# Pandas

Exercises

Brights

# Import the csv-file

Import the csv-file people.csv into a dataframe.

This will be the base for the rest of these exercises, so start each exercise fresh from this step.

Tips: read_csv

# Remove index

When the file is imported from csv a index-column is created.

Remove this index.

Tips:

set_index

drop

# Sort

Sort the dataframe by name, first ascending and then descending

Tips:

sort_values

ascending

Brights

# Find Abigail

Find all rows that has Abigail as name.

Tips:

boolean indexing

# Remove Abigail

Find all rows that has Abigail as name and remove them.

Reset the index after removal

Tips:

- boolean indexing using != will create a dataframe without Abigail
- reset_index(drop=True)

# Find all Abigails that are 35

Find all rows that has Abigail as name and 35 as age.

Tips:

- You can combine two expressions with &

# Remove duplicates

Remove all duplicates from the dataframe and reset index

Tips:

- drop_duplicates

# Print unique names

Several of the names occures many times in the dataframe. Print each name only once.

Tips:

- unique

Brights

# Print column headers

Print all headers of the dataframe

Tips:

- columns

Brights

# People of Austin

Find all rows where city is Austin and age is greater than 30

Tips:

See ”Find Abigails…”

# Average age

Find the average age of people living in Austin. Display with two decimals

Tips:

- mean

- f-string formatting

# Median age

Find the median age of all people in dataframe

Tips:

- median

# Add Salary

Generate random salaries between 20, 000 and 35, 000 for each row and at it as column with the header ’Salary’

Tips:

- randint

- you can add a column to a dataframe just like you add a value to a dictionary[key] = 10

Brights

# Count and display

Count and display how many people live in each city. Display City names and number of people.

Tips:

- groupby
- count

Brights

# Calculate and display

Make a dataframe that holds all city-names, how many people live in each city and the average age for each city (total of 3 columns)

Tips:

- groupby
- count
- mean
- Do this in several steps

# Export to json

Export your dataframe to a json-file, test both the lines=True and lines=False option and open the resulting json-file to view the difference

Tips:

- to_json

# import from json and combine

import the file ”people.json” into another dataframe (so you have two).

Combine your two dataframes into one dataframe.

Tips:

- concat
- ignore_index

# Change space to -

Change all spaces ” ” in City names to a -, so ”New York” becomes ”New-York” for instance.

Tips:

- replace

Brights

# Merge

Merge the following dataframe with your dataframe:

```
salary_df = pd.DataFrame({
    'Salary' : [50000, 60000, 70000, 80000]
}, index=[0,1,2,3])
```

Use index from both dataframes to merge, and all values from your original dataframe. Since salary_df only contains 4 values, we will have a lot of NaN.

Tips:

- merge

- left_index, right_index, how