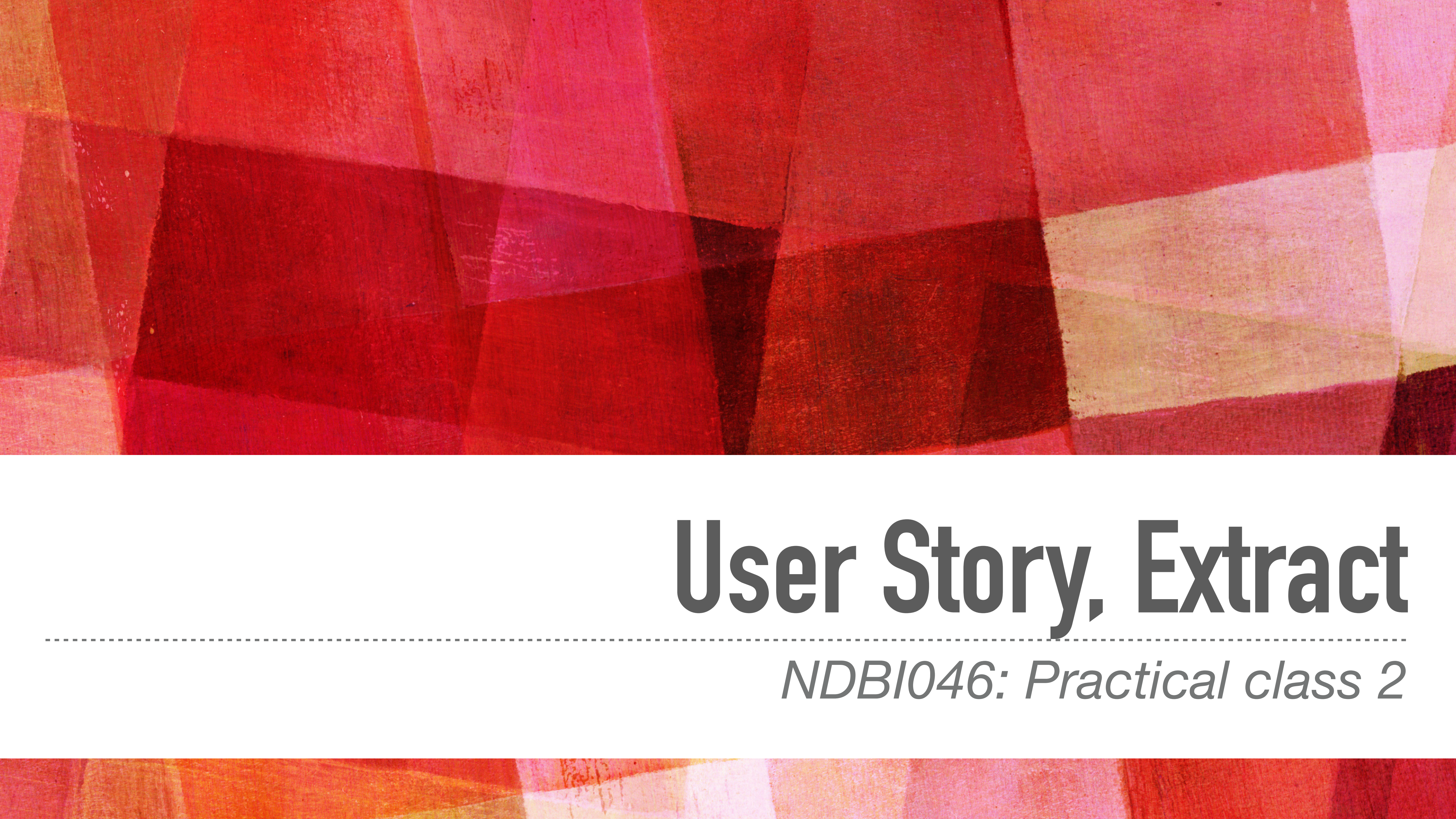




User Story, Extract

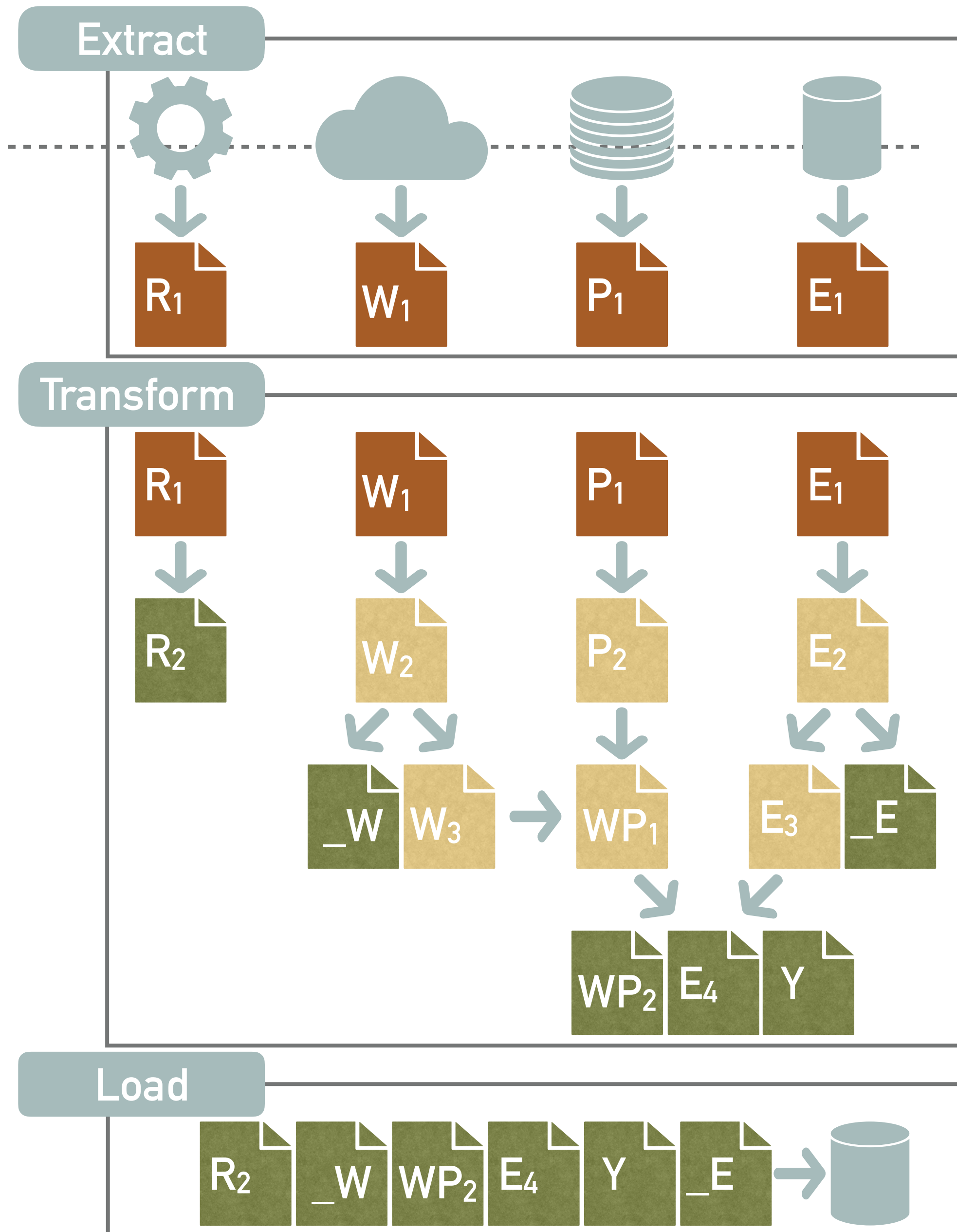
NDBI046: Practical class 2

User Story

- ❖ The data analysts in our company asked us to prepare waste management datasets and load them to the data warehouse so that they could perform the required analyses
 - ❖ Specifically, they need datasets related to waste management in the Czech Republic at the level of individual regions and the extent of funds used to mitigate the environmental burden due to waste management
 - ❖ Their aim is to assess municipal and corporate waste production and address impacts, and they require that the amount of waste can be calculated per capita or per unit area
 - ❖ As a source of data, we are to use open data on waste from Czech open data portal, data published by the Czech Statistical Office and generally known facts can be extracted from Wikipedia
- ❖ **Data Engineer roles:**
 - ❖ *Extract datasets* from (various) sources
 - ❖ *Transform data* into a uniform form, detect and correct inconsistencies, etc. (see later)
 - ❖ *Load* the *data to* a *data warehouse* so that analysts can perform analyses

Extract Transform Load (ETL)

- ❖ **Data extraction** involves extracting data from homogeneous or heterogeneous sources
 - ❖ **various data sources**, e.g., *local files*, *data catalogs*, *databases*, websites, *web services*, sensors (data are all around us)
 - ❖ **various methods**, e.g., *change data capture* (CDC), *full* vs *incremental extraction*, *query-based extraction*, *web scrapping*, *event monitoring*
 - ❖ **various data formats**, e.g., Comma-Separated Values (*CSV*), JavaScript Object Notation (*JSON*), eXtensible Markup Language (*XML*), Excel Spreadsheet XML (*XLSX*), Resource Description Framework (*RDF*)
- ❖ **Data transformation** processes clean and transform data into a suitable format/structure for querying and analysis
- ❖ **Data loading** involves the insertion of data into, e.g., operational data store, data warehouse, data lake, or data mart



Prerequisite: Setting up Python (Linux, macOS)

- ❖ *Check* which *version of Python* is installed (if any)
 - ❖ If Python 3 is not installed, download the (latest) version of Python^{#1} and follow the installation
- ❖ Once installed, *create* any *folder for your NDBI046 project* and navigate to it, e.g., ~/Projects/python-ndbi046
- ❖ *Create* your *Python environment*, e.g., ndbi046_env
- ❖ *Activate* you Python environment
- ❖ *Install* the required *packages*
 - ❖ Download the requirements.txt file from the practical class website
 - ❖ You may also *install additional packages*
 - ❖ You may always export the list of installed packages to a file
- ❖ Exit the Python environment after completing the practical class (not before)
 - ❖ You can return to the environment at any time by activating it

#1 <https://www.python.org/downloads/>

```
1 % python3 --version
2
3 % mkdir ~/Projects/python-ndbi046
4 % cd ~/Projects/python-ndbi046
5
6 % python3 -m venv ndbi046_env
7
8 % source ndbi046_env/bin/activate
9
10 (ndbi046_env) % pip install -r requirements.txt
11
12 (ndbi046_env) % pip install pandas
13
14 (ndbi046_env) % pip freeze > requirements.txt
15
16 (ndbi046_env) % deactivate
17
18 % cat requirements.txt
```

checking the
installed version

creating
and activating a virtual
environment

installation
of the required
packages

export of
installed packages

list installed
packages

deactivating a
virtual environment

Prerequisite: Setting up Python (Windows)

- ❖ *Check* which *version of Python* is installed (if any)
 - ❖ If Python 3 is not installed, download the (latest) version of Python^{#1} and follow the installation
- ❖ Once installed, *create* any *folder for your NDBI046 project* and navigate to it, e.g., C:\Projects\python-ndbi046
- ❖ *Create* your *Python environment*, e.g., ndbi046_env
- ❖ *Activate* you Python environment
- ❖ *Install* the required *packages*
 - ❖ Download the requirements.txt file from the practical class website
 - ❖ You may also *install additional packages*
 - ❖ You may always export the list of installed packages to a file
- ❖ Exit the Python environment after completing the practical class (not before)
 - ❖ You can return to the environment at any time by activating it

^{#1} <https://www.python.org/downloads/>

```
1 > python3 --version
2
3 > mkdir C:\Projects\python-ndbi046
4 > cd C:\Projects\python-ndbi046
5
6 > python3 -m venv ndbi046_env
7
8 > ndbi046_env\Scripts\activate.bat
9
10 (ndbi046_env) > pip install -r requirements.txt
11
12 (ndbi046_env) > pip install pandas
13
14 (ndbi046_env) > pip freeze > requirements.txt
15
16 (ndbi046_env) > deactivate
17
18 > type requirements.txt
```

checking the
installed version

creating
and activating a virtual
environment

installation
of the required
packages

export of
installed packages

list installed
packages

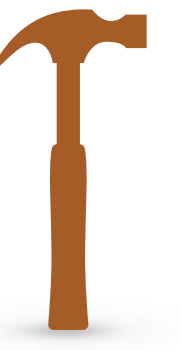
deactivating a
virtual environment

Example 2.1: Extract dataset 'Production of industrial and municipal waste' (CSV)

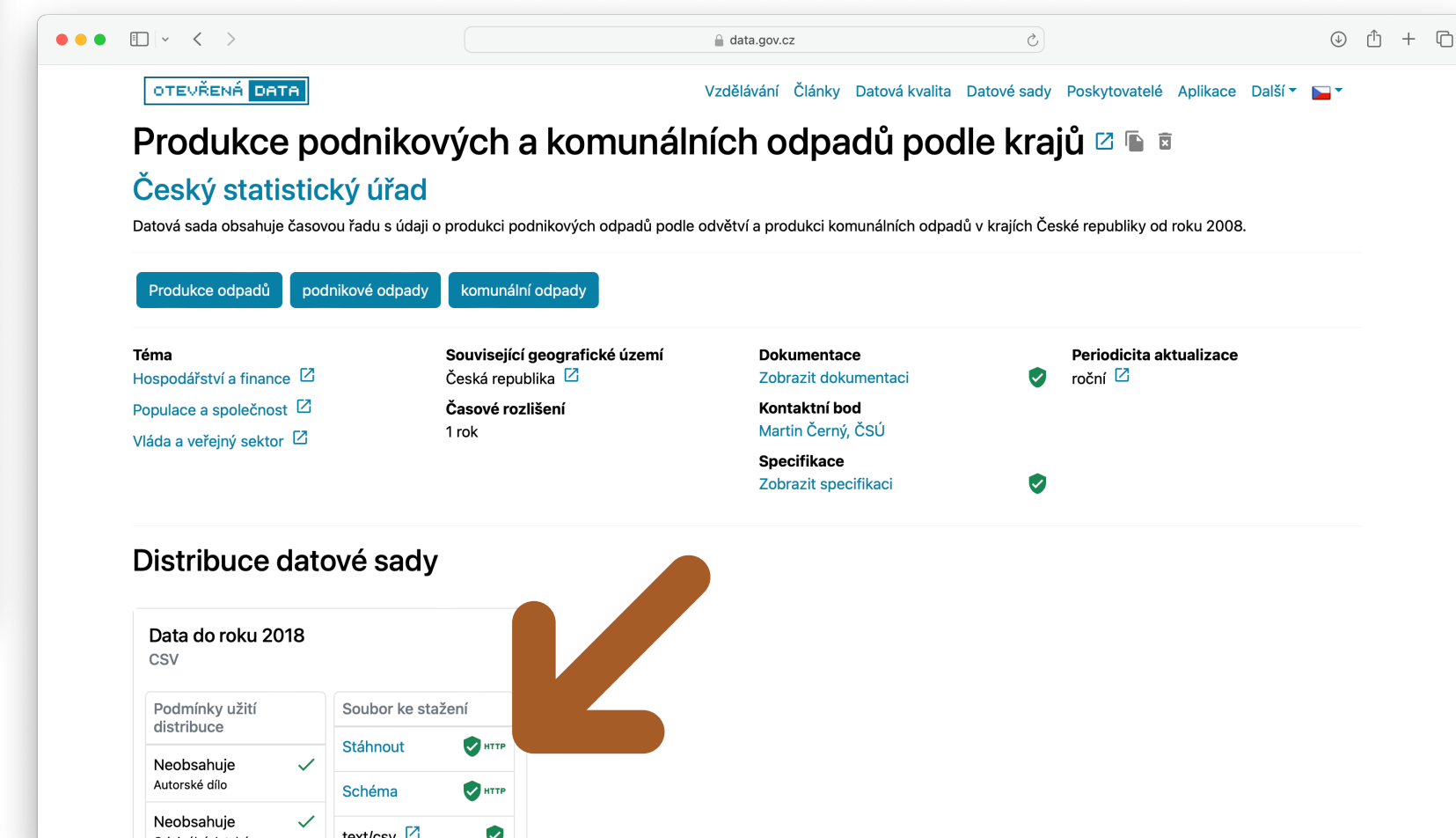
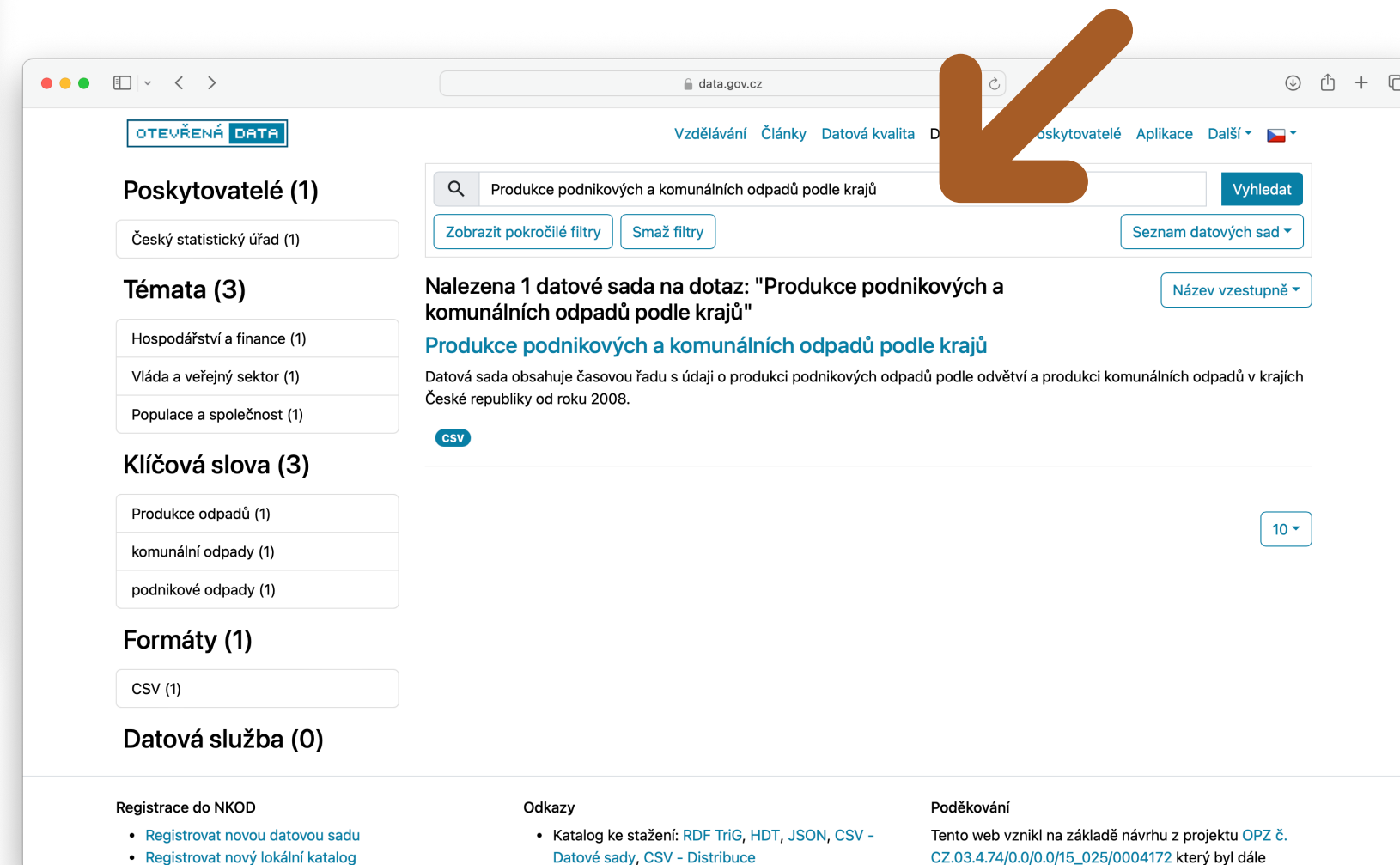
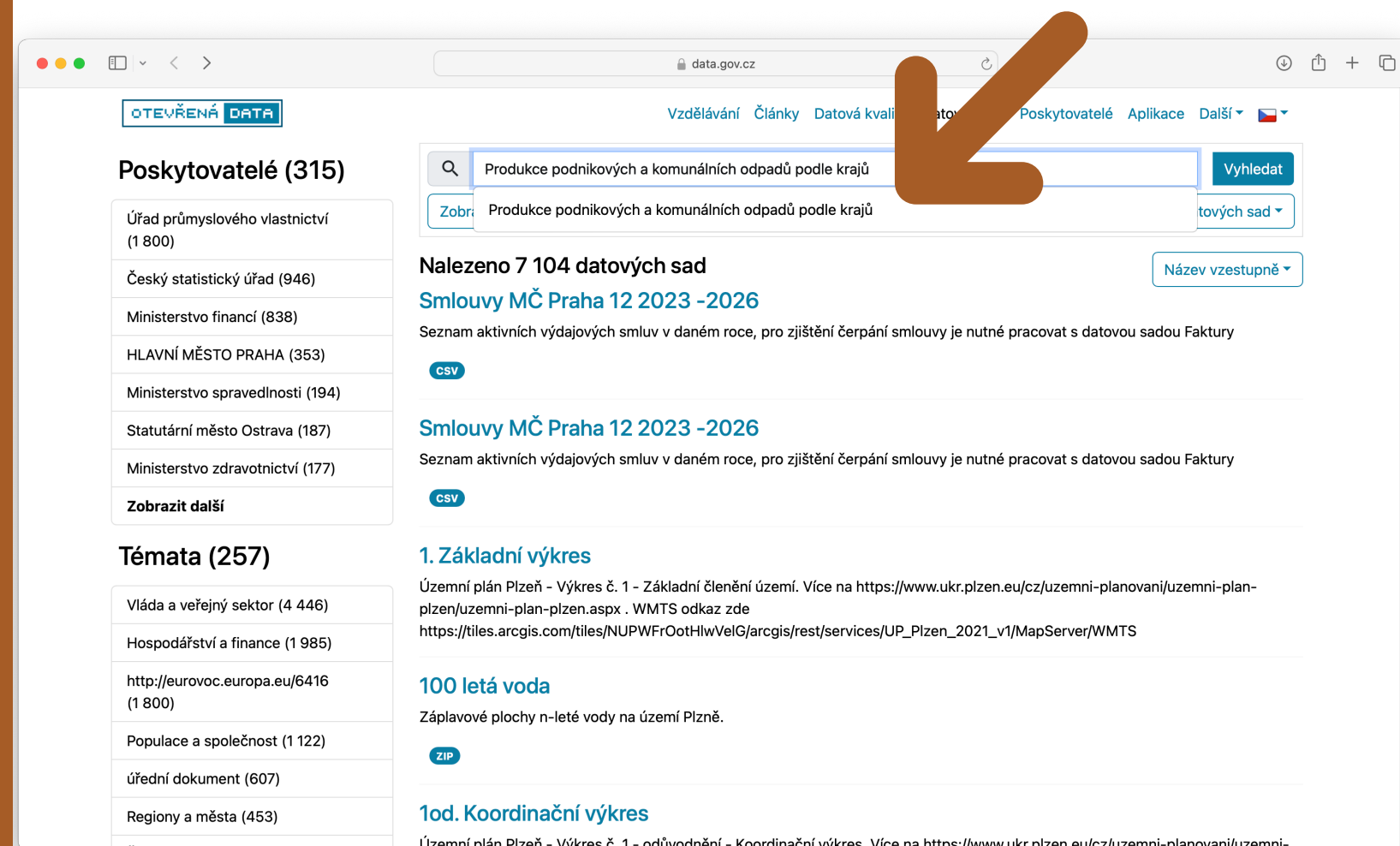
- ❖ *Extract* the *dataset* 'Produkce podnikových a komunálních odpadů podle krajů' published by the Czech Statistical Office in the Czech Open data portal^{#2}
- ❖ Write a function that allows for a *preview of the first 10 rows*
- ❖ *Save* this *dataset into a file* named dataset_waste.csv
- ❖ Implement solution that *downloads data directly* from the web *using Python*

^{#2} <https://data.gov.cz/datové-sady>

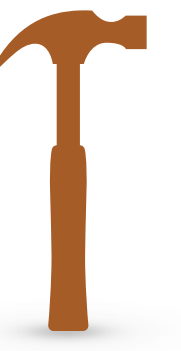
Example 2.1: Extract dataset 'Production of industrial and municipal waste' (Solution)



- ❖ Open the data catalogue at <https://data.gov.cz/datové-sady>
- ❖ Search the dataset 'Produkce podnikových a komunálních odpadů podle krajů'
- ❖ Copy the link leading to dataset^{#3} (i.e., do not download the dataset yet)



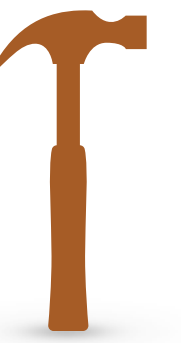
Example 2.1: Extract dataset 'Production of industrial and municipal waste' (Solution)



- ❖ Import the necessary modules (csv, logging, sys, requests)
- ❖ Configure the logging into the console from the INFO level
- ❖ Decompose the program into functions and declare function interfaces (headers) including type hints

```
1 import csv
2 import logging
3 import sys
4
5 import requests
6
7 logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
8
9 def fetch_csv_content(url: str) -> bytes:
10     pass
11
12 def preview_csv_content(data: bytes, num_rows: int = 10) -> None:
13     pass
14
15 def save_as_csv(data: bytes, filename: str) -> None:
16     pass
```

Example 2.1: Extract dataset 'Production of industrial and municipal waste' (Solution)



```
17 def fetch_csv_content(url: str) -> bytes:
18     try:
19         response = requests.get(url)
20         response.raise_for_status() # Raise an exception if unable to download file content
21         return response.content
22     except requests.exceptions.RequestException as e:
23         logging.error(f"Error downloading file from URL: {e}")
24         raise
```

the file is
returned in bytes for further
processing

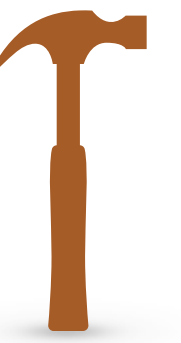
link validation

```
25 def preview_csv_content(data: bytes, num_rows: int = 10) -> None:
26     try:
27         decoded_data = data.decode("utf-8")
28         csv_reader = csv.DictReader(decoded_data.splitlines())
29
30         logging.info(f"Preview of the first {num_rows} rows of the CSV file:")
31         for i, row in enumerate(csv_reader):
32             if i >= num_rows:
33                 break
34             logging.info(row)
35     except Exception as e:
36         logging.error(f"Error previewing CSV content: {e}")
```

decode data into
UTF-8 format

deserialize data
as CSV

Example 2.1: Extract dataset 'Production of industrial and municipal waste' (Solution)



```
37 def save_as_csv(data: bytes, filename: str) -> None:
38     try:
39         with open(filename, "wb") as f:
40             f.write(data)
41     except Exception as e:
42         logging.error(f"Error saving data as CSV file: {e}")
43         raise
```

write bytes to
the named file

```
44 if __name__ == "__main__":
45     if len(sys.argv) != 3:
46         logging.error("Usage: python script.py <url> <output_file_path>")
47         sys.exit(1)
48
49     url = sys.argv[1]
50     output_file_path = sys.argv[2]
51     try:
52         file_content = fetch_csv_content(url)
53         preview_csv_content(file_content, num_rows=10)
54         save_as_csv(file_content, output_file_path)
55         logging.info("File was successfully saved as " + output_file_path)
56     except Exception:
57         logging.error("Failed to download file content.")
```

the program
accepts two arguments: (1) a file
link and (2) the name of the
resulting file

execute program

```
58 python3 extract_waste_csv.py \
    https://csu.gov.cz/docs/107508/159ff6ce-9418-dcef-959b-d55579e332dc/280040-19data112919.csv dataset_waste.csv
```

Exercise 2.2: Extract dataset 'Costs of environmental protection' (JSON)

- ❖ *Extract* the *dataset* 'Náklady na ochranu životního prostředí a ekonomický přínos těchto aktivit' from the practical class website
 - ❖ Download data directly from the web using Python
 - ❖ Write a method that allows for a *preview of the first 10 documents*
 - ❖ Save this dataset into a file named dataset_expenses.json

Note:

- ❖ The dataset was downloaded from the open data portal in CSV format and subsequently transformed into JSON format for teaching purposes
- ❖ Link: <https://data.gov.cz/datová-sada?iri=https://data.gov.cz/zdroj/datové-sady/00025593/15654858d01052b25f929597803588fe>



Exercise 2.3: Web scrapping of the Regions dataset (XML)

- ❖ *Extract the table* 'Základní data o krajích' (Basic data about regions) *from* the *Wikipedia article* about Czech regions^{#4}
- ❖ Download data directly from the web using Python
- ❖ Write a method that allows for a *preview of the first 10 rows*
- ❖ Save extracted table into a file named dataset_regions.csv in the *CSV format*

- ❖ **Tip:** The web page is in XHTML format, hence you are encouraged to use *XPath to navigate* to elements on the page and extract only the table and its content
- ❖ **Tip:** Ideally, elements should contain a *unique identifier @id* that can be used to navigate to exact position in the document. Or, if the desired element does not contain the @id attribute, you should handle it directly, for example, using *relative positioning* or *document order*

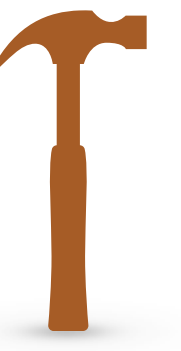


Example 2.4: Extract population dataset from multiple excel sheets (XLSX)

- ❖ *Extract population data* in the regions of the Czech Republic from the Catalog of Products of the Czech Statistical Office
- ❖ Create a *Python script loading* a single *Excel file*, such as 'Vybrané demografické údaje podle krajů v roce 2022'^{#5} (english: 'Selected demographic data by regions in the years 2022')
- ❖ Utilize Python and the *pandas* module to *create a CSV table* with columns region, population, year
 - ❖ All of this data must be extracted from the Excel file
- ❖ Save the result to a file named dataset_population.csv
- ❖ Subsequently, *modify the script* to accept a text file containing links to XLSX files as input, download these files, extract the data into single CSV table (i.e., *process multiple files*)

^{#5} <https://www.czso.cz/csu/czso/4-obyvatelstvo-b096otrb5h>

Example 2.4: Extract population dataset from multiple excel sheets (Solution)

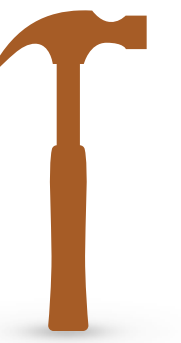


- ❖ Similarly to Example 2.1, import the modules (logging, sys, io, typing, pandas, requests), configure the logging and decompose the program into individual tasks (i.e., functions)

```
60 import logging
61 import sys
62 from io import BytesIO
63 from typing import Any, Dict, List
64 import pandas as pd
65 import requests
66
67 logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
68
69 def fetch_xlsx_content(url: str) -> BytesIO:
70     pass
71
72 def create_header() -> Dict[str, List[Any]]:
73     pass
74
75 def append_data(data: Dict[str, List[Any]], input_content: BytesIO) -> None:
76     pass
77
78 def save_as_csv(data: Dict[str, List[Any]], output_file: str) -> None:
79     pass
```

first we
create the header of the
CSV file and use the next
function to insert individual
lines

Example 2.4: Extract population dataset from multiple excel sheets (Solution)

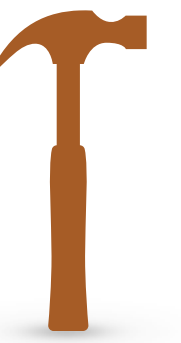


BytesIO allows reading and writing data in memory in the form of bytes

```
80 def fetch_xlsx_content(url: str) -> BytesIO:
81     try:
82         response = requests.get(url)
83         response.raise_for_status() # Raise an exception for bad responses
84         return BytesIO(response.content)
85     except requests.exceptions.RequestException as e:
86         logging.error(f"Error downloading Excel file from URL: {e}")
87         raise
```

```
88 def create_header() -> Dict[str, List[Any]]:
89     return {"region": [], "population": [], "year": []}
```

Example 2.4: Extract population dataset from multiple excel sheets (Solution)



```
90 def append_data(data: Dict[str, List[Any]], input_content: BytesIO) -> None:
91     try:
92         df = pd.read_excel(input_content, header=None)
93         year = str(df.iloc[2, 0])[-4:]
94
95         df = pd.read_excel(input_content, skiprows=5)
96         for column in df.columns[2:16]:
97             region = df.iloc[0][column]
98             index_of_dash = region.find("-")
99             if index_of_dash != -1:
100                 region = region[:index_of_dash] + region[index_of_dash + 2 :]
101                 data["region"].append(region.strip())
102                 data["population"].append(df.iloc[1][column])
103                 data["year"].append(year)
104     except Exception as e:
105         logging.error(f"An error occurred while appending data: {e}")
106     raise
```

extract the year from
the excel file in cell A4 (last four
characters of the string)

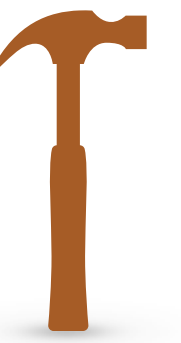
information
on regions is stored in columns
B to P

the
region name can be
split into multiple
lines

```
07 def save_as_csv(data: Dict[str, List[Any]], output_file: str) -> None:
08     try:
09         df = pd.DataFrame(data)
10         df.to_csv(output_file, index=False)
11     except Exception as e:
12         logging.error(f"An error occurred while saving data to CSV: {e}")
13     raise
```

create
pandas dataframe from
data and write its content to
csv file

Example 2.4: Extract population dataset from multiple excel sheets (Solution)



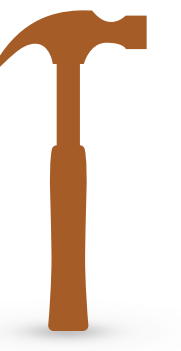
```
14 if __name__ == "__main__":
15     if len(sys.argv) != 3:
16         logging.error("Usage: python script.py <url> <output_file_path>")
17         sys.exit(1)
18
19     url = sys.argv[1]
20     output_file_path = sys.argv[2]
21
22     try:
23         data = create_header()
24
25         excel_content = fetch_xlsx_content(url)
26         append_data(data, excel_content)
27
28         save_as_csv(data, output_file_path)
29         logging.info("File was successfully saved as " + output_file_path)
30     except Exception:
31         logging.error("Failed to download file content.")
```

the program
accepts two arguments: (1) a file
link and (2) the name of the
resulting file

execute program

```
32 python3 extract_population_excel.py https://csu.gov.cz/docs/107508/6a814cc6-4cb4-c0dc-21cf-  
ca892c32a677/2304101.xlsx?version=1.0 dataset_population.csv
```

Example 2.4: Extract population dataset from multiple excel sheets (Solution)



```
35 def read_urls(urls_file: str) -> List[str]:
36     try:
37         with open(urls_file, "r") as file:
38             urls = [line.strip() for line in file.readlines()]
39         return urls
40     except Exception as e:
41         logging.error(f"An error occurred while reading URLs from the file: {e}")
42         raise
```

reading
a list of links from a
text file

```
43 ...
44 urls_file = sys.argv[1]
45 output_file_path = sys.argv[2]
46
47 try:
48     urls = read_urls(urls_file)
49     data = create_header()
50
51     for url in urls:
52         excel_content = fetch_xlsx_content(url)
53         append_data(data, excel_content)
54
55     save_as_csv(data, output_file_path)
56 ...
```

the modified program
accepts two arguments: (1) a text file
with links and (2) the name of the
output file

each input file
is processed and the
data is appended to the
output file

References

Python

- ❖ Python 3.x (LATEST) documentation: <https://docs.python.org/3/>
- ❖ venv documentation. <https://docs.python.org/3/library/venv.html>
- ❖ Python W3Schools Tutorial: <https://www.w3schools.com/python/>
- ❖ Pandas W3Schools Tutorial: <https://www.w3schools.com/python/pandas/default.asp>

XML

- ❖ XPath W3Schools Tutorial: https://www.w3schools.com/xml/xpath_intro.asp

Datasets

- ❖ Czech open data portal: <https://data.gov.cz/>
- ❖ Czech Statistical Office product catalog: <https://www.czso.cz/csu/czso/katalog-produktu>