UNIVERSITY OF AMSTERDAM

CS Staff Course:

Using and Teaching Python

Day 2

Dr. Joanna Strycharz & Dr. A. Marthe Möller

Day 2

- Introduction and (Github) check-in
- Recap: Functions and methods
- Recap: Handling error messages
- Warm-up exercise 3
- In-class exercise 1
- Recap: Reading in datafiles
- Recap: Working with Pandas
- Warm-up exercise 4
- In-class exercise 2
- Teaching exercise

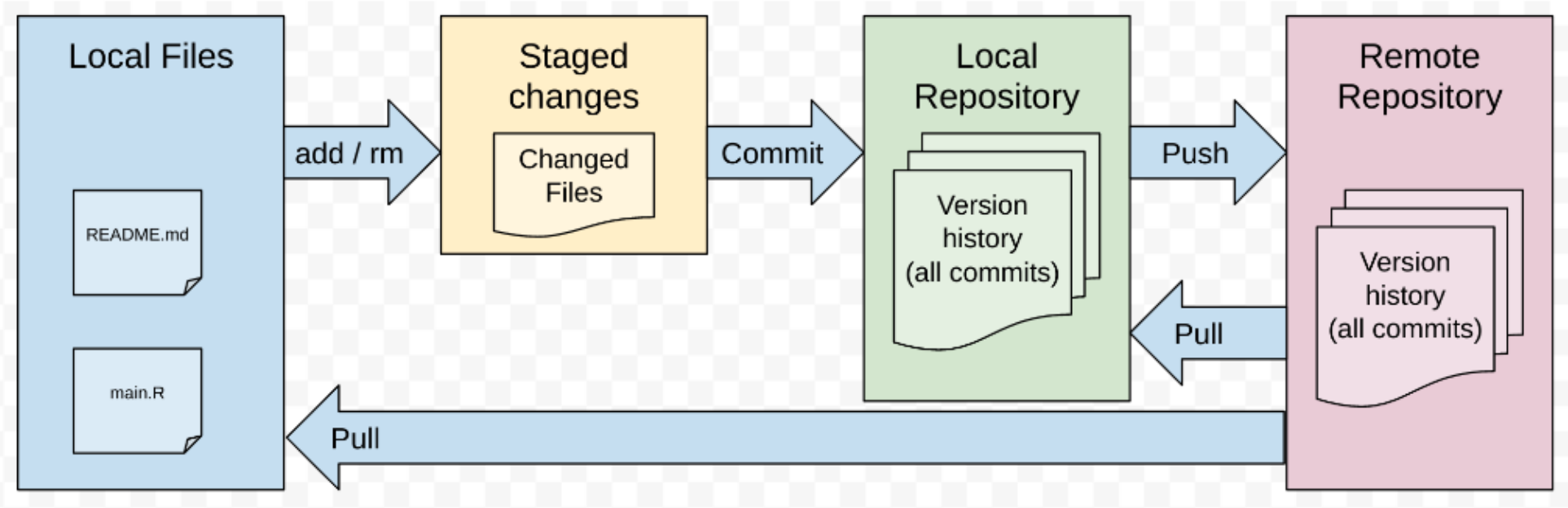
UNIVERSITY OF AMSTERDAM

Introduction

Dr. A. Marthe Möller
Assistant Professor of
Entertainment Communication

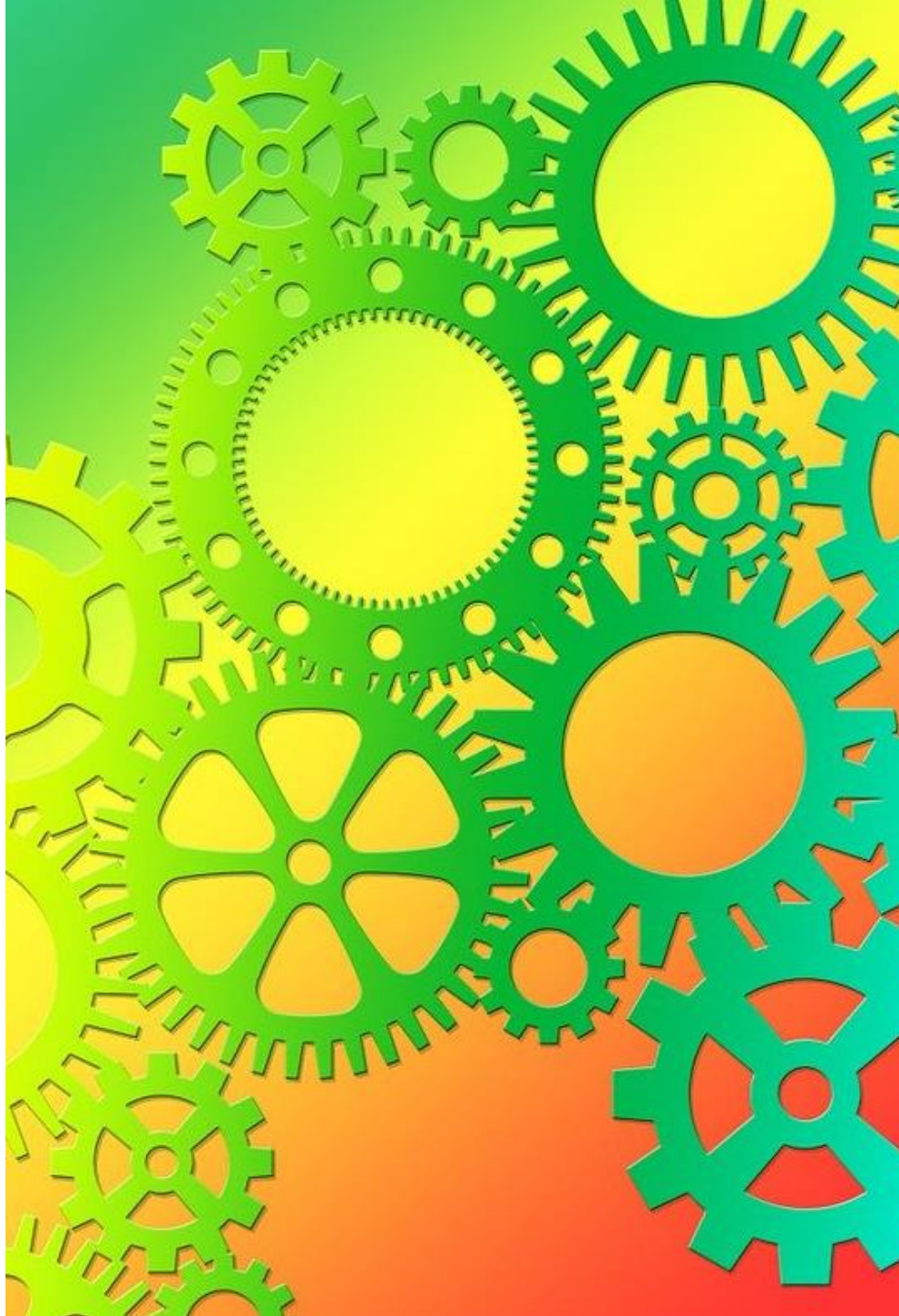
Github: How did (G)it go?

- `git add` : add changes to the *staging area*
- `git commit` : save the *staged* changes in the `local repository`
- `git push` : push the committed changes to the `remote repository`
- `git pull` : if changes have been committed to the `remote repository`, pull them to the `local repository`, and update your local files.



This slide is based on the Github tutorial by Van Atteveldt and Welbers (2021).

Functions and methods



Functions and methods

- Objects that store some statements and operations
- Reusable
- Contributes to simpler code

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 3.3). and the materials of the course CCS-1.

Functions vs. methods

- Function: independent from the object
- Method: associated to specific object

```
stringy = "Bonjour"
stringy.lower()
```

```
'bonjour'
```

```
listy = ["Bonjour"]
listy.lower()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[3], line 2
      1 listy = ["Bonjour"]
----> 2 listy.lower()

AttributeError: 'list' object has no attribute 'lower'
```

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 3.3). and the materials of the course CCS-1.

DIY functions

Arguments that the function uses

- As many as you need
- Arbitrary naming
- Optional

Name of the function (arbitrary)

- No number at the beginning
- No spaces
- A name not used by built-in functions

Indicates that the function has been defined

```
def addone(number):
    new_number = number + 1
    return new_number
```

Defining (creating) the function

Return the result

Defines what the function should do (notice indent!)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 3.3). and the materials of the course CCS-1.

DIY functions: Using loops (applied to a string)

```
mylist = [1,2,3,4,5]

def addone(number):
    new_number = number + 1
    return new_number

for n in mylist:
    print(addone(n))
```

What output would you expect to be printed now?

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 3.3). and the materials of the course CCS-1.

DIY functions: Using loops (applied to a dict)

```
mydict = {1:'one',2:'two',3:'three',4:'four',5:'five'}

def addone(number):
    new_number = number + 1
    return new_number

for k,v in mydict.items():
    print(addone(k))
```

What output would you expect to be printed now?

Error messages

Error messages

- *Where* does the error pop up?
- What *type* of error message are you getting?

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 4.2). and the materials of the course CCS-1.

Error messages: Where?

```
mydict = {1:'one',2:'two',3:'three',4:'four',5:'five'}

def addone(number):
    new_number = number + 1
    return new_number

for k,v in mydict.items():
    print(addone(v))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 8
      5     return new_number
      7 for k,v in mydict.items():
----> 8     print(addone(v))

Cell In[11], line 4, in addone(number)
      3 def addone(number):
----> 4     new_number = number + 1
      5     return new_number

TypeError: can only concatenate str (not "int") to str
```

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 4.2). and the materials of the course CCS-1.

Error messages: What?

```
mydict = {1:'one',2:'two',3:'three',4:'four',5:'five'}

def addone(number):
    new_number = number + 1
    return new_number

for k,v in mydict.items():
    print(addone(v))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 8
      5     return new_number
      7 for k,v in mydict.items():
----> 8     print(addone(v))

Cell In[11], line 4, in addone(number)
      3 def addone(number):
----> 4     new_number = number + 1
      5     return new_number

TypeError: can only concatenate str (not "int") to str
```

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 4.2). and the materials of the course CCS-1.

Error messages: Tips and tricks

- Print a lot
- Simplify – make a simpler version and see if/how it works
- Use sanity checks (e.g., checking if two lists have the same length, if this is required/expected)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 4.2). and the materials of the course CCS-1.

After the break:

- ~~Introduction and (Github) check-in~~
- ~~Recap: Functions and methods~~
- ~~Recap: Handling error messages~~
- Warm-up exercise 3
- In-class exercise 1
- Recap: Reading in datafiles
- Recap: Working with Pandas
- Warm-up exercise 4
- In-class exercise 2
- Teaching exercise

UNIVERSITY OF AMSTERDAM

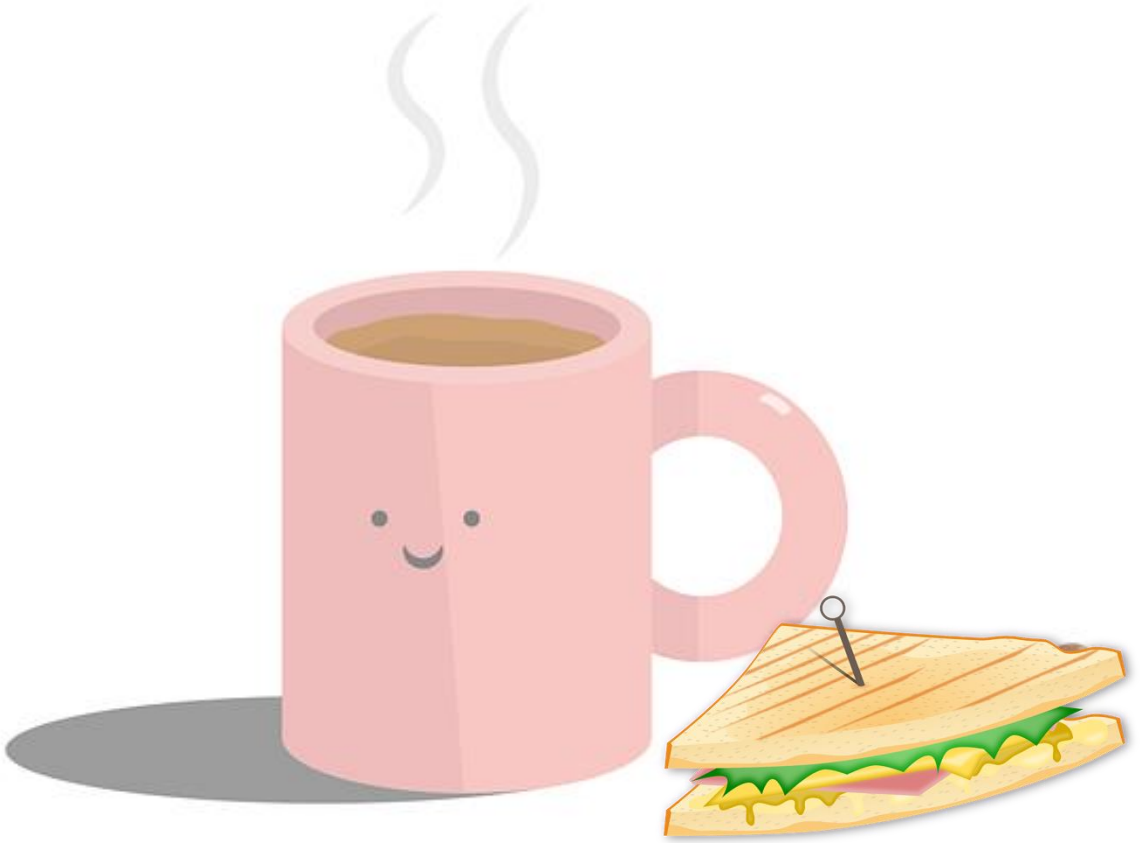
Warm-up Exercise 3

UNIVERSITY OF AMSTERDAM

Peer coding:
In-class exercise I

After lunch:

- ~~Introduction and (Github) check-in~~
- ~~Recap: Functions and methods~~
- ~~Recap: Handling error messages~~
- ~~Warm-up exercise 3~~
- ~~In-class exercise 1~~
- Recap: Reading in datafiles
- Recap: Working with Pandas
- Warm-up exercise 4
- In-class exercise 2
- Teaching exercise



Files

Lists and dicts: Limitations

- What are the limitations of lists?
- What are the limitations of dicts?

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 5). and the materials of the course CCS-1.

Structuring data

- Lists of lists
- Nested data (dictionaries, combined with lists)
- Data frames:
  - Tabular format
  - From list of lists, dict, file
  - Columns and rows can have a name

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 5). and the materials of the course CCS-1.

To dataframe or not to dataframe

Data frames:

- Tabular data
- Easy to inspect
- Easier to analyze statistically
- R/SPSS/State-user friendly

Other formats:

- Non-tabular data (nested, network)
- One dimensional (one column)
- Large – memory and time

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 5). and the materials of the course CCS-1.

Working with data files in Python

1. Read the file into a data frame, list etc. Give it a name, e.g., read in file mydata.csv to df
2. Operations on data, transforming them
3. Write the transformed data into a new file, e.g., mynewdata.csv

Don’t forget: Computers are stupid ☺ Tell them where your files are!

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 5). and the materials of the course CCS-1.

Files and delimiters

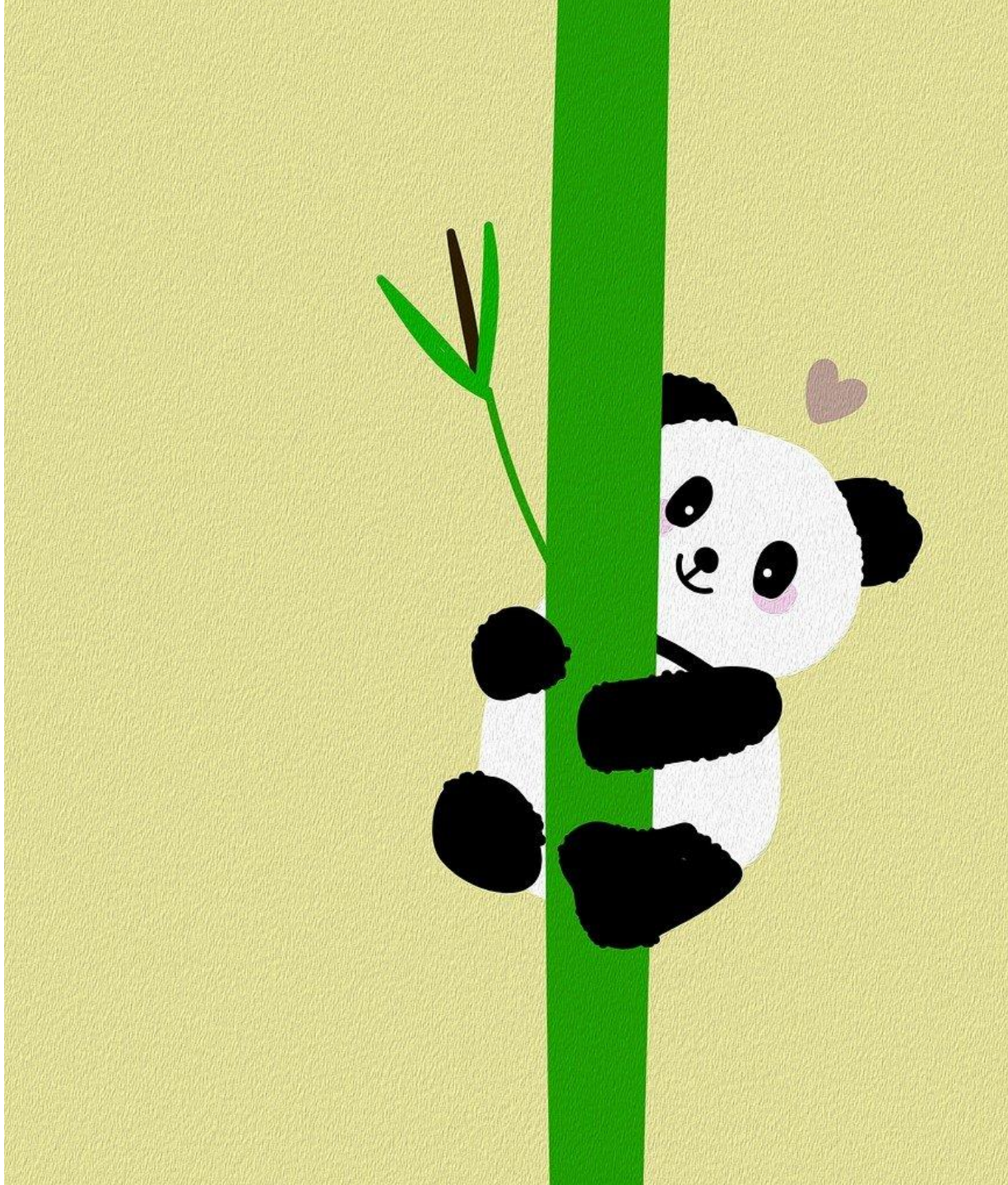
What defines the next line in your file?

1. \n
2. , (comma separated files)
3. \t (tab separated files)
4. ;

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 5). and the materials of the course CCS-1.

UNIVERSITY OF AMSTERDAM

Pandas



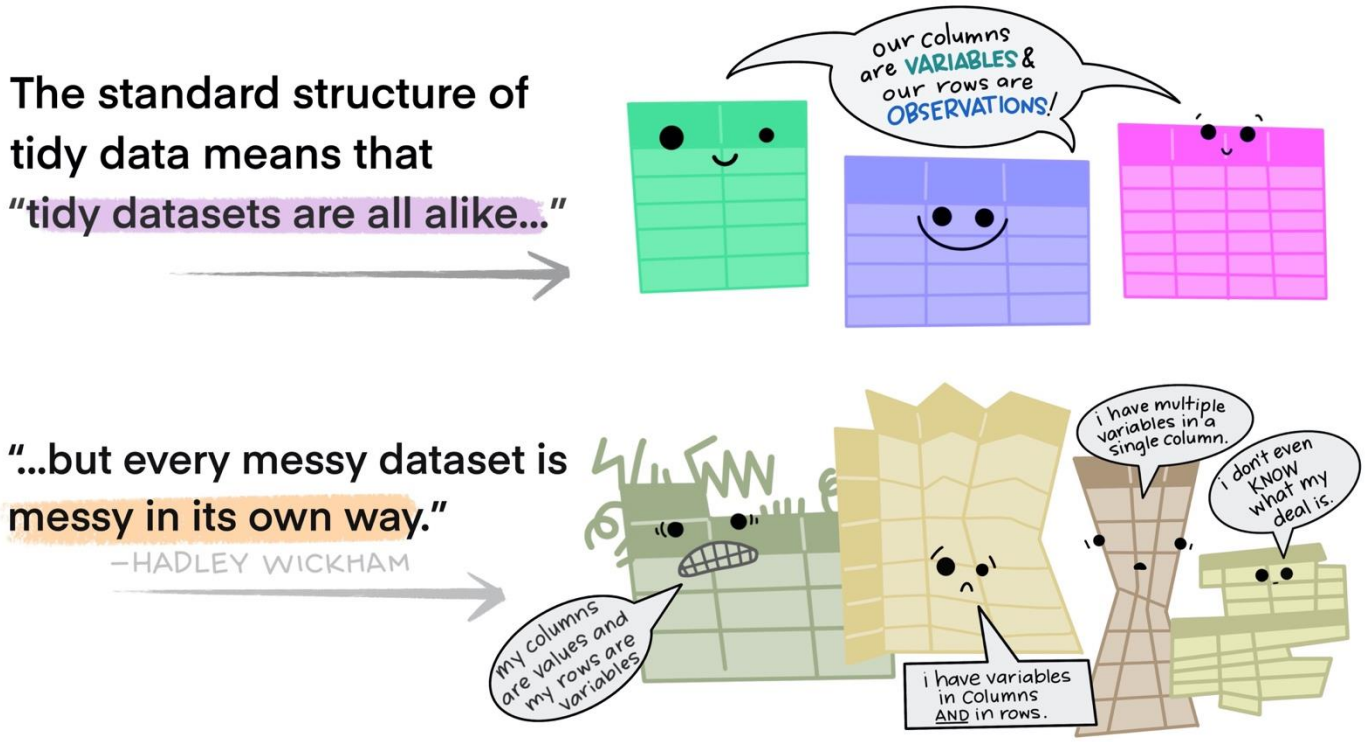
Pandas

- Pandas: fast, powerful, flexible, and easy to use open-source data analysis and manipulation package for Python
- Pandas in generally imported as pd via: import pandas as pd
- You can use it for data **wrangling**:
  - Creating dataframes
  - Reading and writing data (xlsx, csv, json, sav, etc.)
  - Filtering, selecting and renaming data
  - Merging dataframes

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6). and the materials of the course CCS-1.

Data wrangling

Transforming raw data into a shape that is suitable for analysis



This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6). and the materials of the course CCS-1.

Data wrangling: How to

1. Upon leading a dataset, use .head() to examine the data
2. Print all column names using .columns to see all the columns in the dataset
3. Use .isna().sum() to see which columns contain missing values (NaN)
4. Check if all column types are as expected using .dtypes

Make a game plan based on:

- Which columns do you need?
- Which values do you expect?
- How will you deal with missing values (if any)?
- How will you deal with unexpected column types?

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6). and the materials of the course CCS-1.

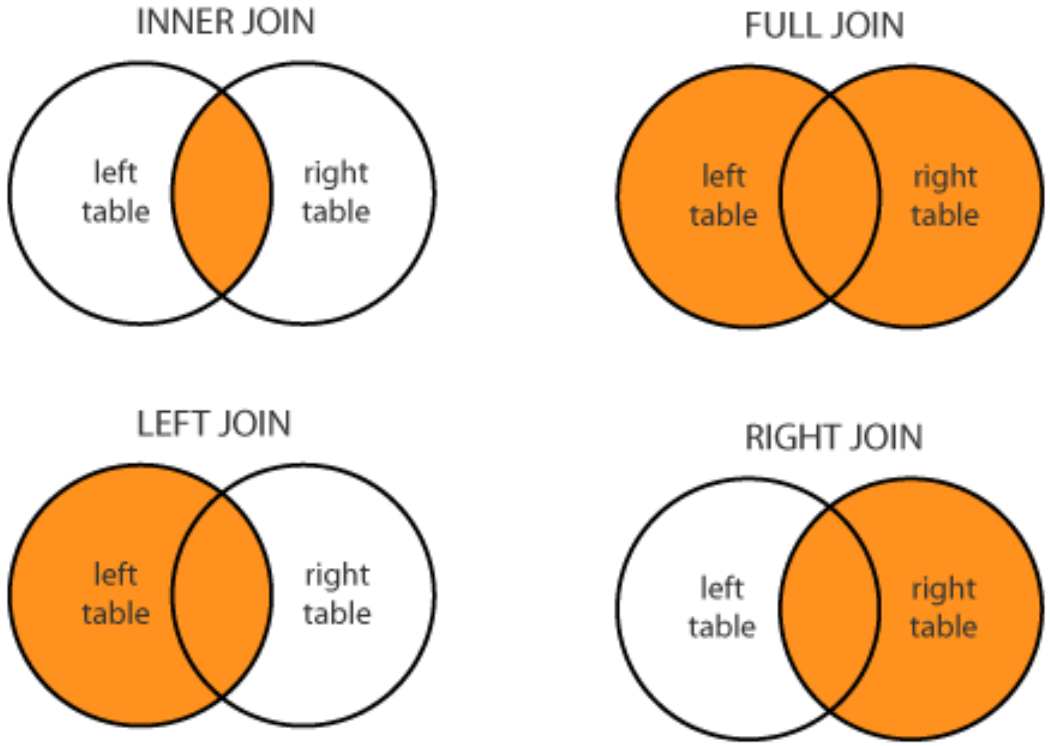
Dealing with missing values: Options

1. Ignore them (not a very good idea)
2. Remove any rows with any NAs (also not a very good idea)
3. Remove any column with Nas (maybe a good idea)
4. Remove Nas in specific columns (maybe a good idea)
5. Replace Nas with a value (maybe a good idea)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6). and the materials of the course CCS-1.

Merging datasets

1. Requires a shared column across the two datasets
2. Use df.merge
3. Different types of merging:



This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6). and the materials of the course CCS-1.

Merging datasets: Things to remember

1. There must be *at least one shared column*
2. The shared column might be named differentlt across the two datasets
    - Rename in one dataset or (rather: and) specify column names when merging
3. Which observations do you want the final dataset to contain?
4. Will your final dataset include NaN values and is that a problem?

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6). and the materials of the course CCS-1.

UNIVERSITY OF AMSTERDAM

Warm-up Exercise 4

After the break:

- ~~Introduction and (Github) check-in~~
- ~~Recap: Functions and methods~~
- ~~Recap: Handling error messages~~
- ~~Warm-up exercise 3~~
- ~~In-class exercise 1~~
- ~~Recap: Reading in datafiles~~
- ~~Recap: Working with Pandas~~
- ~~Warm-up exercise 4~~
- In-class exercise 2
- Teaching exercise

UNIVERSITY OF AMSTERDAM

Peer coding:
In-class exercise II

After the break:

- ~~Introduction and (Github) check-in~~
- ~~Recap: Functions and methods~~
- ~~Recap: Handling error messages~~
- ~~Warm-up exercise 3~~
- ~~In-class exercise 1~~
- ~~Recap: Reading in datafiles~~
- ~~Recap: Working with Pandas~~
- ~~Warm-up exercise 4~~
- ~~In-class exercise 2~~
- Teaching exercise

UNIVERSITY OF AMSTERDAM

Teaching exercise

How to teach Python?

- **Live coding:** demonstrate concepts by writing code in real-time. This shows students how to approach problem-solving and debugging naturally.
- **Encourage pseudocode:** before diving into code, teach students to outline their logic in plain language. This helps to think logically without being overwhelmed by syntax.
- **Peer programming:** Let students work in pairs/small groups. This encourages collaboration, helps them to learn from each other and mimic real-world coding practices.
- **Code reviews:** Ask students to review each others' code to learn alternative approaches, catch mistakes, and teach the ability to critique code constructively.

Try teaching it yourself!

1. Make groups of 3
2. Pick a concept
3. Prepare:
   - Explanation
   - Small exercise
   - Max. 10 minutes
4. Present to your group + feedback

Concepts

1. What are functions and how to use them?
2. How to handle error messages?
3. How to best work with data files?
4. What is Pandas and what can it be used for?
5. How can datasets be merged?

Voor de volgende keer:

- Maak de huiswerkopdracht ter voorbereiding van werkgroep 2.1 in Canvas:
  - Lees de literatuur uit de opdracht
  - Beantwoord de vragen
  - Dien je opdracht in via Canvas voor aanstaande maandag 12.00 uur