



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



CHAI

Humanities-Centered AI

Understanding Data vs. Machine Training

**Malte Luttermann – Institute for Humanities-Centered
Artificial Intelligence (CHAI)**

November 21, 2025

Overview of Contents

- (1) Programming language Python
 - (a) Introduction and first steps
 - (b) Basics
 - (c) Advanced
- (2) Markup languages
 - (a) \LaTeX
 - (b) Markdown
- (3) Development environments
 - (a) Jupyter notebooks
- (4) Version control
 - (a) Git and GitHub
- (5) Scientific computing
 - (a) NumPy and SciPy
- (6) Data processing and visualisation
 - (a) Pandas, matplotlib, and NLTK
- (7) Machine learning (scikit-learn)
 - (a) Basics (datasets, analysis)
 - (b) Simple methods (clustering, ...)
- (8) Deep learning
 - (a) PyTorch

Topics for Today

- (1) Development environments
 - Jupyter notebooks
 - Basics (Bash-)terminal
- (2) Regular expressions (RegEx)

Acknowledgement

- The upcoming slides are taken from the following lecture and have been translated and partially modified:
 - Dr. Magnus Bender: »[Python für Machine Learning und Data Science](#)« as part of the German course »Werkzeuge für das wissenschaftliche Arbeiten«

Development Environments

- Many development environments with Python support are available, e.g.,
 - VS Code
 - PyCharm

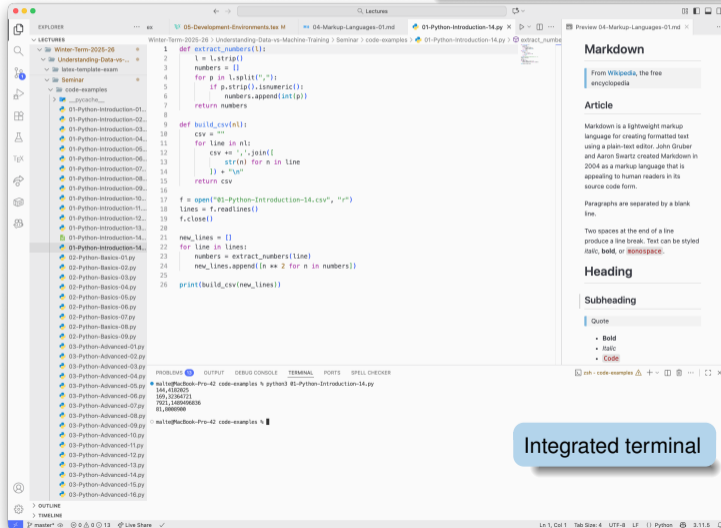
Development Environments

Editor tabs with different files

Git version control

Debugger

Many extensions



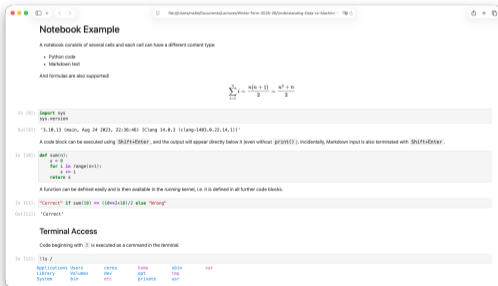
Integrated terminal

Development Environments

- Many development environments with Python support are available, e.g.,
 - VS Code
 - PyCharm
- Development in the Browser
 - <https://vscode.dev>
 - Jupyter notebooks

Jupyter Notebooks

- Rendering in browser (and also, e.g., VS Code)
- Installation local or usage in the cloud
- Combination of program code, visualizations, and text (Markdown)



Kernel (e.g., Python)



HTTP

HTTP connection
(locally or via internet)



Notebook (Python code,
output, Markdown text)

Jupyter Notebooks

Notebook Example

A notebook consists of several cells and each cell can have a different content type:

- Python code
- Markdown text

And formulas are also supported!

$$\sum_{i=1}^n i = \frac{n(n+1)}{2} = \frac{n^2 + n}{2}$$

In [9]: `import sys`
`sys.version`

Out[9]: '3.10.13 (main, Aug 24 2023, 22:36:46) [Clang 14.0.3 (clang-1403.0.22.14.1)]'

A code block can be executed using `Shift+Enter`, and the output will appear directly below it (even without `print()`). Incidentally, Markdown input is also terminated with `Shift+Enter`.

In [10]: `def sum(n):`
 `s = 0`
 `for i in range(n+1):`
 `s += i`
 `return s`

A function can be defined easily and is then available in the running kernel, i.e. it is defined in all further code blocks.

In [11]: `"Correct" if sum(10) == (10**2+10)/2 else "Wrong"`

Out[11]: 'Correct'

Terminal Access

Code beginning with `!` is executed as a command in the terminal.

In [12]: `!ls /`

| | | | | | |
|--------------|---------|-------|---------|------|-----|
| Applications | Users | cores | home | sbin | var |
| Library | Volumes | dev | opt | tmp | |
| System | bin | etc | private | usr | |

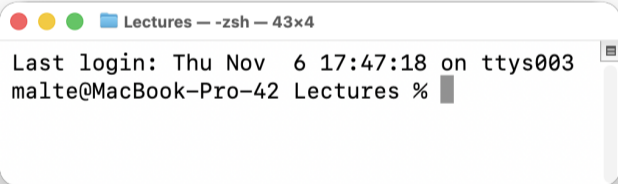
Jupyter in the Cloud

- Example: Google Colab
 - <https://colab.research.google.com>
 - Free to use
 - Access to GPUs



The (Bash-)Terminal

- We already had some examples of terminal commands, ...
- ... but we did not really look into the terminal itself!



```
Lectures — zsh — 43x4  
Last login: Thu Nov  6 17:47:18 on ttys003  
malte@MacBook-Pro-42 Lectures %
```

Unix-Like Operating Systems

■ Basics:

- »Everything is a file«
- »Unix-Shell«
- User administration (Admin user: root)

■ Examples:

- Linux, macOS, BSD
- On Windows: Usage of Linux via WSL (Windows Subsystem for Linux) possible

Paths

- `/`: Root path
- `./`: Current directory
- `../`: Parent directory
- `~`: Home directory of the current user
- Hidden (file-)names start with `.`

```
/
bin/
dev/
etc/
home/
  malte/
mnt/
media/
sys/
usr/
var/
log/
  auth.log
tmp/
```

`bin/`: Programs
`dev/`: Devices
`etc/`: Configurations
`home/`: User directories
`mnt/`, `media/`: Removable media
`sys/`: Kernel
`usr/`: Program files
`var/`: Various files
`tmp/`: Temporary files

File System

```

malte -- -zsh -- 102x28
Last login: Mon Nov 10 16:25:42 on ttys007
malte@MacBook-Pro-42 ~ % ls -la /
total 10
drwxr-xr-x  22 root  wheel   704 Sep 25 09:03 .
drwxr-xr-x  22 root  wheel   704 Sep 25 09:03 ..
lrwxr-xr-x   1 root  admin   36 Sep 25 09:03 .VolumeIcon.icns -> System/Volumes/Data/.VolumeIcon.icns
-----   1 root  admin    0 Sep 25 09:03 .file
drwxr-xr-x   2 root  wheel   64 Sep 25 09:03 .nofollow
drwxr-xr-x   2 root  wheel   64 Sep 25 09:03 .resolve
drwxr-xr-x   2 root  wheel   64 Sep 25 09:03 .vol
drwxrwxr-x  42 root  admin  1344 Nov  5 09:21 Applications
drwxr-xr-x  68 root  wheel  2176 Sep 30 19:57 Library
drwxr-xr-x@ 10 root  wheel   320 Sep 25 09:03 System
drwxr-xr-x   5 root  admin   160 Sep 30 19:56 Users
drwxr-xr-x   3 root  wheel   96 Nov  6 17:45 Volumes
drwxr-xr-x@ 39 root  wheel  1248 Sep 25 09:03 bin
drwxr-xr-x   2 root  wheel   64 Sep  9 09:15 cores
dr-xr-xr-x   4 root  wheel  4850 Nov  4 10:03 dev
lrwxr-xr-x@  1 root  wheel   11 Sep 25 09:03 etc -> private/etc
lrwxr-xr-x   1 root  wheel   25 Nov  4 10:04 home -> /System/Volumes/Data/home
drwxr-xr-x   4 root  wheel   128 Oct  8 11:58 opt
drwxr-xr-x   6 root  wheel   192 Nov  4 10:03 private
drwxr-xr-x@ 76 root  wheel  2432 Sep 25 09:03 sbin
lrwxr-xr-x@  1 root  wheel   11 Sep 25 09:03 tmp -> private/tmp
drwxr-xr-x@ 11 root  wheel   352 Sep 25 09:03 usr
lrwxr-xr-x@  1 root  wheel   11 Sep 25 09:03 var -> private/var
malte@MacBook-Pro-42 ~ %
  
```

- File type
- Access rights
- Owner and group
- File size (not folder size!)
- Last modified date
- Name

Terminal Commands

- Program execution via the terminal

```
<program> [[-p <value>] --<parameter> <value>] <arguments>
```

- Program name
- Optional parameters
 - Parameters often have a short (-p) and a long version (--parameter)
- Arguments
- Separation by spaces

Useful Commands I

| | | | |
|-------------------------------------|-------|------------------------------|----------------------------------|
| Navigation in the file system | cd | cd .. | Change directory |
| | ls | ls . | List files in a directory |
| | pwd | pwd | Show current directory |
| | find | find . -name "*.txt" -type f | Search for files |
| | grep | grep "hi" ./file.txt | Search within files |
| Directories | mkdir | mkdir ./new/ | Create a new directory |
| | rmdir | rmdir ./old/ | Remove an empty directory |
| Files | cat | cat ./file.txt | Output the contents of a file |
| | tail | tail -n 5 ./file.txt | Output the last lines of a file |
| | head | head -n 5 ./file.txt | Output the first lines of a file |
| | touch | touch ./new.txt | Create an empty file |
| | nano | nano ./file.txt | Edit a file |
| | vim | vim ./file.txt | Edit a file |
| | rm | rm ./file.txt | Delete a file (or directory) |

Useful Commands II

| | | | |
|-------------------|----------------------|---|----------------------------|
| System monitoring | <code>top</code> | | Show running processes |
| | <code>htop</code> | | Task manager (GUI-like) |
| | <code>free</code> | | Show memory usage |
| | <code>df -h</code> | | Show hard drives and usage |
| Help | <code>man</code> | <code>man ls</code> | Open help page |
| | <code>whereis</code> | <code>whereis python</code> | Show path to program |
| | <code>whoami</code> | | Show current user name |
| Network | <code>wget</code> | <code>wget http://example.com/file.zip</code> | Download file |
| | <code>curl</code> | <code>curl -I http://example.com</code> | Download file |

The Terminal

- Execution environment for commands
 - Terminate via `Ctrl+C`
- Various designs, similar structure
- Auto completion using the tab key (including program and file names)
- Navigation through recent commands using the arrow keys up/down

Lectures — -zsh — 43x4

Last login: Thu Nov 6 17:47:18 on ttys003

malte@MacBook-Pro-42 Lectures %

The screenshot shows a terminal window with a title bar 'Lectures — -zsh — 43x4'. The main content displays the login message 'Last login: Thu Nov 6 17:47:18 on ttys003' followed by the prompt 'malte@MacBook-Pro-42 Lectures %'. Three red boxes highlight the components: 'malte' (User), 'MacBook-Pro-42' (Host), and 'Lectures' (Current directory).

User

Host

Current directory

The Symbols `>` `&|>` `>>` `&&` `*` `?`

- Redirect output to a file (`>` to overwrite and `>>` to append)

```
echo "Hello world" > ./file.txt
```

```
echo "Bye!" >> ./file.txt
```

- Pipe stdout of first program to stdin of second program

```
echo "1,2,3" | grep "2"
```

```
cat ./file.txt | python ./process.py
```

- Execute two commands after each other (`&&`) or at the same time (`&`)

```
sleep 1 && echo "Hi" & echo "Hi2"
```

- Wildcards (`*` for any character sequence, `?` for a single character)

```
cat *.txt
```

```
cat ?.txt
```

```
1 Hello world  
2 Bye!
```

file.txt

Hi2
Hi

file1.txt file2.tex a.txt
file1.txt file2.tex a.txt

Regular Expressions (RegEx)

- A powerful and simple way to process strings
- Implemented in Python in the `re` package
- Not only supported in Python, but also in, e.g., Java with `String.matches()` or in PHP with `preg_match()`
- Useful tool to create and test RegEx patterns: <https://regexr.com>

RegEx – Idea

- Detect (partial) character strings with specific patterns
 - »Hello« or »Hallo« → `H(e|a)llo`
- Extract parts
 - »Warburgstrasse 28« → `.*strasse (\d+)`
 - Capturing group extracts »28«

RegEx – Syntax I

| | | |
|-------------------|--------------------------------|---|
| Beginning and end | ^ | Beginning of the string |
| | \$ | End of the string |
| Escape | \ | Escape symbol |
| Character sets | . | An arbitrary character (no newline) |
| | [A-Z], [abcd], [0-9], [a-z0-9] | Set of characters |
| | [^A-z], [^\- \- ()] | Negation of a set of characters |
| | \w, \d, \s | Characters, numbers, spaces (all types) |
| Repetitions | \W, \D, \S | No characters, numbers, spaces (all types) |
| | * | Any number of repetitions of the character/-group before |
| | + | At least one repetition of the character/-group before |
| | ? | One or none repetition of the character/-group before |
| | {m}, {m,n} | m or m to n repetitions of the character/group before, respectively |

RegEx – Beginning, End, Characters

```
1 import re
2
3 re.match(r'^https?://www.example.com/?', "https://www.example.com/")
4 # match
5 re.match(r'^https?://www.example.com/?', "http://www.example.com/")
6 # match
7
8 re.match(r'^https?://www.example.com/?', "http://www0example0com/")
9 # match (but it should not!)
10 re.match(r'^https?://www\.example\.com/?', "http://www0example0com/")
11 # no match
12
13 re.match(r'^https?://www\.example\.com/?', "http://www.example.com/hello")
14 # match (but it should not!)
15 re.match(r'^https?://www\.example\.com/?$', "http://www.example.com/hello")
16 # no match
```

RegEx – Repetitions and Sets

```
1 pattern = r'^@[a-z]\.?uni-luebeck\.de '  
2 re.match(pattern, "luttermann@ifis.uni-luebeck.de")  
3 # no match  
4 re.match(r'^@[a-z]+\.?uni-luebeck\.de', "luttermann@ifis.uni-luebeck.de")  
5 # match  
6  
7 re.match(r'^@[a-z]+\.?uni-luebeck\.de', "m.luttermann@uni-luebeck.de")  
8 # no match  
9 re.match(r'^@[a-z]*\.?uni-luebeck\.de', "m.luttermann@uni-luebeck.de")  
10 # match  
11 re.match(r'\w+@\w*\.?uni-luebeck\.de', "luttermann@ifis.uni-luebeck.de")  
12 # match  
13  
14 re.match(r'\w+@\w*\.?uni-luebeck\.de', "m.luttermann@uni-luebeck.de")  
15 # no match (\w contains no symbols)  
16 re.match(r'\w+@\w*\.?uni-luebeck\.de', "luttermann@FAKEuni-luebeck.de")  
17 # match (but it should not!)
```

RegEx – Syntax II

| | | |
|--------------------|------------------------|---|
| Disjunction | <code>... ...</code> | Matches if any of the RegEx ... matches |
| Groups | <code>(...)</code> | Capturing group for RegEx ... |
| | <code>(?: ...)</code> | Non-capturing group for RegEx ... |
| | <code>(?! ...)</code> | Matches if the RegEx ... does not match |
| Special characters | <code>\n</code> | New line |
| | <code>\r</code> | Carriage return |
| | <code>\t</code> | Tab |

RegEx – Groups I

```
1 import re
2
3 m = re.match(r'^H(e|a)llo$', "Hello")
4 print(m.groups()) # ('e',)
5
6 m = re.search(r'H(e|a)llo', "Hallo Malte, Hello Alice,")
7 print(m.groups()) # ('a',)
8
9 m = re.findall(r'H(e|a)llo', "Hallo Malte, Hello Alice,")
10 print(m) # ['a', 'e']
11
12 m = re.findall(r'(H(e|a)llo)', "Hallo Malte, Hello Alice,")
13 print(m) # [('Hallo', 'a'), ('Hello', 'e')]
14
15 m = re.findall(r'H(?:e|a)llo ([^,]+)', "Hallo Malte, Hello Alice,")
16 print(m) # ['Malte', 'Alice']
```

RegEx – Groups II

```
1 mail = re.compile(r'[^@]+@(?:[a-z]*\.)?uni-luebeck\.de')
2 print(mail.match("m.luttermann@uni-luebeck.de")) # match
3 print(mail.match("luttermann@ifis.uni-luebeck.de")) # match
4 print(mail.match("luttermann@FAKEuni-luebeck.de")) # no match
```

RegEx – Groups and Replacements

```
1 import re
2
3 program = """
4 class Train:
5     def __ini__(self, a, b, c):
6         pass
7     def _internal(self):
8         pass
9 class Locomotive():
10    """
11
12 m = re.search(r'class ([a-zA-Z_][0-z]*) (?:(\(\)))?:', program)
13 print(m.group(0), m.group(1))
14
15 print(re.findall(r'class ([a-zA-Z_][0-z]*) (?:(\(\)))?:', program))
16 print(re.findall(r'class ([a-zA-Z_][0-z]*) (\(\))?:', program))
17
18 print(re.sub(r'def __ini__\((self[\^]*)\)':', r'def __init__(\1):', program))
```

```
$> python3 script.py
```

```
class Train: Train
['Train', 'Locomotive']
[(('Train', ''), ('Locomotive', '()'))]
```

```
class Train:
    def __init__(self, a, b, c):
        pass
    def _internal(self):
        pass
class Locomotive():
```

RegEx – Groups and Replacements

```
1 import re
2
3 program = """ Multi-line strings via """
4 class Train:
5     def __ini__(self, a, b, c):
6         pass
7     def _internal(self):
8         pass
9 class Locomotive():
10    """
11
12 m = re.search(r'class ([a-zA-Z_][0-z]*) (?:(\(\)))?:', program)
13 print(m.group(0), m.group(1))
14
15 print(re.findall(r'class ([a-zA-Z_][0-z]*) (?:(\(\)))?:', program))
16 print(re.findall(r'class ([a-zA-Z_][0-z]*) (\(\))?:', program))
17
18 print(re.sub(r'def __ini__\((self[^\)]*)\):', r'def __init__(\1):', program))
```

re.search() searches for the first occurrence

re.findall() searches for all occurrences

re.sub(x, y, z) replaces all occurrences of x in z with y

\1 refers to the first capturing group

Summary

- Development environments
 - Jupyter notebooks
 - Basics (Bash-)terminal
- Regular expressions (RegEx)

We now have program code, documents and notebooks.

Next, we need to manage and organise them all!