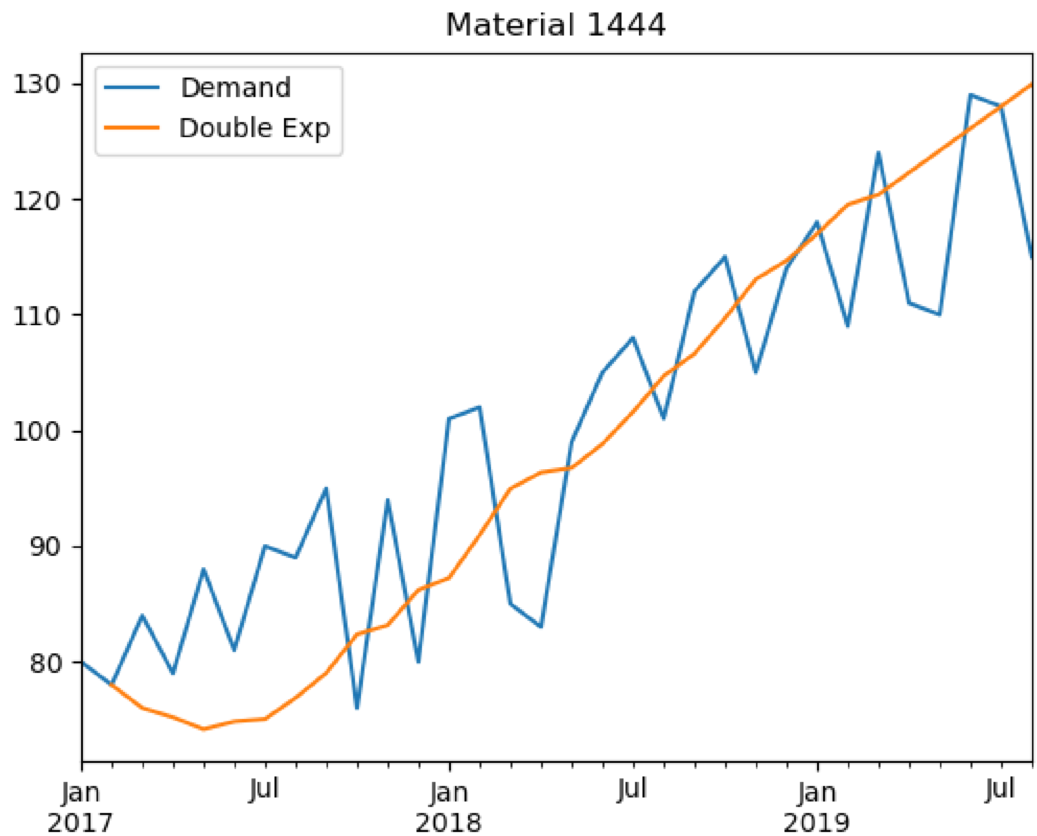
	1374	1388	1444
SimExp_alpha	0.5	0.5	0.4
SimExp_MAE	8.366103337456783	30.80118375768264	11.401801485921837
DoubleExp_alpha/beta	(0.2, 0.3)	(0.1, 0.4)	(0.1, 0.3)
DoubleExp_MAE	5.517155520178801	22.620163808914043	6.200666871796571

c) Rewrite your function (or copy/paste) to use the RMSE as performance indicator. Use again materials 1374, 1388 and 1444. Find out the best parameters using your new function and setting the `expost_periods` to 3 again. Discuss the results.

	1374	1388	1444
SimExp_alpha	0.5	0.5	0.4
SimExp_RMSE	10.83776629809219	35.92147345055881	13.063986519858172
DoubleExp_alpha/beta	(0.1, 0.5)	(0.1, 0.4)	(0.1, 0.5)
DoubleExp_RMSE	7.135777922209676	27.84417955715141	7.015839806131384

d) Repeat the parameter calculation for materials 1374, 1388 and 1444 with RMSE indicator, but set `expost_periods` to 6 periods now. Discuss the results and keep in mind that we already saw that demand for material 1444 obviously has a trend.

   Plot demand and forecast into a graph using Double Exponential Smoothing with `alpha=0.1` and `beta=0.5` and material 1444.


Material 1444
Demand
Double Exp
130
120
110
100
90
80
Jan
2017
Jul
Jan
2018
Jul
Jan
2019
Jul

Exercise 12: Triple Exponential Smoothing (optional)

a) Implement a `triple_exp_smoothing` method that takes the following parameters:

- the demand dataframe `d`,
- the number `extra_periods` of periods to forecast,
- the smoothing factor `alpha` for the level and,
- the smoothing factor `beta` for the trend,
- the smoothing factor `gamma` for the seasonal factors and
- the periodicy as number of distinct season periods `slen`

The method shall return:

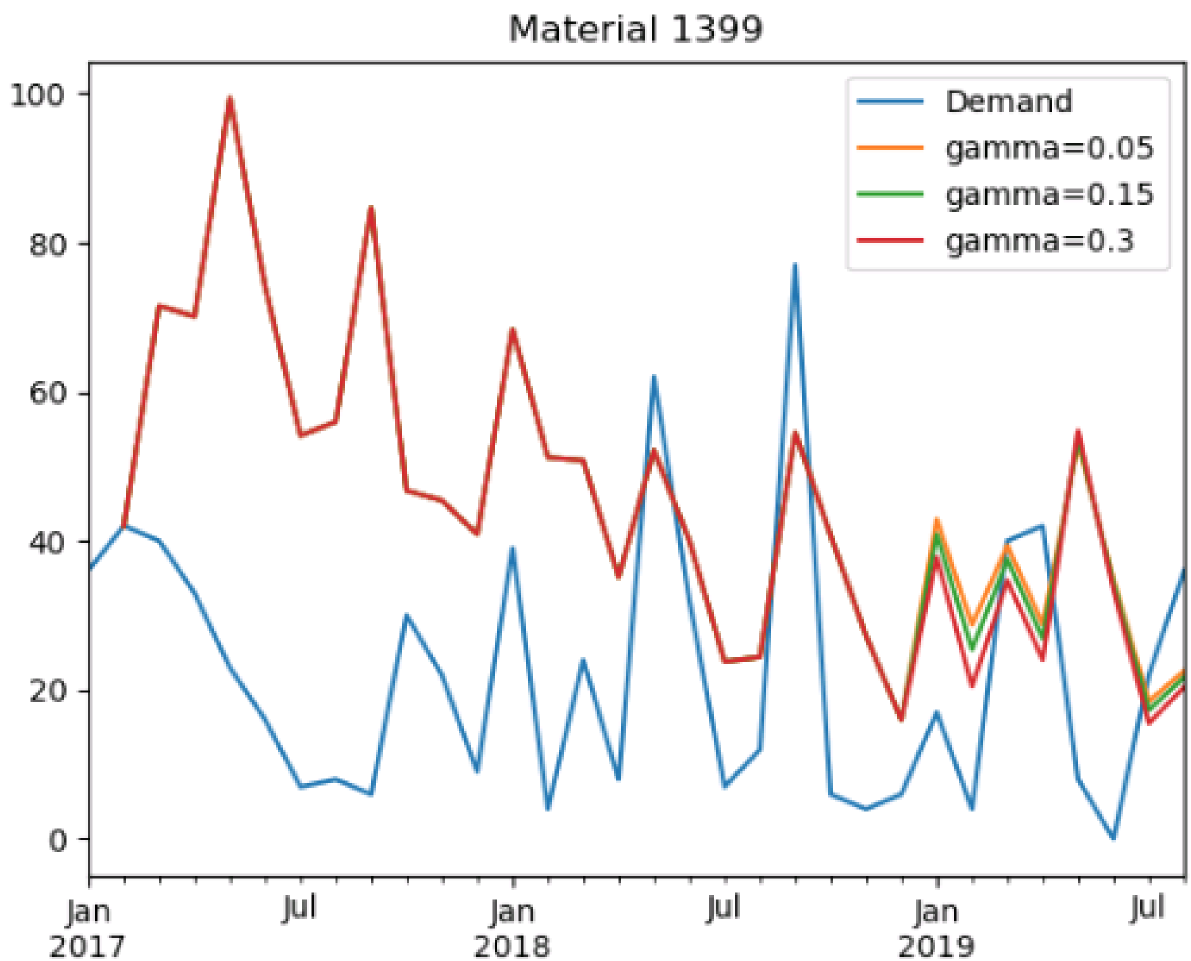
- The original dataframe `d` with `extra_periods` additional rows with "nan" content and correct index (subsequent periods in YYYY-MM notation, see Pandas tooling for this purpose).
- A forecast dataframe `f` with `extra_periods` rows (same as new `d`) that contains the forecasts as triple exponential smoothing, beginning with period `n+1` until the given number `extra_periods` of forecast periods.
- Use $a_0 = d_0 - s_0$ and $b_0 = (d_1 - s_1) - (d_0 - s_0)$ as initialization values.
- Initialize the seasonal factors s with the difference of the overall mean and the seasonal mean.
- Pay attention to create and return Dataframes of dtype "float64".

b) Play with the `triple_exp_smoothing` function: Use material 1399 and calculate the RMSE for $\gamma \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. Use $\alpha = 0.3$ and $\beta = 0.1$. Split the dataset into two full years (i. e. '2017-01' to '2018-12') for training and '2019-01' to '2019-08' for testing.

Which `gamma` performs best in terms of RMSE?

	0.05	0.1	0.15	0.2	0.25	0.3
RMSE	24.72453	24.53454	24.37538	24.24767	24.15190	24.08845

c) Use the parameters as before (material 1399, $\alpha = 0.3$ and $\beta = 0.1$). Plot the demand and the forecasts for $\gamma \in \{0.05, 0.15, 0.3\}$ into a graph and discuss the results.


Material 1399
Demand
gamma=0.05
gamma=0.15
gamma=0.3
100
80
60
40
20
0
Jan
2017
Jul
Jan
2018
Jul
Jan
2019
Jul