Week 4 - Data organising with Pandas

Author: Johanne Sejrskild
Date: 22.09.2025

Good afternoon
Today we are going to work with loops, condions and using ´pandas´to manipulate data. The green excercises will be highly linked to what you livecoded with Anna. If you find them challenging use yesterdays work as a help or ask. If you want to challenge yourself, try and do them all without using any help. In the yellow excercises we will do some data manipulation challenges using pandas. And we will skip the red tasks today as we have a lot on the program

Structure of the notebook:
Green excercises

- Data wrangling on the iris dataset

Yellow excercises

- Music sales challenge
- Space mission challenge
- Supervillan challenge

Start with the first excercise, and then continue in order. Feel free to work together, and see how far you can get.
The important thing is to learn, not to solve all the challenges!

In [ ]:
```
%pip install lxml
#%pip install sklearn
```

```
Collecting lxml
  Downloading lxml-6.0.2-cp312-cp312-macosx_10_13_universal2.whl.metadata
(3.6 kB)
Downloading lxml-6.0.2-cp312-cp312-macosx_10_13_universal2.whl (8.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 22.4 MB/s eta 0:00:00
00:0100:01
Installing collected packages: lxml
Successfully installed lxml-6.0.2

[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python3 -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
ERROR: Could not find a version that satisfies the requirement xelatex (from
versions: none)
ERROR: No matching distribution found for xelatex

[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python3 -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

```
# Before we start we need to import the necessary packages
import lxml
import numpy as np
import pandas as pd
import requests # We might need this package to get some data from the web
from sklearn import datasets
```

Green excercises

Data organisation using a dataset about flowers

In [ ]:

```
flower = datasets.load_iris()

# convert to DataFrame
df = pd.DataFrame(flower.data, columns=flower.feature_names)

df.head()
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Lets take a look at the data frame

In [ ]:

```
# There are some commands in the library pandas that can give you a quick ov
```

```python
df.head()      # first 5 rows, if you put a number into the paranthesis you
df.tail()      # last 5 rows
df.info()      # summary of columns and types
df.describe()  # quick statistics (for numbers)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Selecting columns and rows

Try to run the cell below and figure out which output is linked to the code

In [ ]:
```python
# If you want to select a specific column you can select it using the name:
print(df['sepal length (cm)'].head())

# If you would like to print one row, you can use the index of the row:
print(df.iloc[0])

# if you want to select a few rows of only a few columns you can also use in
print(df.iloc[0:3 , 0:2])  # first three rows, first two columns

# And if you want to select specific data, you can specify a single row and
print(df.iloc[2,0])  # second row, first column

# Or use the column name:
print(df.loc[2, "sepal length (cm)"]  )
```

```
0    5.1
1    4.9
2    4.7
3    4.6
4    5.0
Name: sepal length (cm), dtype: float64
sepal length (cm)    5.1
sepal width (cm)     3.5
petal length (cm)    1.4
petal width (cm)     0.2
Name: 0, dtype: float64
   sepal length (cm)  sepal width (cm)
0                5.1               3.5
1                4.9               3.0
2                4.7               3.2
4.7
4.7
```

Subsetting data

Subsetting is the process of retrieving just the parts of large files which are of interest for a specific purpose.

This will come in handy for your projects when you have to work with potentially large data files

In [ ]:
```
# Let's try to select some data using conditionals

# Here we select all rows where the sepal length is larger than or equal to
lengt_above_five = df[df["sepal length (cm)"] >= 5]
lengt_above_five.head()
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
7	5.0	3.4	1.5	0.2
10	5.4	3.7	1.5	0.2

In [ ]:
```
# Here we select all rows where the sepal length is larger than or equal to
length_and_width = df[(df["sepal length (cm)"] >= 5) & (df["sepal width (cm)
length_and_width.head()
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
53	5.5	2.3	4.0	1.3
60	5.0	2.0	3.5	1.0
62	6.0	2.2	4.0	1.0
68	6.2	2.2	4.5	1.5
80	5.5	2.4	3.8	1.1

In [ ]:
```
# Excercise - Find the longest petal length and the median petal length
# and subset the flowers that are between the median and one centimeter shor
max_petal_length = df["petal length (cm)"].max()
median_petal_length = df["petal length (cm)"].median()

subset_flowers = df[(df["petal length (cm)"] >= median_petal_length) & (df["
```

Sorting data

We can choose to sort our data in order of something of interest.

In [ ]:
```
# we could sort the data by a specific column in both ascending and descendi
df_sorted = df.sort_values(by="sepal length (cm)", ascending=False) # change
df_sorted.head()
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
131	7.9	3.8	6.4	2.0
135	7.7	3.0	6.1	2.3
122	7.7	2.8	6.7	2.0
117	7.7	3.8	6.7	2.2
118	7.7	2.6	6.9	2.3

In [ ]:
```
# Excercise - sort the data by petal width in ascending order and select the
df_sorted = df.sort_values(by="sepal width (cm)", ascending=True) # change a
smallest_petal_width = df_sorted.head(10)
smallest_petal_width
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
60	5.0	2.0	3.5	1.0
62	6.0	2.2	4.0	1.0
119	6.0	2.2	5.0	1.5
68	6.2	2.2	4.5	1.5
41	4.5	2.3	1.3	0.3
53	5.5	2.3	4.0	1.3
93	5.0	2.3	3.3	1.0
87	6.3	2.3	4.4	1.3
81	5.5	2.4	3.7	1.0
80	5.5	2.4	3.8	1.1

Flipping
Should you work with time seires data and would like to mirror (flip) your data, you can do this using pandas

In [ ]:
```
print(df.head(5))

reversed_df = df.iloc[::-1]   # Flipping the dataframe horisontally (reverse

print(reversed_df.head(5))
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (c
m)
149                5.9               3.0                5.1
1.8
148                6.2               3.4                5.4
2.3
147                6.5               3.0                5.2
2.0
146                6.3               2.5                5.0
1.9
145                6.7               3.0                5.2
2.3
```

Joining Sometimes we have multiple dataframes we woudl like to add together. Maybe you have been subsetting parts of an old dataframe to substract important information and would now like join them so you can begin your analysis.

In [ ]:
```
# Joining a bit of the iris data with a new dataframe (we will make up some
first_10 = df.iloc[0:10, :]  # selecting the first 10 rows of the iris data
new_data = {"color": ["red", "blue", "green", "yellow", "purple", "red", "bl
            "height": [80, 80, 70, 100, 90, 80, 80, 70, 100, 90]}
# Right now new_df is a dictionary, we need to convert it to a dataframe
new_df = pd.DataFrame(new_data)

#Now we join the two dataframes
joined = first_10.join(new_df, how='left') # There are 4 different types of

joined
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	color	height
0	5.1	3.5	1.4	0.2	red	80
1	4.9	3.0	1.4	0.2	blue	80
2	4.7	3.2	1.3	0.2	green	70
3	4.6	3.1	1.5	0.2	yellow	100
4	5.0	3.6	1.4	0.2	purple	90
5	5.4	3.9	1.7	0.4	red	80
6	4.6	3.4	1.4	0.3	blue	80
7	5.0	3.4	1.5	0.2	green	70
8	4.4	2.9	1.4	0.2	yellow	100
9	4.9	3.1	1.5	0.1	purple	90

Different types of how to join two data frames
This is important if your dataframes do not have the same amount of rows

left → all rows from the left DataFrame (default).

right → all rows from the right DataFrame.

inner → only rows with matching index values in both.

outer → all rows from both, fill missing with NaN.

In [ ]:

```
# Excercise - Which types of join (the 'how=') will work in the example abov
joined_left = first_10.join(new_df, how='left')
joined_right = first_10.join(new_df, how='right') # last row = NaN
joined_inner = first_10.join(new_df, how='inner')
joined_outer = first_10.join(new_df, how='outer') # last row = NaN
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	color	height
0	5.1	3.5	1.4	0.2	red	80
1	4.9	3.0	1.4	0.2	blue	80
2	4.7	3.2	1.3	0.2	green	70
3	4.6	3.1	1.5	0.2	yellow	100
4	5.0	3.6	1.4	0.2	purple	90
5	5.4	3.9	1.7	0.4	red	80
6	4.6	3.4	1.4	0.3	blue	80
7	5.0	3.4	1.5	0.2	green	70
8	4.4	2.9	1.4	0.2	yellow	100
9	4.9	3.1	1.5	0.1	purple	90
10	NaN	NaN	NaN	NaN	orange	85

In [ ]:

```
# Excercise 2 - Add a row to one of the dataframes and see what happens when
new_row = pd.DataFrame({"color": ["orange"], "height": [85]})
new_df = pd.concat([new_df, new_row], ignore_index=True)

joined = first_10.join(new_df, how='right')
joined
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	color	height
0	5.1	3.5	1.4	0.2	red	80
1	4.9	3.0	1.4	0.2	blue	80
2	4.7	3.2	1.3	0.2	green	70
3	4.6	3.1	1.5	0.2	yellow	100
4	5.0	3.6	1.4	0.2	purple	90
5	5.4	3.9	1.7	0.4	red	80
6	4.6	3.4	1.4	0.3	blue	80
7	5.0	3.4	1.5	0.2	green	70
8	4.4	2.9	1.4	0.2	yellow	100
9	4.9	3.1	1.5	0.1	purple	90
10	NaN	NaN	NaN	NaN	orange	85

Concatenating

You can also join two dataframes bu simply gluing them together.

In [ ]:
```
# We just made a subset of the original dataframe called 'first_10' now we f
last_10 = df.iloc[-10:, :]   # selecting the last 10 rows using one of the m


# Now we concatenate the two dataframes together
concatenated = pd.concat( [first_10, last_10], axis=0)  # axis=0 means we co
concatenated
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

Now you have played around with some of the basics manipulation in pandas! Now lets jump into some challenges

Yellow excercises

OBS: To ensure you can go back in 3 months time and read you code and understand the logics behind it it needs to be well commented.
So, while you solve the yellow excercises ensure that you add some meaningful

comments about the logics and coding choices.

:))

The Yellow excercises is borrowed from last years couse and written by Ethan Weed

Music sales challenge

Write a script that:

1. Combines the tables of best-selling physical singles and best-selling digital singles on the Wikipedia page "List_of_best-selling_singles"
2. Outputs the artist and single name for the year you were born. If there is no entry for that year, take the closest year after you were born.
3. Outputs the artist and single name for the year you were 15 years old.

In [ ]:

```
# Starter code
#musicdata = pd.read_html("https://en.wikipedia.org/wiki/List_of_best-sellin
url_music = "https://en.wikipedia.org/wiki/List_of_best-selling_singles"

# Add a User-Agent header so Wikipedia doesn't block it
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"}
response = requests.get(url_music, headers=headers)

# Pass the HTML text to pandas
musicdata = pd.read_html(response.text)


#Extracting physical and digital singles from the musicdata
physical_singles = musicdata[0]
digital_singles = musicdata[3]

physical_singles['Type'] = 'Physical'
digital_singles['Type'] = 'Digital'

# Combining the two tables
combined_singles = pd.concat([physical_singles, digital_singles])
combined_singles.head()
```

```
/var/folders/3x/d81s3p9d121_p6l9czkfl60r0000gp/T/ipykernel_9769/1297234199.p
y:10: FutureWarning: Passing literal html to 'read_html' is deprecated and w
ill be removed in a future version. To read from a literal string, wrap it i
n a 'StringIO' object.
  musicdata = pd.read_html(response.text)
```

Out[ ]:

	Artist	Single	Released	Sales (in millions)	Source	Type
0	Bing Crosby	"White Christmas"	1942	50	[1]	Physical
1	Elton John	"Something About the Way You Look Tonight"/"Ca...	1997	33	[1]	Physical
2	Bing Crosby	"Silent Night"	1935	30	[2]	Physical
3	Tino Rossi	"Petit Papa Noël"	1946	30	[3]	Physical
4	Bill Haley & His Comets	"Rock Around the Clock"	1954	25	[4][5]	Physical

In [ ]:

```
# Print the arrtist and single from the year you were 15 years old.
subset_singles = combined_singles[combined_singles['Released'] == 2018]  # I
subset_singles[['Artist', 'Single']]
```

Out[ ]:

	Artist	Single
13	Lil Nas X featuring Billy Ray Cyrus	"Old Town Road"
20	Drake	"God's Plan"

Space challenge

1. Make a single dataframe that combines the space missions from the 1950's to the 2020's
2. Write a script that returns the year with the most launches
3. Write a script that returns the most common month for launches
4. Write a script that ranks the months from most launches to fewest launches

In [ ]:

```
# Starter code.
url_space =  "https://en.wikipedia.org/wiki/Timeline_of_Solar_System_explora

# Add a User-Agent header so Wikipedia doesn't block it
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"}
response = requests.get(url_space, headers=headers)

# Pass the HTML text to pandas
spacedata = pd.read_html(response.text)

# combine all tables into data frame
combined_space = pd.concat(spacedata, ignore_index = True)

# Dropping column we dont need
combined_space = combined_space.iloc[:, 0:3]
combined_space.head()
```

```
/var/folders/3x/d81s3p9d121_p6l9czkfl60r0000gp/T/ipykernel_9769/4030390085.p
y:9: FutureWarning: Passing literal html to 'read_html' is deprecated and wi
ll be removed in a future version. To read from a literal string, wrap it in
a 'StringIO' object.
  spacedata = pd.read_html(response.text)
```

Out[ ]:

	Mission name	Launch date	Description
0	Sputnik 1	4 October 1957	First Earth orbiter
1	Sputnik 2	3 November 1957	Earth orbiter, first animal in orbit, a dog na...
2	Explorer 1	1 February 1958	Earth orbiter; discovered Van Allen radiation ...
3	Vanguard 1	17 March 1958	Earth orbiter; oldest spacecraft still in Eart...
4	Luna 1	2 January 1959	First lunar flyby (attempted lunar impact?); f...

In [ ]:

```
## The year with the most launches
# Split Launch date into day, month, year
launch_dates = combined_space['Launch date'].str.split(' ', expand=True)
launch_dates.columns = ['Day', 'Month', 'Year']

# Add the year column to the original dataframe
combined_space = pd.concat([combined_space, launch_dates['Year']], axis=1)
max_launches_year = combined_space['Year'].value_counts().idxmax()
max_launches_count = combined_space['Year'].value_counts().max()
print(f"The year with the most launches is {max_launches_year} with {max_lau
```

```
The year with the most launches is 1965 with 12 launches.
```

In [ ]:

```
# The month with the most launches
# Add the month column to OG df
combined_space = pd.concat([combined_space, launch_dates['Month']], axis=1)
max_launches_month = combined_space['Month'].value_counts().idxmax()
max_launches_month_count = combined_space['Month'].value_counts().max()
print(f"The most common launch month is {max_launches_month} with {max_launc
```

```
The month with the most launches is November with 30 launches.
```

In [ ]:

```
# Ranking of months with the most to the fewest launches
month_ranking = combined_space['Month'].value_counts()
print("Ranking of months by number of launches:")
print(month_ranking)
```

```
Ranking of months by number of launches:
Month
November     30
August       27
September    25
October      24
January      21
July         21
December     19
February     18
May          18
March        15
June         14
April        13
Name: count, dtype: int64
```

Supervillain challenge

1. Write a script that combines the tables showing supervillain debuts from the 30's through the 2010's
2. Write a script that ranks each decade in terms of how many supervillains debuted in that decade
3. Write a script that ranks the different comics companies in terms of how many supervillains they have, and display the results in a nice table (pandas dataframe)

In [ ]:
```
# Starter code.
url_space =  "https://en.wikipedia.org/wiki/Timeline_of_Solar_System_explora

# Add a User-Agent header so Wikipedia doesn't block it
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"}
response = requests.get(url_space, headers=headers)

# Pass the HTML text to pandas
spacedata = pd.read_html(response.text)

# combine all tables into data frame
combined_space = pd.concat(spacedata, ignore_index = True)

# Dropping column we dont need
combined_space = combined_space.iloc[:, 0:3]
combined_space.head()
```

In [ ]:
```
#supervillandata = pd.read_html("https://en.wikipedia.org/wiki/List_of_comic

url_villan = "https://en.wikipedia.org/wiki/List_of_comic_book_supervillain_

# Add a User-Agent header so Wikipedia doesn't block it
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"}
response = requests.get(url_villan, headers=headers)

# Pass the HTML text to pandas
supervillandata = pd.read_html(response.text)
```

```
df_supervillan = pd.concat(supervillandata, ignore_index = True)

# combine all tables into data frame
df_supervillan = df_supervillan.iloc[5 :, 2:]  # Dropping rows and columns t
df_supervillan.head()
```

```
/var/folders/3x/d81s3p9d121_p6l9czkfl60r0000gp/T/ipykernel_9769/1210619977.p
y:10: FutureWarning: Passing literal html to 'read_html' is deprecated and w
ill be removed in a future version. To read from a literal string, wrap it i
n a 'StringIO' object.
  supervillandata = pd.read_html(response.text)
```

Out[ ]:

	Character / Team	Year Debuted	Company	Creator/s	First Appearance
5	Ultra-Humanite	1939 (June)	DC	Jerry Siegel, Joe Shuster	Action Comics (vol. 1) #13
6	Dr. Death	1939 (July)	DC	Bob Kane, Bill Finger	Detective Comics (vol. 1) #29
7	The Monk	1939 (September)	DC	Bob Kane, Bill Finger	Detective Comics (vol. 1) #31
8	The Claw	1939 (December)	Lev Gleason Publications	Jack Cole	Silver Streak Comics #1
9	Hath-Set	1940 (January)	DC	Gardner Fox, Dennis Neville	Flash Comics #1

In [ ]:
```
# 1. Write a script that combines the tables showing supervillain debuts fro
# I just did above?
```

In [ ]:
```
# 2. Write a script that ranks each decade in terms of how many supervillain
# Remove non-numbers for Year Debuted column
df_supervillan = df_supervillan[df_supervillan['Year Debuted'].apply(lambda

# Remove last digit of each year to get the decade
df_supervillan['Decade'] = df_supervillan['Year Debuted'].astype(str).str[:-
decade_ranking = df_supervillan['Decade'].value_counts()
print("Ranking of decades by number of supervillain debuts:")
print(decade_ranking)
```

```
Ranking of decades by number of supervillain debuts:
Decade
1970s    79
1960s    78
1990s    74
1980s    71
2000s    40
2010s     5
1950s     4
1940s     2
Name: count, dtype: int64
```

In [ ]:
```
# 3. Write a script that ranks the different comics companies in terms of ho
company_ranking = df_supervillan['Company'].value_counts()
print("Ranking of comic companies by number of supervillains:")
print(company_ranking)
```

```
Ranking of comic companies by number of supervillains:
Company
DC                   197
Marvel               130
Image                  5
Dark Horse             5
Disney/Hyperion        4
Marvel/Timely          3
Eternity               3
Fawcett Comics/DC      1
Comico                 1
Image Comics           1
Name: count, dtype: int64
```