Course Overview [1][2]

IT Systems, Summer Term 2026
Dr. Matthes Elstermann

1 Assorted Topics

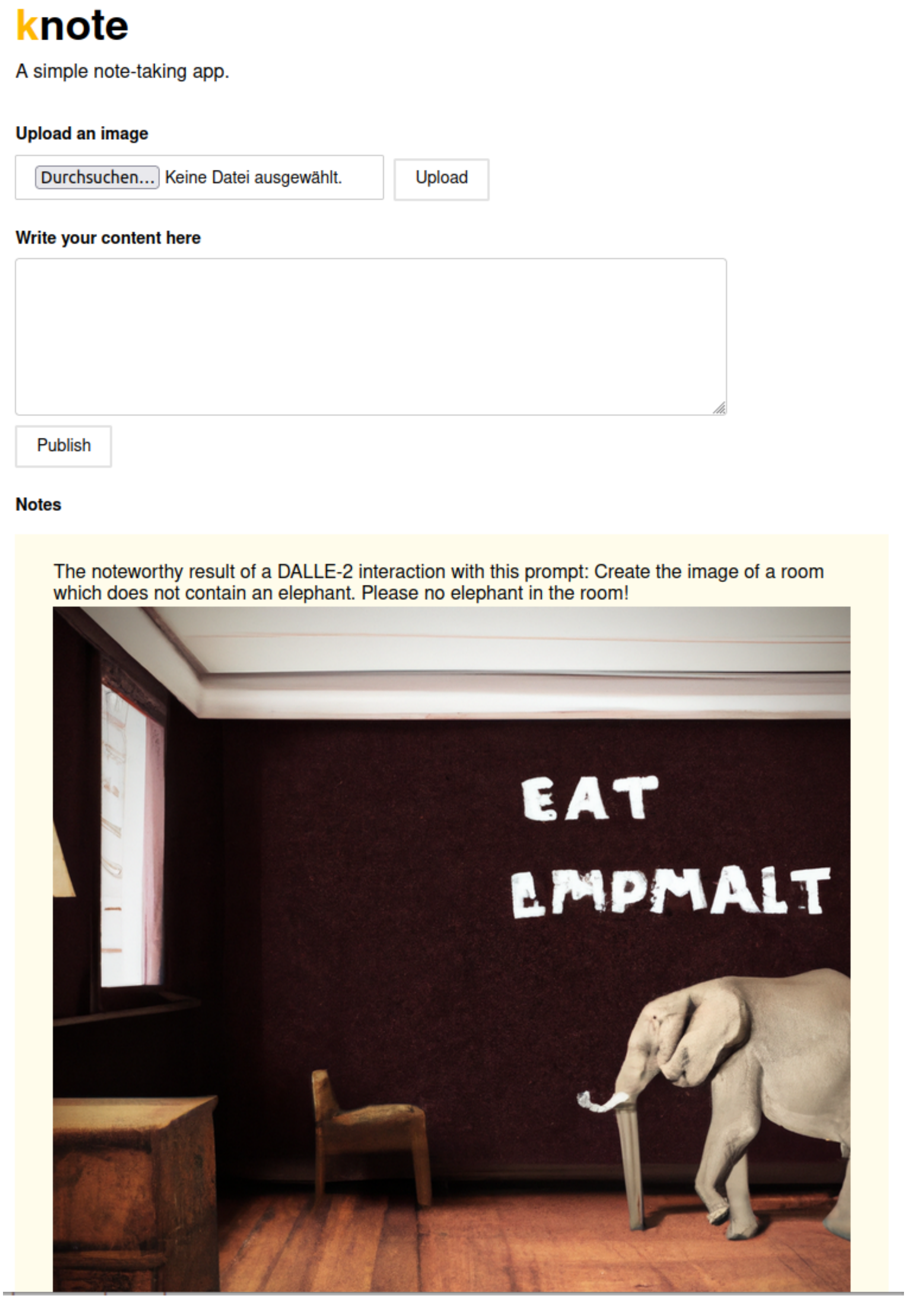
- Fire alarms
  - Keep calm, leave swiftly, but leave no one behind
- IT Systems is the successor module to CSOS
  - CSOS students are very welcome, relevant is Learnweb course of 2023
- eLectures recordings
  - Available if no technical problems, but please use only in exceptional cases
- Exchange students?

2 Motivation

- What do you see?

[1] This PDF document is an inferior version of an OER in HTML format; free/libre Org mode source repository.

[2] Material created by Jens Lechtenbörger; see end of document for license information.



knote

A simple note-taking app.

Upload an image

Keine Datei ausgewählt.

Write your content here

Notes

The noteworthy result of a DALLE-2 interaction with this prompt: Create the image of a room which does not contain an elephant. Please no elephant in the room!

Figure 1: "Screenshot of knote web application" under CC0 1.0; from GitLab

- What might be happening?
  - From a systems' perspective?
  - Abstractions?

This is a screenshot of the note taking application knote that is usable in a web browser and runs "in the cloud".

Clearly, just as every other application, this application requires some computers to do something. Throughout the course, we will revisit this application in a bottom-up fashion as explained next, starting from individual computers over operating systems to cloud environments.

2.1 Course Objectives and Goals

- **Objectives**
  - Discuss how hardware and software systems are built, using **abstraction**, and how they work together
  - What is happening underneath?
    - **Cloud Infrastructure**: Explain basic concepts, deploy simple containerized system
    - **Operating System** (OS): Explain how OSs do their job, use them, inspect what is happening

    - **Computer Architecture**: Build (simulated, yet realistic) computer by breaking task into simpler ones

- **(Long-term) Goals**
  - Inspect and **control** any computer, at any level of interest
  - **Digital sovereignty, sustainability**
    - Knowledge empowers to use/build better solutions that serve our interests
    - E.g., end-of-life for 240 million PCs with Windows 11, more millions when Apple ends support for Intel CPUs
      - Campaign End of 10 offers help with GNU/Linux

2.2 Course at a Glance (1/4)

- Method: Explore **abstractions bottom-up**

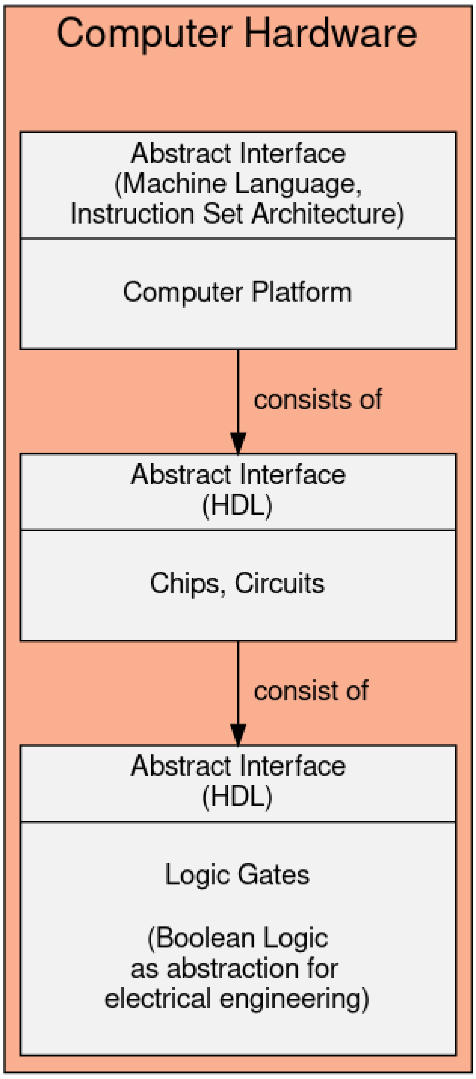


Figure 2: Computer hardware with layers of abstraction

1. **Computer Architecture**
   - **Build** a complete, general-purpose, programmable computer system, called **Hack**, from ground up, starting with elementary logic gates
     - Simulated, Nand2Tetris
     - Sequence of projects
   - Play and experiment with this computer, at any level of interest
   - (Prerequisite: Binary numbers; tutorial with self-tests)

2.3 Course at a Glance (2/4)

- Method: Explore **abstractions bottom-up**

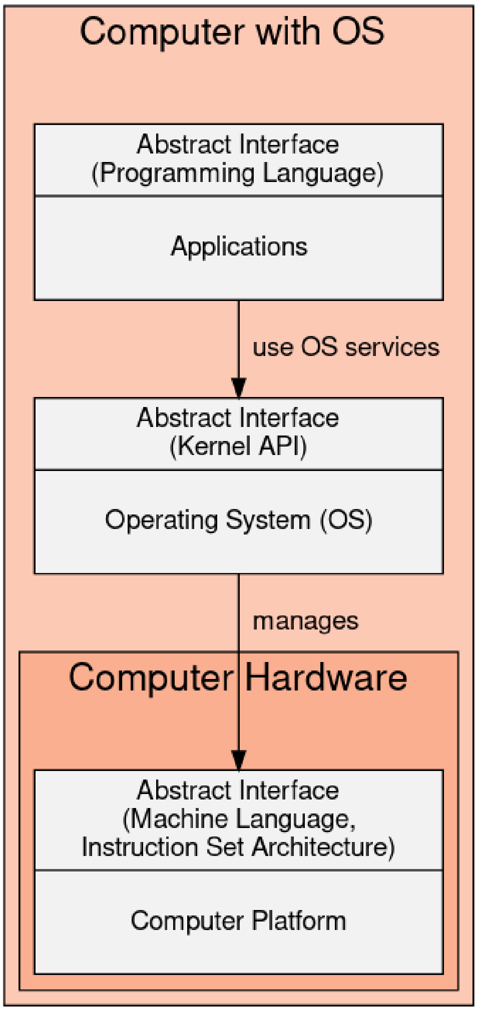


Figure 3: Computer with OS and Kernel API as hardware abstraction

1. Computer Architecture
2. **Experiment with OS concepts**
   - Explain core OS **management** concepts, e.g., processes, threads, virtual memory
   - Use GNU/Linux command line and explore system

Figure 4: “Tux, the Linux mascot” under CC0 1.0; from Wikimedia Commons

     - OS part starts with The Command Line Murders
   - Explore sample Java code

- (Prerequisite: Java programming, compilation, execution)

2.4 Course at a Glance (3/4)

- Method: Explore **abstractions bottom-up**

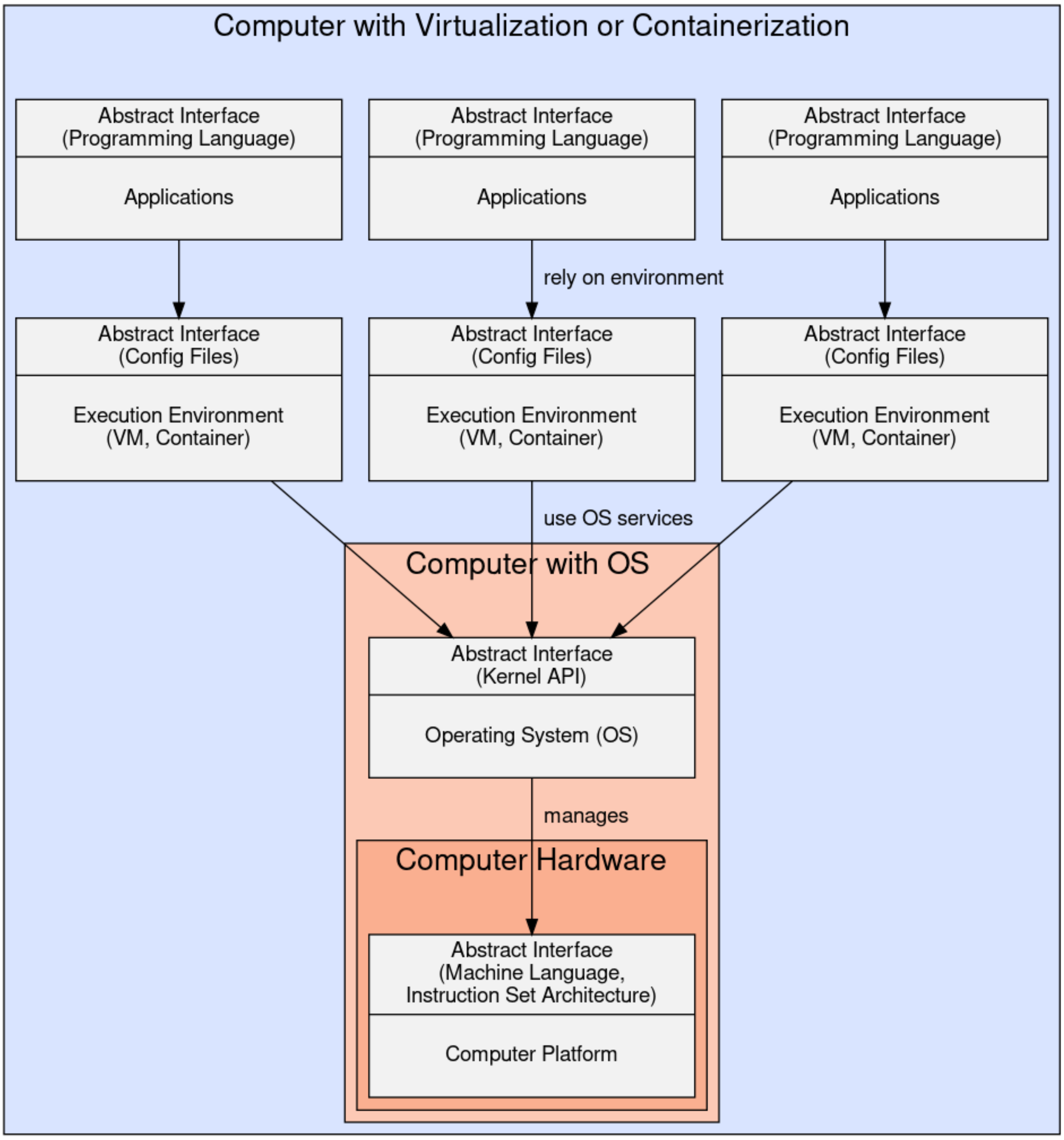

Computer with Virtualization or Containerization
Abstract Interface (Programming Language)
Applications
Abstract Interface (Programming Language)
Applications
Abstract Interface (Programming Language)
Applications
rely on environment
Abstract Interface (Config Files)
Execution Environment (VM, Container)
Abstract Interface (Config Files)
Execution Environment (VM, Container)
Abstract Interface (Config Files)
Execution Environment (VM, Container)
use OS services
Computer with OS
Abstract Interface (Kernel API)
Operating System (OS)
manages
Computer Hardware
Abstract Interface (Machine Language, Instruction Set Architecture)
Computer Platform

Figure 5: Container as abstract execution environment

1. Computer Architecture
2. Experiment with OS concepts
3. **Explain virtualization, experiment with containerization**
   - Explain core concepts
   - Understand images, run Docker **containers**

Figure 6: “Docker logo” under Docker Brand Guidelines; from Docker

- Build and run knote web application seen initially

2.5 Course at a Glance (4/4)

- Method: Explore **abstractions bottom-up**

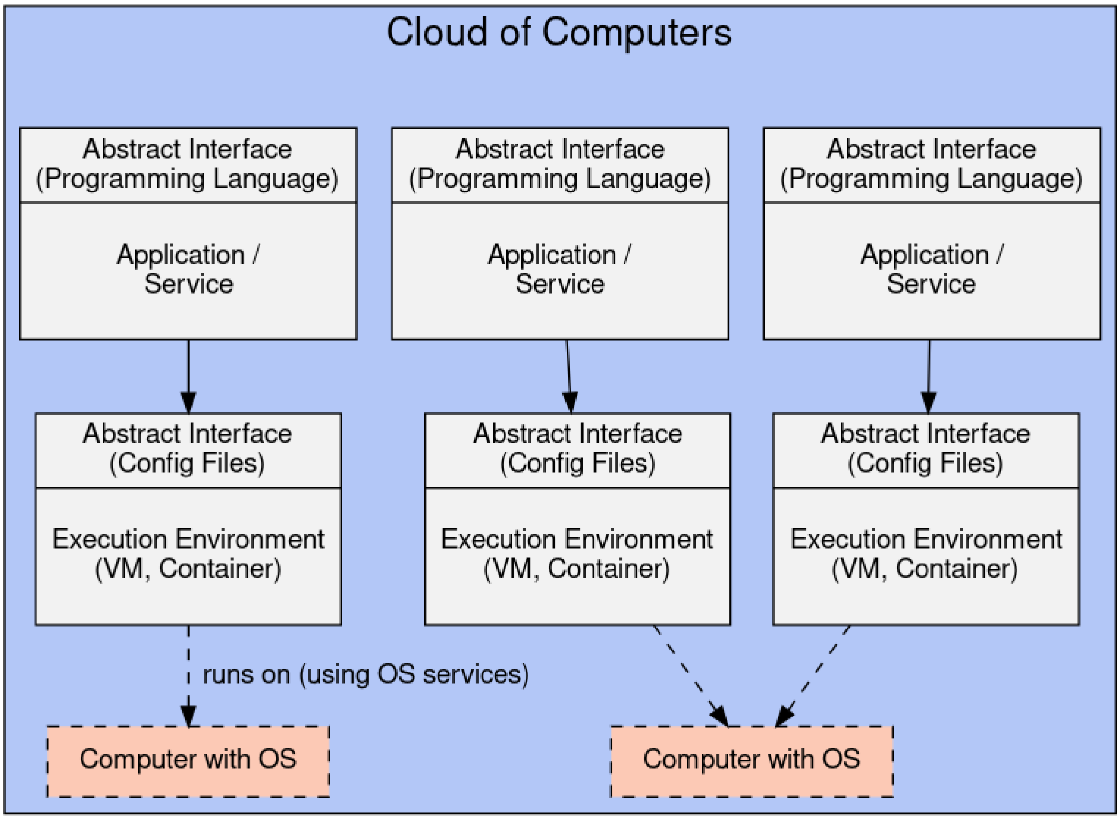

Cloud of Computers
Abstract Interface (Programming Language)
Application / Service
Abstract Interface (Programming Language)
Application / Service
Abstract Interface (Programming Language)
Application / Service
Abstract Interface (Config Files)
Execution Environment (VM, Container)
Abstract Interface (Config Files)
Execution Environment (VM, Container)
Abstract Interface (Config Files)
Execution Environment (VM, Container)
runs on (using OS services)
Computer with OS
Computer with OS

Figure 7: Cloud of computers as abstract execution environment

1. Computer Architecture
2. Experiment with OS concepts
3. Experiment with containerization
4. **Set up simple cloud application**
   - Run Kubernetes cluster on local machine

Figure 8: “Kubernetes logo” under Kubernetes Branding Guidelines; from GitHub

   - Deploy knote web application seen initially

2.6 Learning: Retrieve Taking

- What important topics are we going to cover?
- What do you want to study (maybe on your own)?

3 Course Organization

3.1 Course Components

- Course with 6 CP, at least 8h per week
  - **Joint sessions** in class (no traditional lectures, details below)
    - Tuesday (10:15 a.m.): Recap of lecture material
    - Thursday (2:15 p.m.): Exercises
    - 2x90 minutes = 3h
  - **Self-study**, 5h per week
    - Flipped classroom (details below)
    - Including quizzes in Learnweb: Published as course progresses
      - Self-study quizzes to support your learning
      - Quizzes for **study work** (50% of total points required), **deadlines** on Thursdays (except public holidays, then Friday). Passed study work from earlier term remains valid.
- **Final exam** for 100% of final grade
  - You **must pass both**, study work and exam, for credits
    - Not recommended, but can be done in different terms/years

3.1.1 Registration at Examination Office

- Students are responsible for **registering** for **all required components** of their courses (e.g., study work, examination). Procedures vary by program. If you have questions, contact your student council or the Examinations Office well in advance.
- For the Information Systems (Wirtschaftsinformatik) program, the Examinations Office publishes key exam dates and registration/deregistration deadlines: `https://www.wiwi.uni-muenster.de/pam/en/examinations/schedule-examination-offer-examination-rooms`

3.2 Course Material

- Everything provided in or linked from Learnweb
  - Material developed and published as OER on Gitlab
- Computer Architecture
  - (Nisan and Schocken 2005) The Elements of Computing Systems, MIT Press
    - Book chapters, project material at `https://www.nand2tetris.org/course`
      - (Book chapters hyperlinked from icon for reading person)
    - Gratis course at Coursera
- Operating Systems
  - (Hailperin 2019) Operating Systems and Middleware
- Cloud Infrastructures
  - Variety of papers and software documentation

3.2.1 OER on GitLab

- Presentations such as this one are maintained as Open Educational Resources (OER) on GitLab
  - Sources: `https://gitlab.com/oer/oer-courses/it-systems`
    - Please, contribute with bug reports (issues) or merge requests!
  - Presentations: `https://oer.gitlab.io/oer-courses/it-systems/`
    - Note: **PDF formats**
  - Usage hints: `https://oer.gitlab.io/hints.html`
    - Note: **URL parameters**
      - `https://oer.gitlab.io/oer-courses/it-systems/03-Boolean-Logic-I.html?audio-advance=-1&audio-speed=1.5`
  - Work in progress, presentations "ready" when link in Learnweb

3.3 Tentative Schedule

- April 14/16: Course Introduction
- April 21/23: Boolean Logic
- April 28/30: Combinational Circuits
- May 5: Machine Language
- May 12: Computer Architecture
- May 19/21: OS Introduction
- Pentecost
- June 2: Interrupts and I/O
- June 9/11: Threads and Scheduling
- June 16/18: Mutual Exclusion
- June 23/25: Virtual Memory
- June 30/July 2: Processes
- July 7/9: Virtualization and Containers
- July 14/16: Cloud Computing and Kubernetes
- July 21: Course Recap

3.4 Prerequisites

- We suppose that you can **convert** between decimal, binary, and hexadecimal **numbers**
  - Tutorial with self-tests
- We suppose that you can **program in Java**
  - Including compilation and execution
    - Which requires installation of JDK
- Quickstart with Nand2Tetris software tomorrow (requires JRE)
  - Please install ahead of time and come with Laptop

3.5 Past Course Evaluations and Results

- IT Systems is a new module
  - First incarnation in 2024
    - **Nominated for teaching award** by student council
  - Successor to Computer Structures and Operating Systems (CSOS)
    - CSOS evaluation in 2023 highly positive
- Students usually report that our type of interaction and work is unknown to them

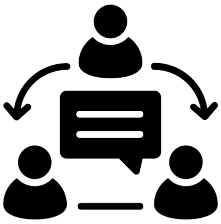


Figure 9: “group discussion” by ProSymbols under CC BY 3.0 US; cropped from the Noun Project

  - Please trust me and overwhelming scientific evidence (alluded to next), and try this out

3.6 Q&A

Figure 10: "Uncovering questions" under CC0 1.0; background changed from Pixabay

4 On Learning and Teaching

4.1 Learning Objectives

- Later presentations contain **Learning Objectives**
  - What we want you to have learned (after lecture and exercises)
- Content + action verb

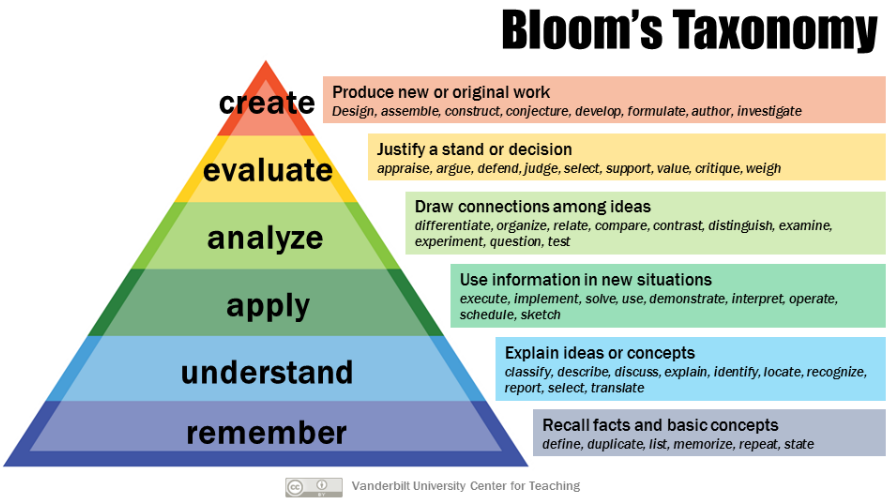

Bloom's Taxonomy
create
evaluate
analyze
apply
understand
remember
Produce new or original work
Design, assemble, construct, conjecture, develop, formulate, author, investigate
Justify a stand or decision
appraise, argue, defend, judge, select, support, value, critique, weigh
Draw connections among ideas
differentiate, organize, relate, compare, contrast, distinguish, examine, experiment, question, test
Use information in new situations
execute, implement, solve, use, demonstrate, interpret, operate, schedule, sketch
Explain ideas or concepts
classify, describe, discuss, explain, identify, locate, recognize, report, select, translate
Recall facts and basic concepts
define, duplicate, list, memorize, repeat, state
Vanderbilt University Center for Teaching

Figure 11: "Bloom's Taxonomy" by Center for Teaching Vanderbilt University under CC BY 2.0; from flickr

  - Action verb specifies level of skill
    - Think of **exam question**!
  - Bloom's taxonomy
- Examples
  - Apply algorithm X in sample scenario

  - Argue about relative strengths and weaknesses of Y and Z
  - Course Objectives on earlier slide

4.2 Learning (1/2)

- Learning
  - Requires **active work**

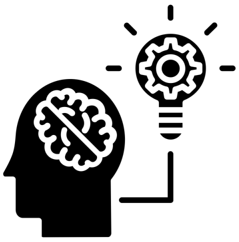


Figure 12: "Brain training" by Shocho under CC BY 3.0 US; cropped from the Noun Project

    - To **change protein structures** in brains, just like muscles
    - E.g., deliberate practice, retrieval practice, spaced repetition
  - Getting information **out of heads**
    - Misconception: Learning = getting information in
    - Precondition for silent learning: **Writing material**
      - Preferably a **laptop** (for writing and experiments)
  - Suggestions to learn about learning
    - Learning Platform Information Systems (**student-driven**, in Learnweb, with videos)
    - See book Make it stick (student **recommendation**!)

4.3 Learning (2/2)

- Consequences
  - During (traditional) lectures, I learn, you do not (much)
    - (Physics Nobel laureate Carl Wieman compares lecturing in education to bloodletting in medicine; both are bad approaches that were popular once)
  - This course provides **learning opportunities** during our meetings
    - Where you can **benefit from my presence**
    - Which requires your **preparation**
    - Which requires **active work** on your part, instead of passive listening
    - (Which may not meet your expectations and may contradict your feeling of learning . . .)
- (My teaching statement justifies the above with scientific references.)

4.4 Flipping IT Systems

- CSOS has been **flipped** since 2017, IT Systems continues in that tradition
  - Since 2023 following (Kapur et al. 2022):
  - **Improved learning outcomes** based on productive failure, active learning, and instructor support: Fail, Flip, Fix, Feed

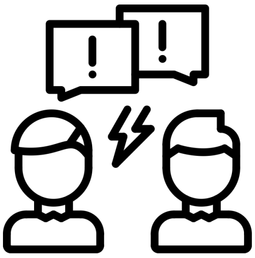


Figure 13: “Conflict” by lastspark under CC BY 3.0 US; cropped from the Noun Project

- **Fail**
  - You attempt to solve a task before being instructed
  - Possibly without success
  - You activate prior knowledge, diagnose own learning, stimulate (meta-) cognitive processes
- **Flip**: You work on self-study material ahead of meeting
- **Fix**, **Feed**
  - Class meetings are shaped by you: **What are *your* goals?**
  - We discuss and work on tasks (“failed” and new ones, exercises and previous exam tasks)
  - This is where I am around and we spend limited, **valuable time**

4.5 Course Rhythm

- On Thursdays
  - Publication of new course material and quiz for study work
    - **Unlocked** when you **submit fail task**
      - Unlocking suggested by students in 2023
    - New tasks on the “Completion Progress” in Learnweb
  - Session

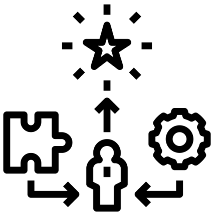


Figure 14: “experience” by Nithinan Tatah under CC BY 3.0 US; cropped from the Noun Project

    - Conclude current topic, **learn**, **work on tasks**
      - Including Q&A on current **study work**
    - Outlook on new material, initial work on **Fail** task
- **Flip**
  - Learn on your own, with self-study tasks
- On Tuesdays
  - Revisit **Fail** task, **Fix** jointly if necessary, **Feed**: **Learn**

4.6 Session Goals

- My goal: Support your learning

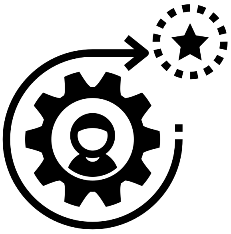


Figure 15: "training" by Nithinan Tatah under CC BY 3.0 US; cropped from the Noun Project

- My major challenge: Heterogeneity regarding knowledge and preparation
  - In particular for flipped classrooms, not so much for lectures
- Your **goals**?

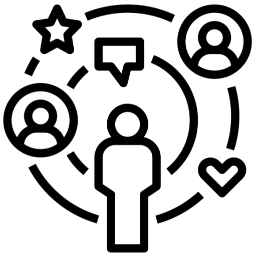


Figure 16: "Society" by Nithinan Tatah under CC BY 3.0 US; cropped from the Noun Project

  - Starting next week, I will ask you for your goals
    - Different students may work towards different goals
      - Individually or in small groups
      - See upcoming pads for suggestions; feel free to **add own goals**
    - I will be around to help

4.7 Your Thoughts

- **Anonymous pads** in Learnweb for our sessions

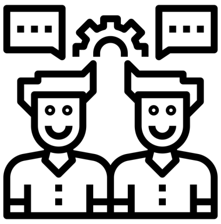


Figure 17: "dialogue" by Template under CC BY 3.0 US; cropped from the Noun Project

  - For your input and notes as well as my session plans
- My questions
  - Why did you enrol in a presence university?
    - Why do you attend sessions? On Campus?
    - Why would you **like** to come to campus?
    - Is everything fine as it is?
      - Proposed rhythm?
    - What stresses you? What brings you joy?
  - How **should** learning at a presence university look like?
    - What are your and my roles?

5 Conclusions

- Let's learn
  - What are the two most important aspects that you take away from this session?

IT Systems investigates how hardware and software systems are built, using **abstraction**, in a **bottom-up** fashion:

- Build computers
- Explore OS
- Experiment with containers and container orchestration

We will use a flipped classroom approach, which might not meet your expectations but which is based on scientific evidence regarding learning.

Your instructor appreciates feedback and discussions.

Bibliography

Hailperin, Max. 2019. *Operating Systems and Middleware – Supporting Controlled Interaction.* revised edition 1.3.1. https://github.com/Max-Hailperin/Operating-Systems-and-Middleware--Supporting-Controlled-Interaction.

Kapur, Manu, John Hattie, Irina Grossman, and Tanmay Sinha. 2022. "Fail, Flip, Fix, and Feed – Rethinking Flipped Learning: A Review of Meta-Analyses and a Subsequent Meta-Analysis." *Frontiers in Education* 7. https://doi.org/10.3389/feduc.2022.956416.

Nisan, Noam, and Shimon Schocken. 2005. *The Elements of Computing Systems: Building a Modern Computer from First Principles.* The MIT Press. https://www.nand2tetris.org/.

The bibliography contains references used in this presentation.

License Information

This presentation is distributed as Open Educational Resource under freedom granting license terms.

Source files are available on GitLab, where the author would be happy about contributions, e.g., in terms of issues and merge requests.