

Non-Research Tips for Information Science Researchers (Summer 2026)

Apr 15, 2026

Week 1: Introduction and Review of Fundamental Concepts



Yusuke Matsui
(UTokyo)



Koya Narumi
(Keio Univ.)



Yuki Koyama
(UTokyo)



Jun Kato
(AIST)

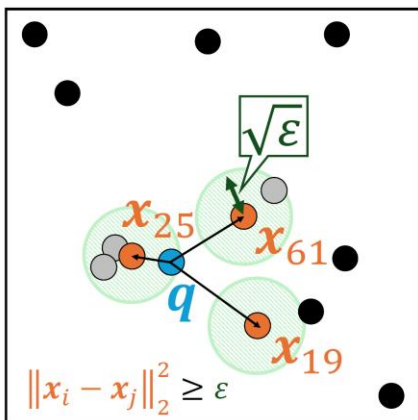


Yusuke Matsui

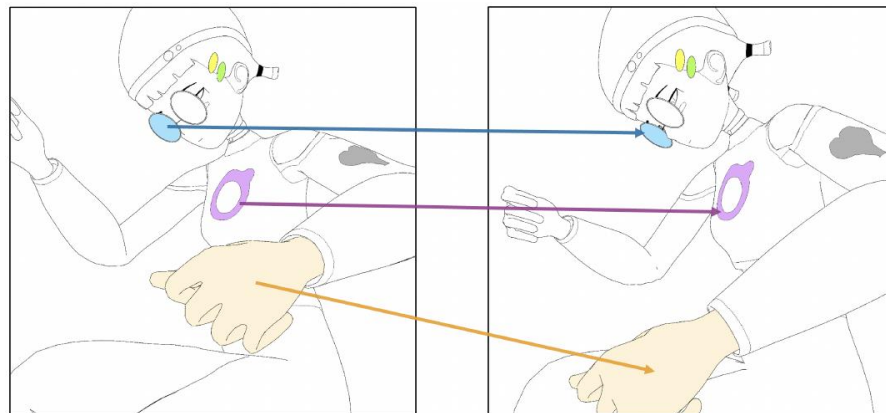
Senior Assistant Professor, the University of Tokyo, Japan

🌐 <http://yusukematsui.me> 🐦 @utokyo_bunny 🐙 @matsui528

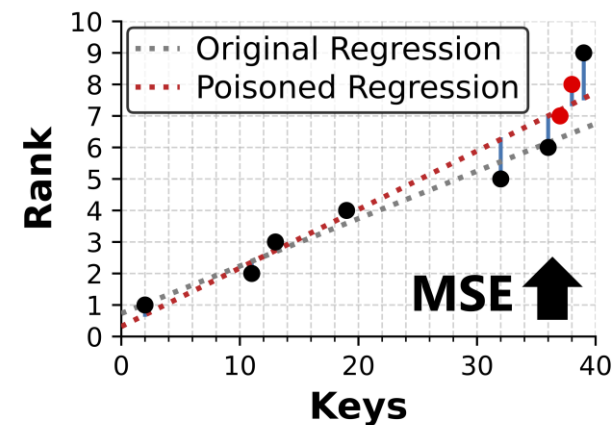
- ✓ Computer vision, database, machine learning
- ✓ RAG, vector databases, learned data structures



Diverse search
[Matsui, CVPR 25]



Correspondence for manga regions
[Li+, CVPR 26]



Poisoning attack to learned index
[Sato+, SIGMOD 26]

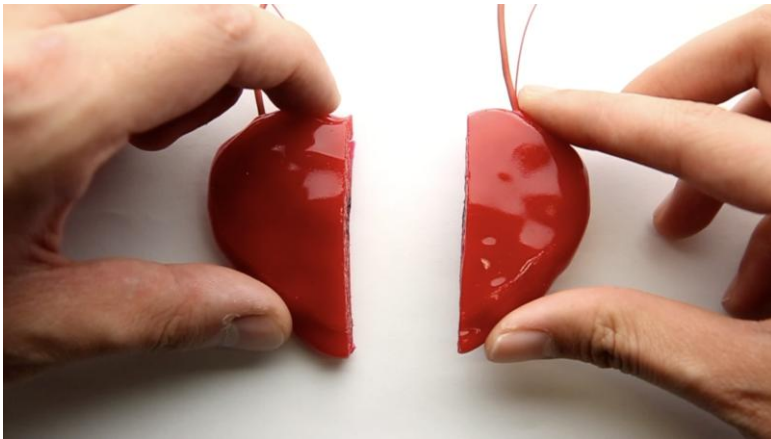


Koya Narumi Currently in Barcelona, Spain

Associate Professor, Keio University, Japan

<https://narumi.me/> [@koya_narumi](https://twitter.com/koya_narumi)

- ✓ Digital Fabrication
- ✓ Shape-changing Interfaces
- ✓ Human-Computer Interaction



Self-healing UI

[Narumi+, UIST 18]



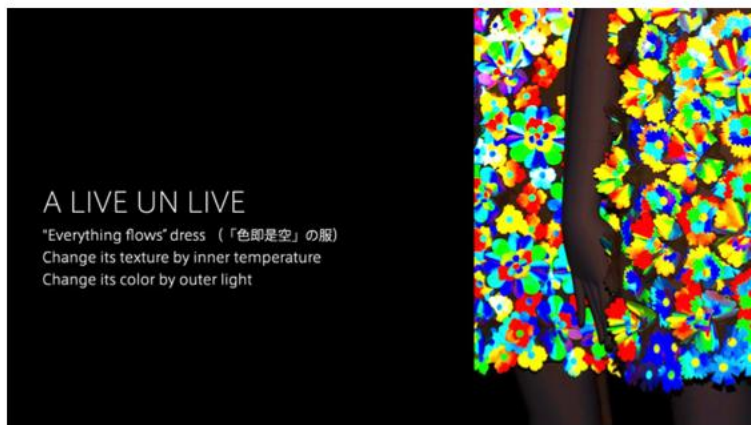
Inkjet 4D Print

[Narumi & Koyama+, SIGGRAPH 23]

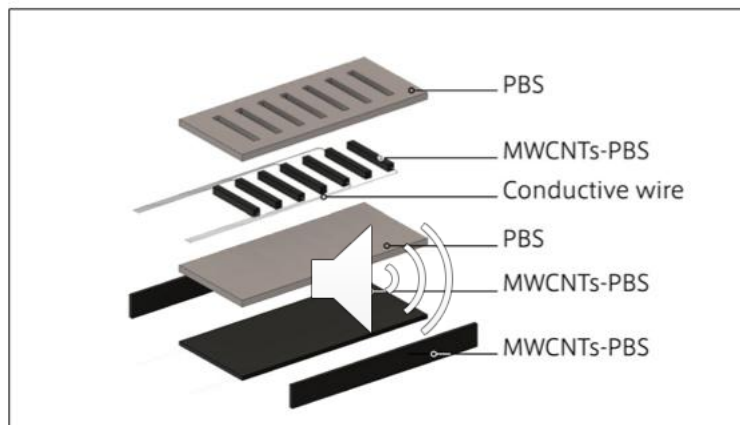


TYPE-X

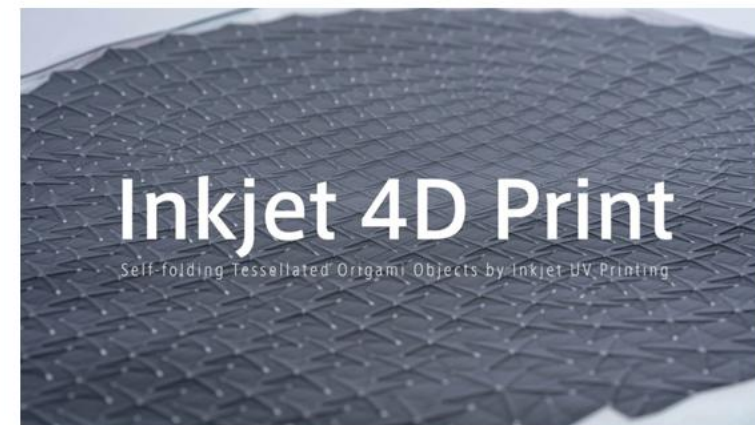
[A-POC ABLE ISSEY MIYAKE]



Slides



Figures



Videos

Koya will cover core visual components for research: **Slides, Figures, and Videos.**




Yuki Koyama

Associate Professor, Precision Engineering, UTokyo, Japan

 <https://koyama.xyz>

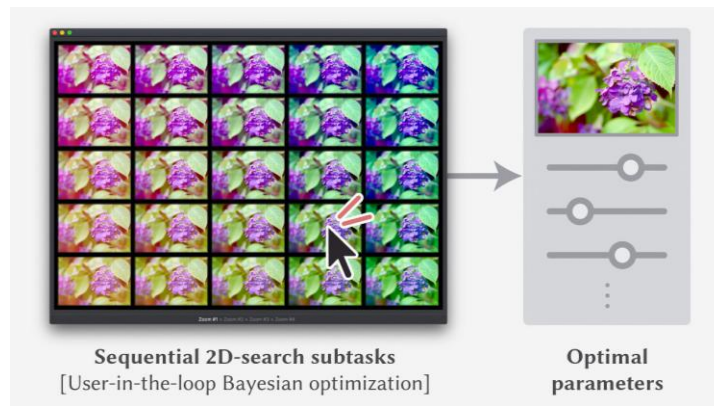
 @bravery_

 @yuki-koyama

- ✓ Computer graphics (and anime production)
- ✓ Human-computer interaction
- ✓ Human-in-the-loop design optimization



Design suggestion
[Koyama+, UIST 2022]



Interactive design
optimization
[Koyama+, SIGGRAPH 2020]

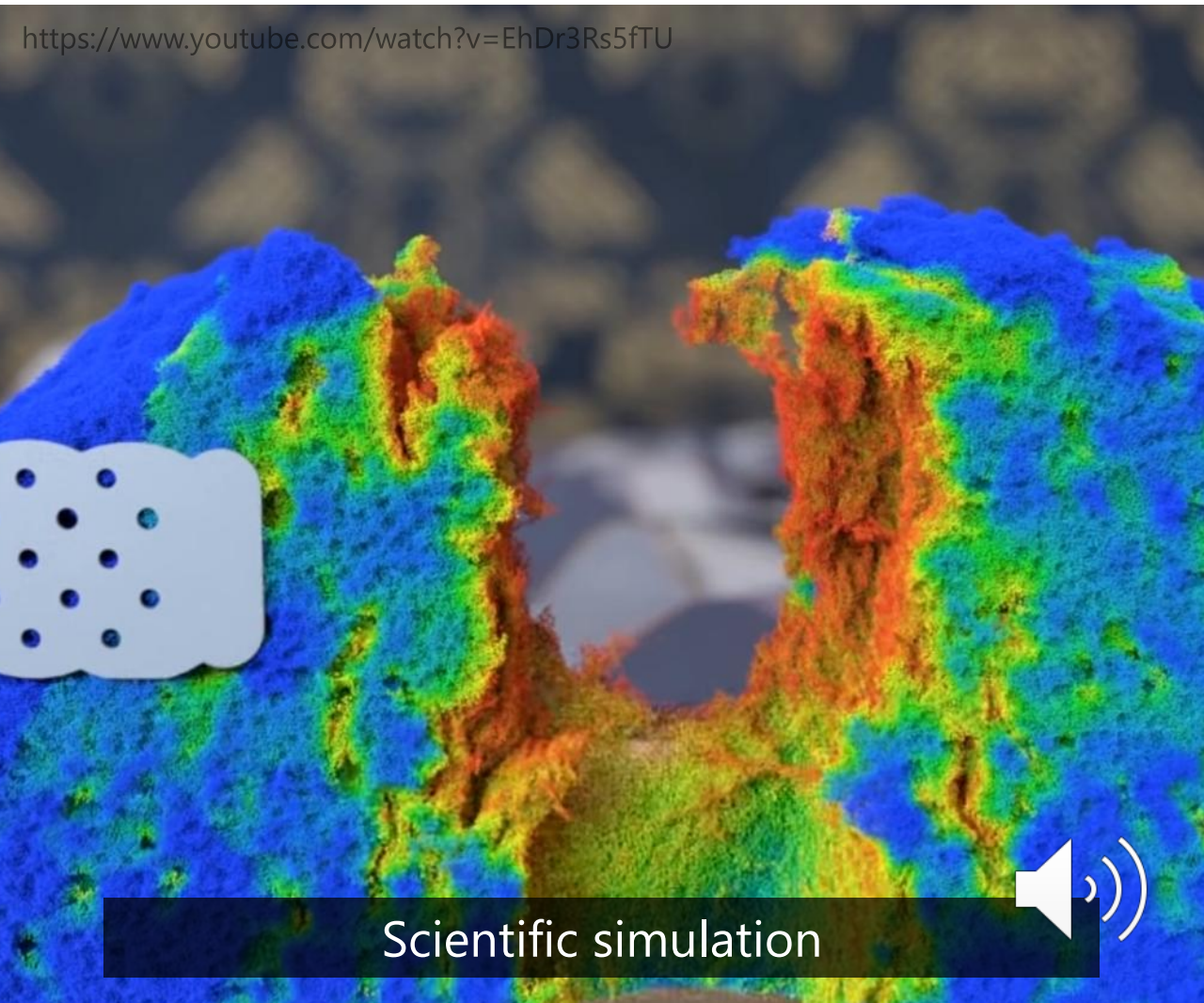
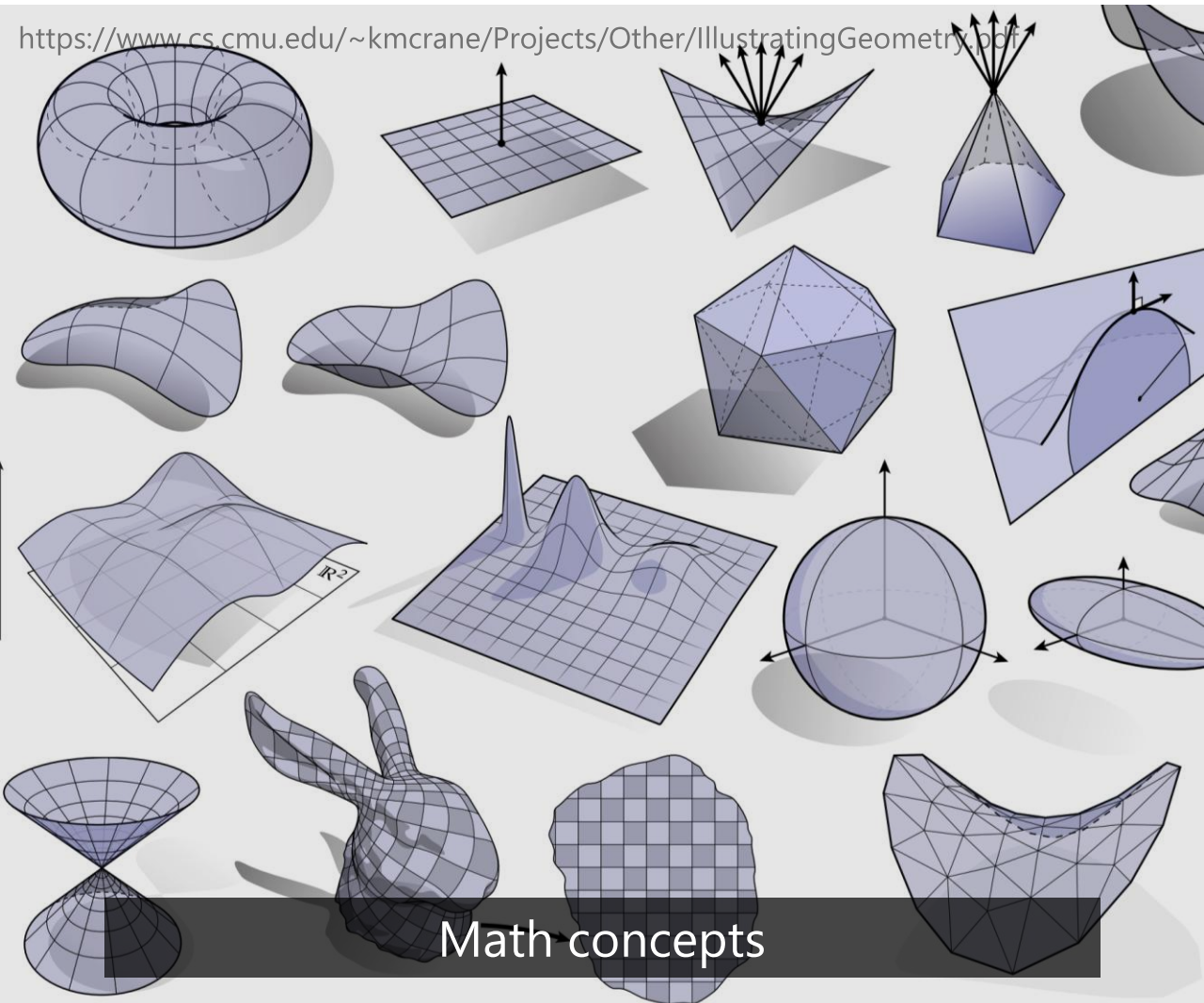


Stylized 3D animation
[Todo+, SA 2024 TComm.]



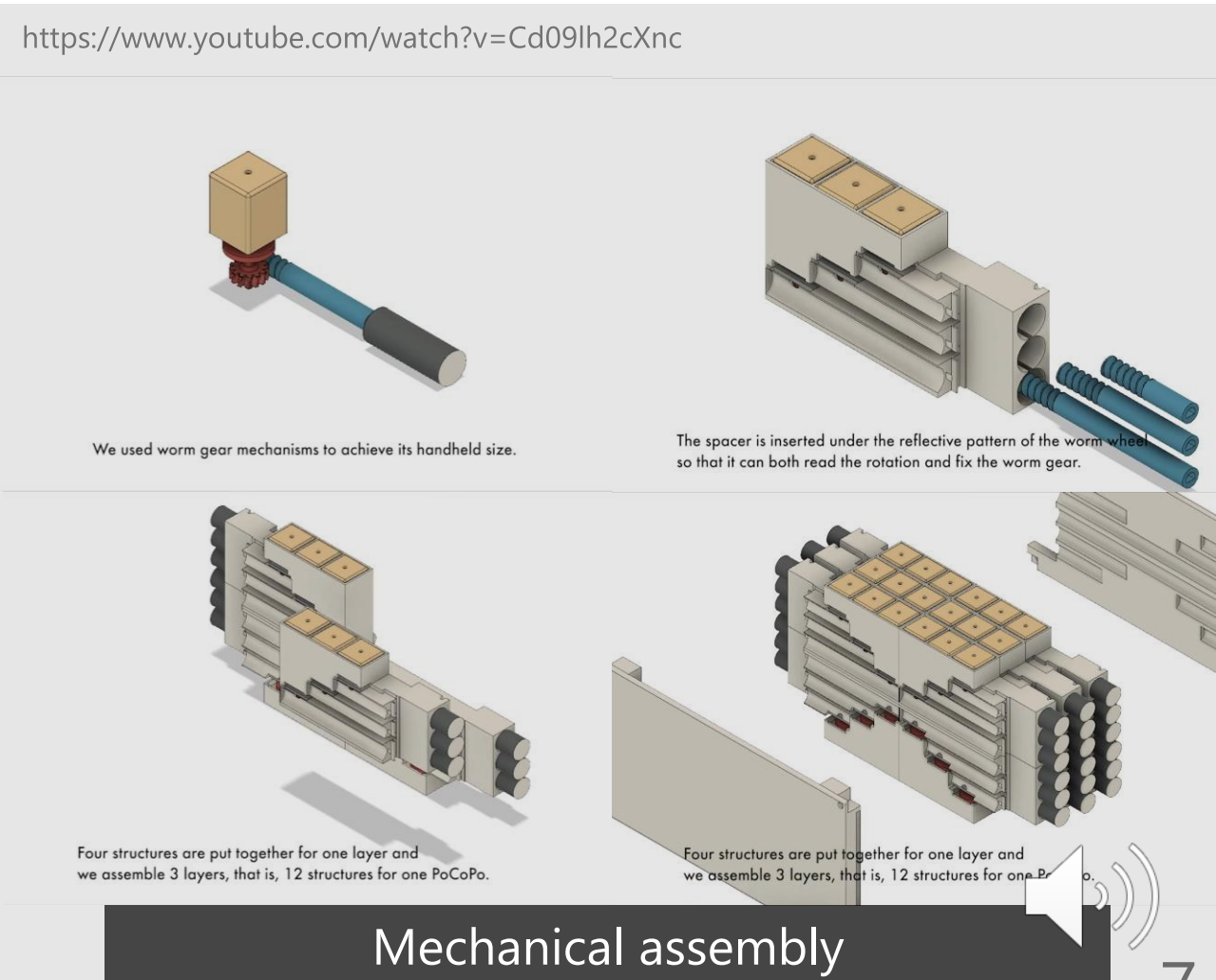
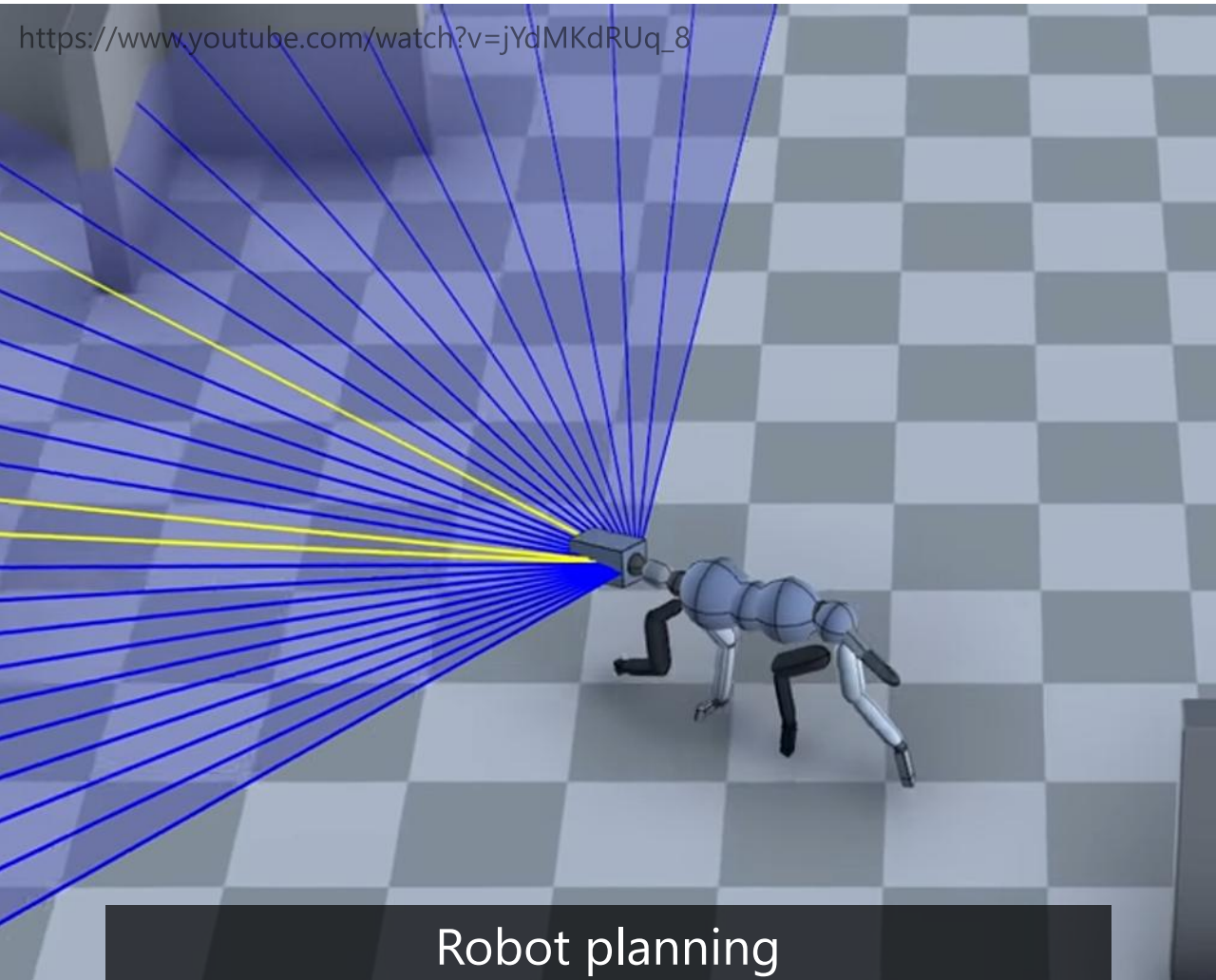
The topic that I will cover in this course: **3DCG illustrations**

Good illustrations help people better understand your research!!
3DCG is sometimes an effective technique for this purpose.



The topic that I will cover in this course: **3DCG illustrations**


Good illustrations help people better understand your research!!
3DCG is sometimes an effective technique for this purpose.





Jun Kato

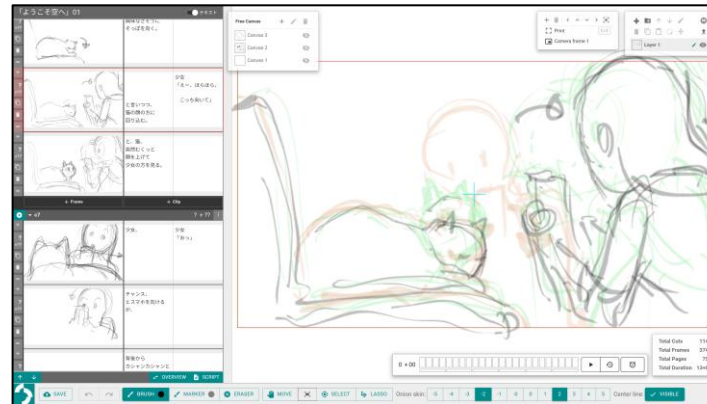
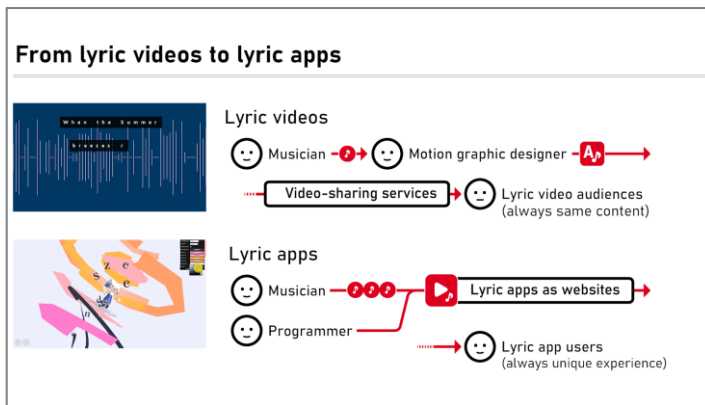
Chief Senior Researcher, AIST / Technical Advisor, Arch Inc, Japan

 <https://junkato.jp>

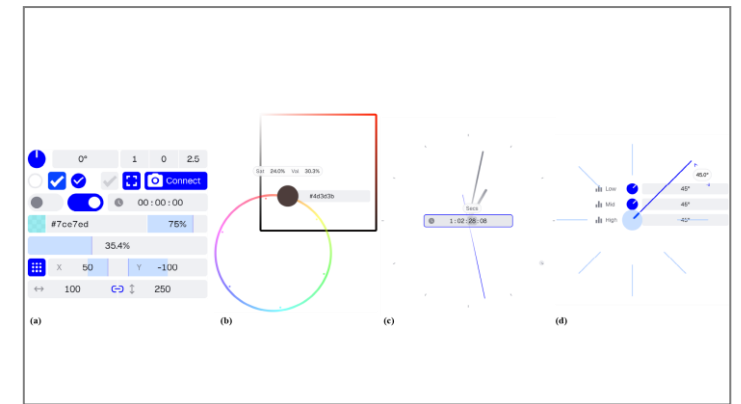
 @junkato

 @arcatdmz

- ✓ Human-Computer Interaction
- ✓ Creativity Support (“Toolsmith researcher” for programmers, musicians, anime creators, etc.)



Storyboarding tool “Griffith”
[Kato+, ACM CHI '24]



GUI widgets “Tweeq”
[Hashimoto & Kato, ACM UIST '25]

“Lyric app” dev framework
[Kato+, ACM CHI '23]

Study

e.g., Attending tutorials

Help

Research

- Reading papers
- Proposing methods
- Coding
- Writing papers
- Etc...

Help

Discussion

e.g., Visiting labs

Help

Lectures

e.g., Parallel Computing

Study

e.g., Attending tutorials

Help

Research

- Reading papers
- Proposing methods
- Coding
- Writing papers
- Etc...

Help

Lectures

e.g., Parallel Computing

Discussion

e.g., Visiting labs

Help

Underpin

Important but not taught knowledge

e.g., writing pseudo code, creating a demo video, and managing a research community

- Our target! "**Non-research tips**"
- By mastering these skills, you can focus more on your research itself

Information

- Dates: Wednesday, 2nd period (10:25 - 12:10)
- Location: Faculty of Engineering Bldg. 2 #241 (This room)
- Style: The lecture will be held in-person. No streaming is available.
- (Japanese materials may also be distributed.)

Assessment

- Final presentations

Schedule

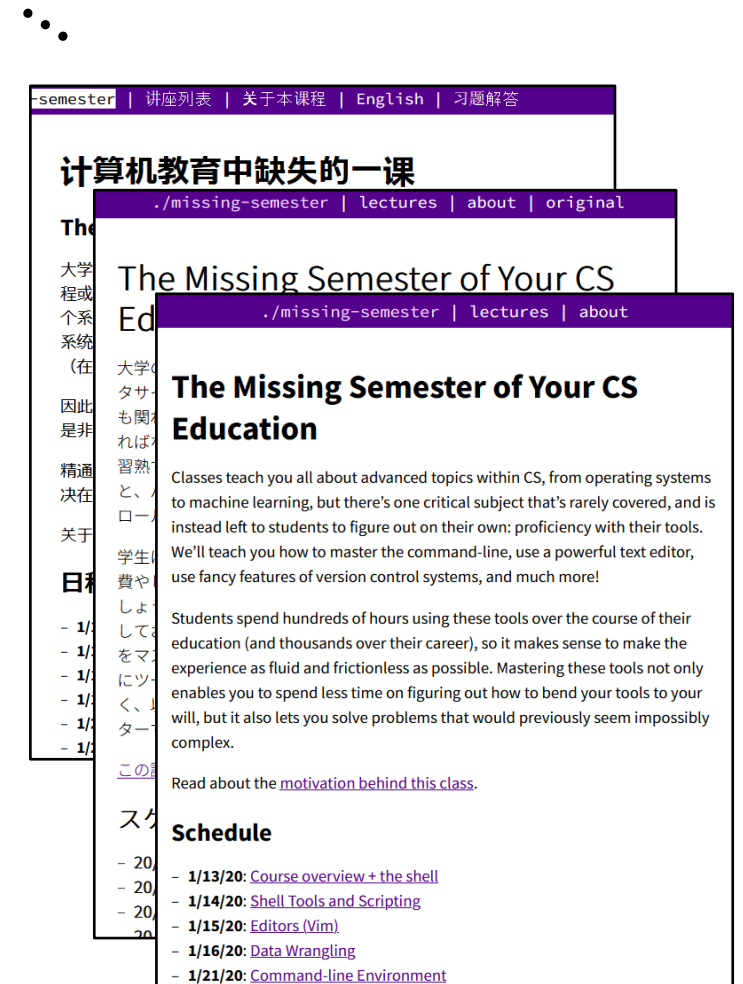
Date (2026)	Contents	Presented by
Week 1, Apr 15	Introduction. Review of fundamental concepts	Yusuke, Koya, Yuki, Jun
Week 2, Apr 22	Equations and pseudo-codes	Yusuke Matsui
Week 3, May 13	Slides	Koya Narumi
Week 4, May 20	Research community	Jun Kato
Week 5, May 27	Figures	Koya Narumi
Week 6, June 1	3DCG illustrations	Yuki Koyama
Week 7, June 10	Invited Talk 1: Research tips in the private sector: From topic selection to social implementation (temp.)	Dr. Antonio Tejero de Pablos (CyberAgent)
Week 8, June 17	Tables and plots	Yusuke Matsui
Week 9, June 24	Invited Talk 2: Portable and Reproducible Research Environments in the Age of AI Agents	Dr. Mai Nishimura (OSX)
Week 10, July 1	DevOps for research	Jun Kato
Week 11, July 8	Videos	Koya Narumi
Week 12, July 15	Final presentations	YOU



Prerequisites


The Missing Semester of Your CS Education

- Short lecture series at MIT
 - ✓ <https://missing.csail.mit.edu/>
- Similar concept to us
 - ✓ Fundamental technical tools
 - ✓ Shell, command-line, git, ...
- Translated into several languages
 - ✓ Matsui's team translated into Japanese
- Be sure to read it in advance



Website

- <https://non-research-tips.github.io/2026.html>
- We'll upload all materials
- Cancellations, administrative notices, etc., will be announced here



The screenshot shows the website interface for 'non-research-tips'. The header includes the site name and a search bar. The main content area displays the course title '4840-1055: Non-Research Tips for Information Science Researchers / 情報科学研究補助技法 (Summer 2026)'. Below the title is a paragraph describing the lecture's location and target audience. A 'News' section contains two bullet points: 'Feb 23: IMPORTANT: There will be no class on April 8. The first class will start on April 15.' and 'Feb 3: The website is now online.' The 'Overview' section is partially visible at the bottom.

non-research-tips

Q Search non-research-tips

Summer 2026

Summer 2024

4840-1055: Non-Research Tips for Information Science Researchers / 情報科学研究補助技法 (Summer 2026)

This lecture is offered at the Graduate School of Information Science and Technology, Department of Information and Communication Engineering, the University of Tokyo. This lecture is tailored for graduate students majoring in information science.

News

- Feb 23: **IMPORTANT:** There will be no class on April 8. The first class will start on April 15.
- Feb 3: The website is now online.

Overview

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

Agenda

- **Git / GitHub (basic usage)**
- Make
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

Git / GitHub

➤ Git

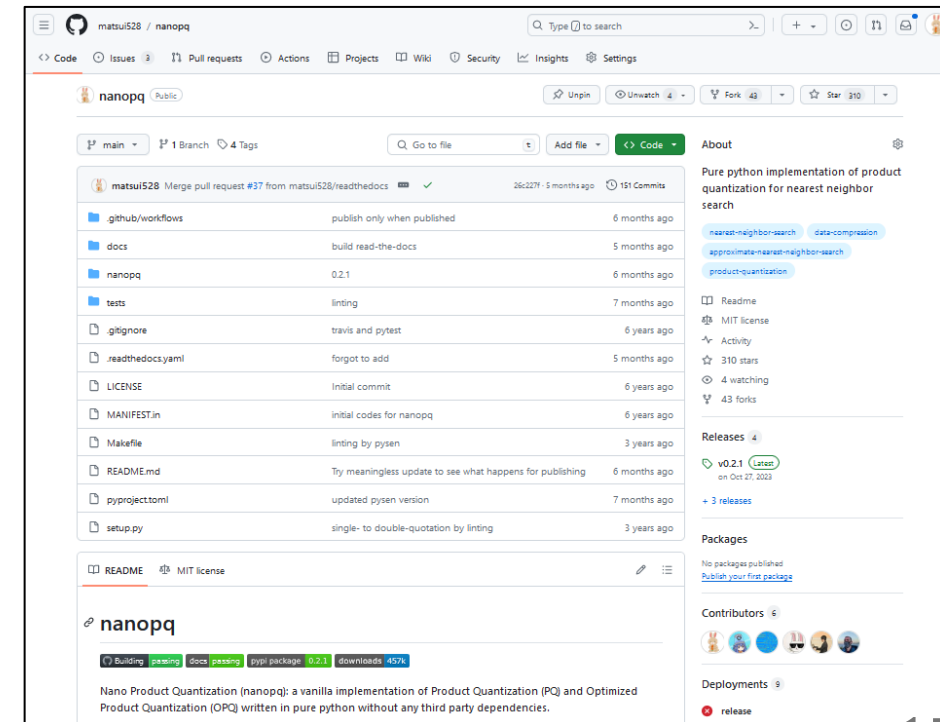
- ✓ Command for version control: `$ git add xxx.py`

➤ GitHub

- ✓ Narrow sense: Webservice to host repositories
- ✓ Similar services: Bitbucket, GitLab, ...

➤ What you need to know

- ✓ Backup your code
 - ❑ git add & commit & push
 - ❑ You **MUST** backup your codes!
- ✓ Collaborations
 - ❑ fork & pull requests & review



Git / GitHub

- Resources (English)
 - ✓ Pro Git: <https://git-scm.com/book/en/v2>
 - ✓ Learn Git @GitKraken: <https://www.gitkraken.com/learn/git>
- Resources (Japanese)
 - ✓ GitHub Lecture by Prof. Hiroshi Watanabe
<https://github.com/kaityo256/github>
 - ✓ 渡辺 宙志, "ゼロから学ぶGit/GitHub 現代的なソフトウェア開発のために", 講談社, 2024
<https://www.kspub.co.jp/book/detail/5352199.html>
- The easiest way: Ask your friends!

Agenda

- Git / GitHub (basic usage)
- **Make**
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

Make

- General purpose workflow description (not just compiling c/c++)
- Manual: <https://www.gnu.org/software/make/manual/make.html>
- Makefile highlights:
 - ✓ What users can (should) do
 - ✓ The order of operations
- The only tool everyone can understand
 - ✓ 🤔 cmake? Bazel? Meson?
- Can be used for **general** purposes

GNU make

Next: [Overview of make](#), Previous: [\(dir\)](#), Up: [\(dir\)](#) [[Contents](#)][[Index](#)]

GNU make

This file documents the GNU `make` utility, which determines automatically which pieces of a large program need to be recompiled, and issues the commands to recompile them.

This is Edition 0.77, last updated 26 February 2023, of The GNU Make Manual, for GNU `make` version 4.4.1.

Copyright © 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover Texts being "A GNU Manual," and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled "GNU Free Documentation License."

(a) The FSF's Back-Cover Text is: "You have the freedom to copy and modify this GNU manual. Buying copies from the FSF supports it in developing GNU and promoting software freedom."

Table of Contents

- [1 Overview of `make`](#)
 - [1.1 How to Read This Manual](#)
 - [1.2 Problems and Bugs](#)
- [2 An Introduction to Makefiles](#)
 - [2.1 What a Rule Looks Like](#)
 - [2.2 A Simple Makefile](#)

Agenda

- Git / GitHub (basic usage)
- Make
- **Docker / Singularity**
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

Docker / Singularity



<https://www.docker.com/ja-jp/company/newsroom/media-resources/>

- A de facto standard tool for virtualization (container)

- In modern software engineering
 - ✓ One would like to perfectly use the same runtime environment even 1 month later, on a different OS, on different hardware
 - ✓ Virtualization is the only option; Docker won the *VM war*

- In research
 - ✓ Reproducibility is the key
 - ✓ Docker provides the easiest way to reproduce your results

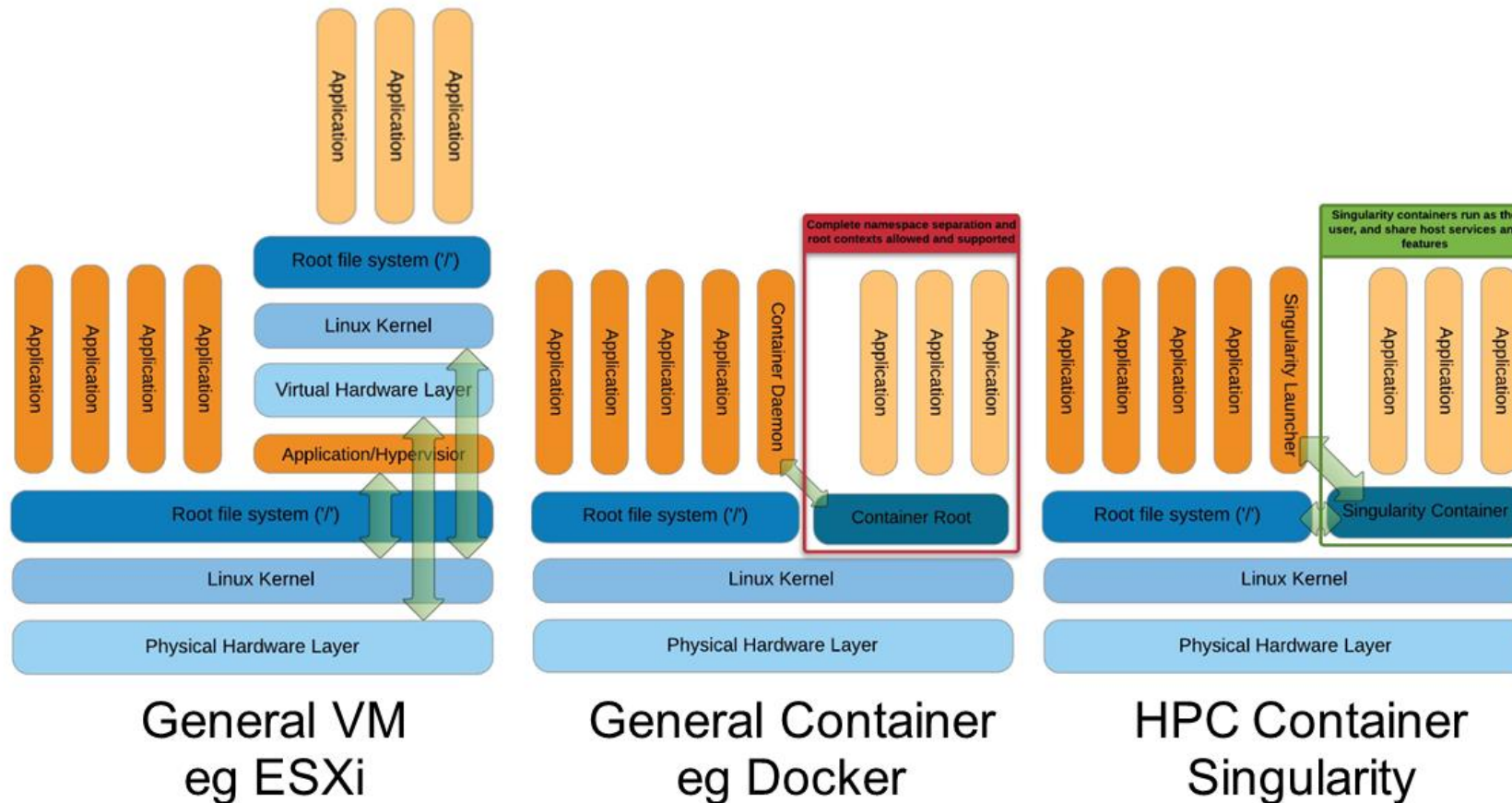
- Docker Desktop is freely available for students for research purposes
 - ✓ Not free for a big company. Be careful!

Docker / Singularity

- tldr; Docker for HPC
- The default option for ABCI is Singularity



<https://sylabs.io/docs/>



Source: [Greg Kurtzer keynote at HPC Advisory Council 2017 @ Stanford](#)

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- **uv/pixi**
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

uv / pixi

- Fast python package & project manager
- The Python environment setup war is **over**

- If you need just standard Python stuff

- ✓ Use uv



<https://github.com/astral-sh/uv>

- If you need more (e.g., c++)

- ✓ Use pixi



<https://prefix.dev/>

- That's it!

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- uv/pixi
- **Markdown (and structured text description)**
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

Markdown (and structured text description)

```
# This is H1 title

## This is H2 title
usual text

- list item 1
- list item 2
- list item 3

this is [link](https://www.google.com)
```

README.md

Render



This is H1 title

This is H2 title

usual text

- list item 1
- list item 2
- list item 3

this is [link](#)

- Programmable (version-controllable), structured text
- Computers like Markdown; easily reusable for later purposes
 - ✓ e.g., convert to tex, pdf, html, etc...
- Structured texts are important to communicate with others

Markdown (and structured text description)

- Advanced usage for GitHub markdown
 - ✓ [Copy & paste images](#)
 - ✓ [Diagrams](#)
 - ✓ [Equations](#)
 - ✓ [Alerts](#)

Markdown (and structured text description)

- Recommend: write google-doc and MS word as if it's markdown

If you use "header" properly, the index is constructed properly 😊

要約

概要

This is H1 title

This is H2 title

This is H1 title

This is H2 title

usual text

- list item 1
- list item 2
- list item 3

this is [link](#)

Use a list functionality to create a list

Don't change the "visual" thing. The text should be structured.

Don't make a header by changing the font size!

☹️

This is H1 title

This is H2 title

usual text

- list item 1
- list item 2
- list item 3

- Don't add "spaces"
- Don't create your own bullets

Why bad? It's hard to maintain by others!

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- **Notebook environment**
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- UTokyo services

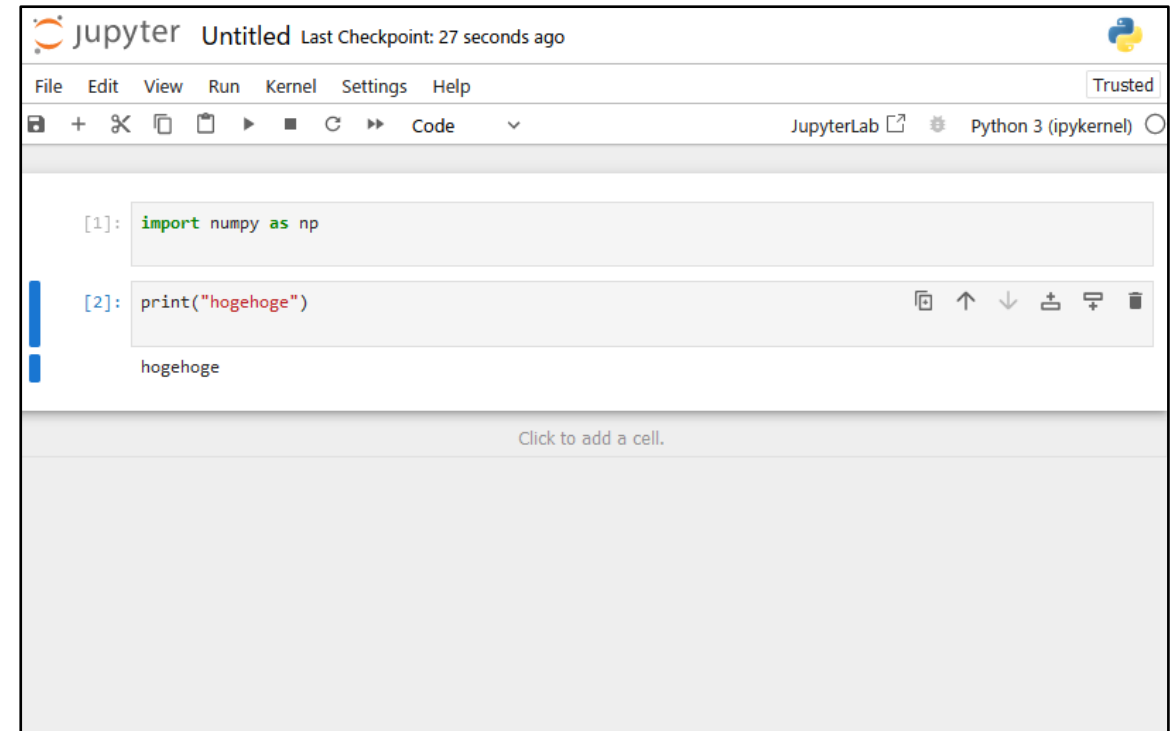


- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

Notebook environment



Google Colaboratory

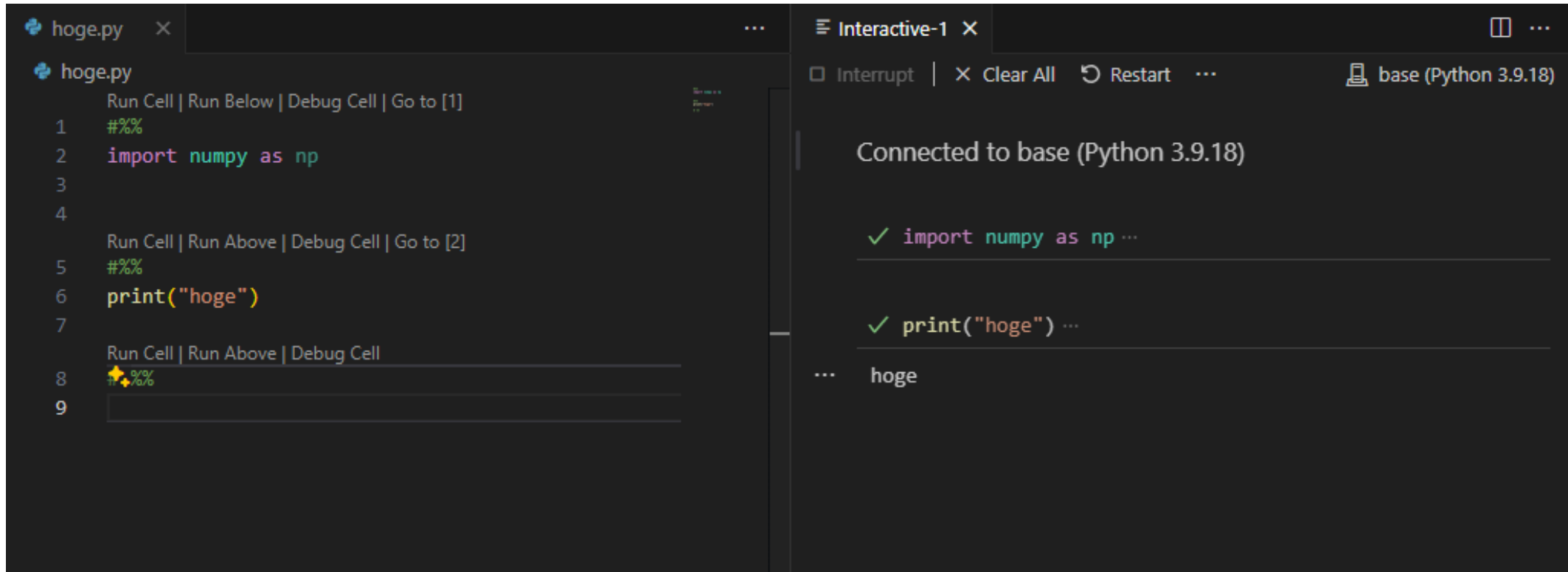


Jupyter notebook

- Some people may argue that notebooks are not useful
- However, it's beneficial to at least understand what a notebook is

Notebook environment

- Python interactive mode: sometimes useful (not .ipynb, but .py)
- Version-controllable (for source codes only)



The image shows a code editor interface with two main panels. The left panel displays a Python file named 'hoge.py' with the following code:

```
1  #%%  
2  import numpy as np  
3  
4  
5  Run Cell | Run Above | Debug Cell | Go to [2]  
6  #%%  
7  print("hoge")  
8  
9  Run Cell | Run Above | Debug Cell  
10 #%%  
11
```

The right panel shows an interactive terminal window titled 'Interactive-1' connected to 'base (Python 3.9.18)'. It displays the execution of the code from the left panel:

```
Connected to base (Python 3.9.18)  
  
✓ import numpy as np ...  
  
✓ print("hoge") ...  
  
... hoge
```

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- **LaTeX / Overleaf**
- Mental model for computation
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

LaTeX / Overleaf

- You must use LaTeX. Overleaf is a popular tool for collaborative writing
- You can back up data by git or GitHub. You must do it!

The screenshot shows the Overleaf LaTeX editor interface. The 'Menu' icon in the top left is circled in red. A red arrow points from the 'Menu' icon to the 'Download' dropdown menu, which is also open. In the 'Download' menu, the 'Git' and 'GitHub' options are circled in red. The main editor area shows LaTeX source code for a document titled 'icassp2022'. The right sidebar contains a 'Recompile' button and a 'Chat' window. The bottom part of the image shows a diagram comparing three asymmetric distance computation (ADC) lookup operations: (a) ADC with $K=256$, (b) ADC with 128-bit SIMD register, and (c) Proposed ADC with 128x2-bit SIMD registers on ARM. The diagram includes mathematical expressions and flowcharts illustrating the data flow and operations.

Fig. 1: Comparison of asymmetric distance computation (ADC) lookup operations

Given an input vector, $\mathbf{x} \in \mathbb{R}^D$, let us define a vector quantizer, $Q: \mathbb{R}^D \rightarrow \{1, \dots, K\}$, as follows:

$$Q(\mathbf{x}) = \underset{k \in \{1, \dots, K\}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{c}_k\|_2^2. \quad (1)$$

Here, $\{\mathbf{c}_k\}_{k=1}^K \subset \mathbb{R}^D$ provides codewords, which are typically created by running k -means clustering over the training dataset. Note that K is the number of codewords. With Q , a D -dimensional vector is quantized into a short code (integer) k , which can be represented by only $\log_2 K$ bits. Here, K is usually set to 256, so that each code takes $\log_2 256 = 8$ bits ≈ 1 B (unsigned char). Given k , we can lossily-reconstruct the original vector, \mathbf{x} , by fetching the codeword, \mathbf{c}_k .

We apply Q for each database vector, \mathbf{x}_n , which results in the quantized codes, $\{k_n\}_{n=1}^N$, where $k_n = Q(\mathbf{x}_n) \in \{1, 2, \dots, K\}$. Here, we discard the original database vectors, $\{\mathbf{x}_n\}_{n=1}^N$, and maintain only $\{k_n\}_{n=1}^N$. Compared with the original vectors, which require $2ND$ bits, the quantized codes require only $N \log_2 K$ bits.

Given \mathbf{q} , our task is to find a similar item from the quantized codes, $\{k_n\}_{n=1}^N$. A straightforward approach is to reconstruct all vectors, $\mathbf{c}_{k_1}, \mathbf{c}_{k_2}, \dots, \mathbf{c}_{k_N}$, and run a linear scan, but this requires considerable time and memory. Alternatively, we can obtain the same result via an asymmetric distance computation (ADC) [1]. Let us define a K -dimensional vector, $\mathbf{T} \in \mathbb{R}^K$, whose k -th element is defined as follows:

$$T[k] = \|\mathbf{q} - \mathbf{c}_k\|_2^2. \quad (2)$$

We create \mathbf{T} only once when given \mathbf{q} . To approximate the distance between \mathbf{q} and \mathbf{x}_n , we look up \mathbf{T} by k_n as the key:

hardware that can efficiently perform the same arithmetic operation on multiple elements. The problem is that the SIMD register is small (typically 128 or 256 bits). Thus, we must modify the algorithm to fully employ it. The 4-bit PQ achieves an approximate but fast PQ via the SIMD register.

We next describe the 4-bit PQ. Recall that \mathbf{T} is a K -dimensional vector. Thus, \mathbf{T} is represented by a K -length array with floating points (32K bits total). Under the typical setting, $K = 256$, the total memory cost of \mathbf{T} is 8192 bits; hence, we cannot load it using the SIMD register. Hence, we apply a drastic approximation:

- Set $K = 16$ so that each code takes only $\log_2 16 = 4$ bits. This explains the “4-bit” aspect of PQ.
- Apply a scalar quantization for each element in \mathbf{T} , so that each element is represented by an 8-bit unsigned char.

By this, we obtain a new look-up table, $\mathbf{T}_{\text{SIMD}} \in \{1, \dots, 256\}^{16}$. Using this table, we can replace the Eq. (3) as follows:

$$\|\mathbf{q} - \mathbf{x}_n\|_2^2 \sim \|\mathbf{q} - \mathbf{c}_{k_n}\|_2^2 \sim f(\mathbf{T}_{\text{SIMD}}[k_n]), \quad (4)$$

where f is the reconstruction of an unsigned char to float, which is trivial. This table is represented by an array of 16 unsigned chars, which is 128 bits total and can be loaded onto the SIMD register. Because each code, k_n , is represented by only 4 bits, we can load 16 elements (e.g., k_1 to k_{16}) at once. Using the shuffle operation, we can run the table lookup inside the SIMD register in parallel, as shown in Fig. 1b. Then, we store the result in another SIMD register. This approach is fast because the SIMD-register lookup is much faster than a memory lookup, and we can process 16 elements at once.

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- **Mental model for computation**
- SSH + server coding
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

Mental model for computation

- Various mental models for computation
- It's important to carefully consider
 - ✓ which model to use
 - ✓ what tools to employ



My PC

Code locally

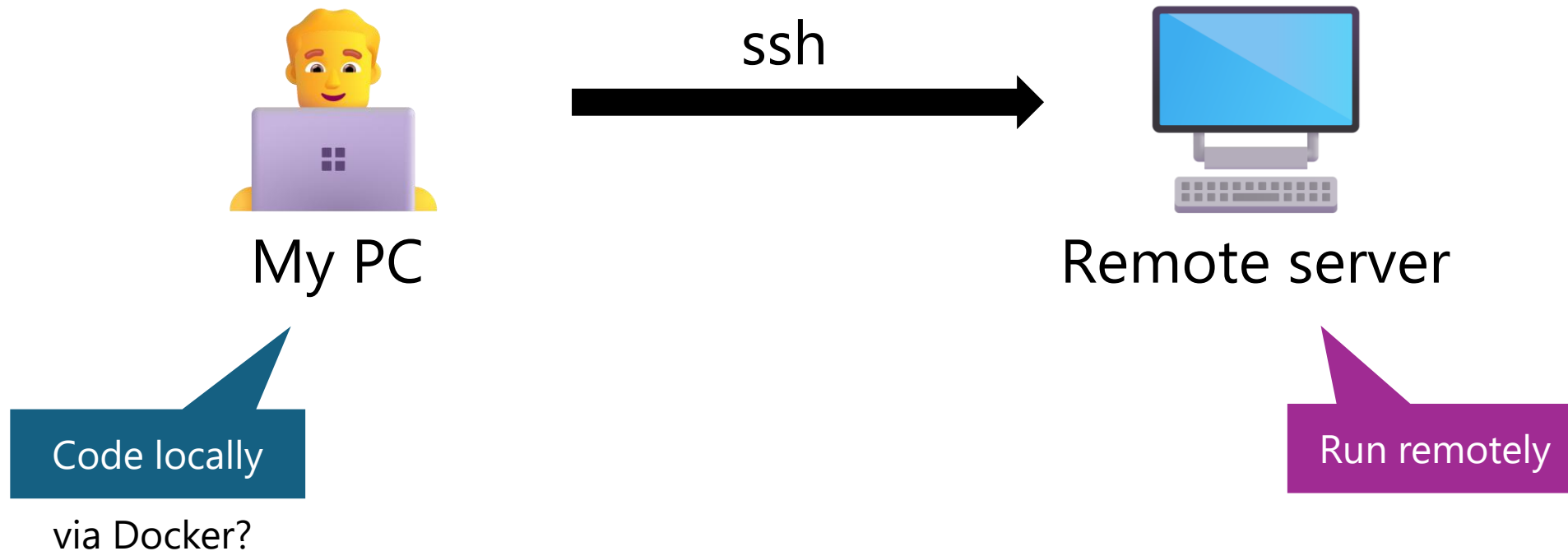
Run locally



Remote server

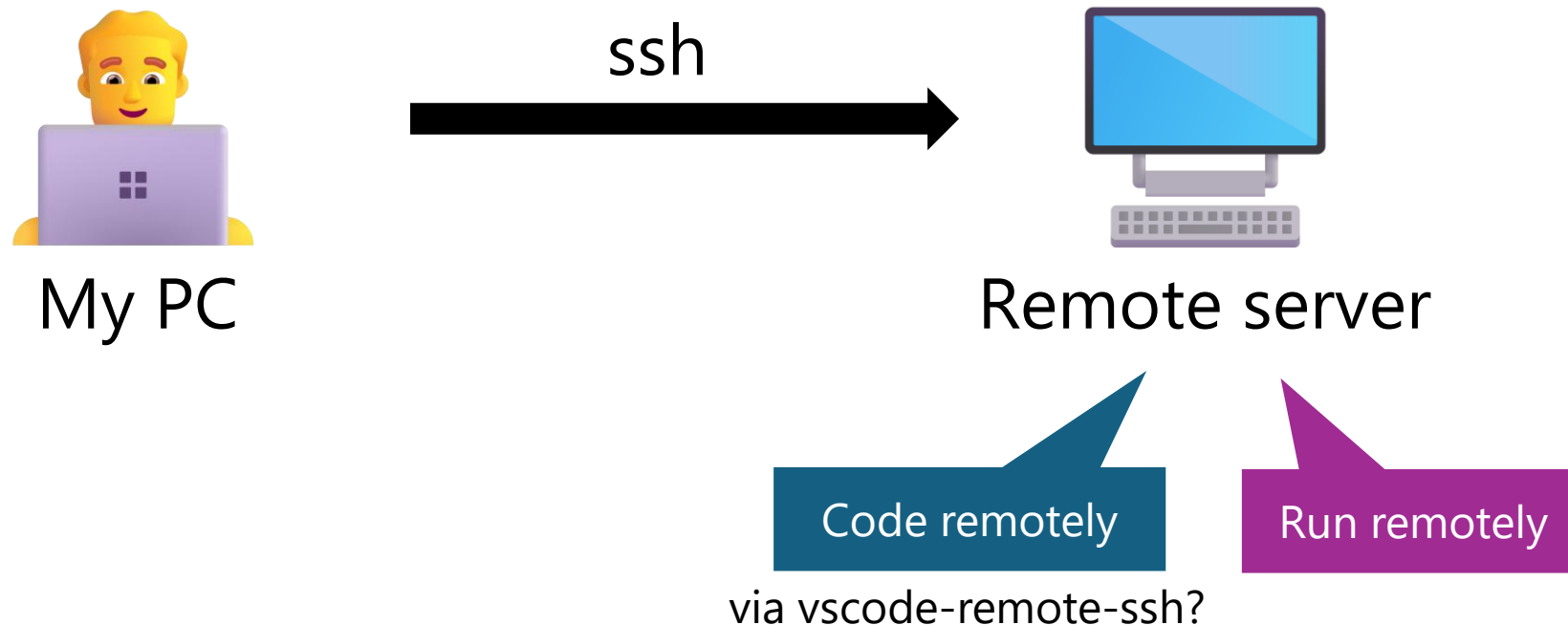
Mental model for computation

- Various mental models for computation
- It's important to carefully consider
 - ✓ which model to use
 - ✓ what tools to employ



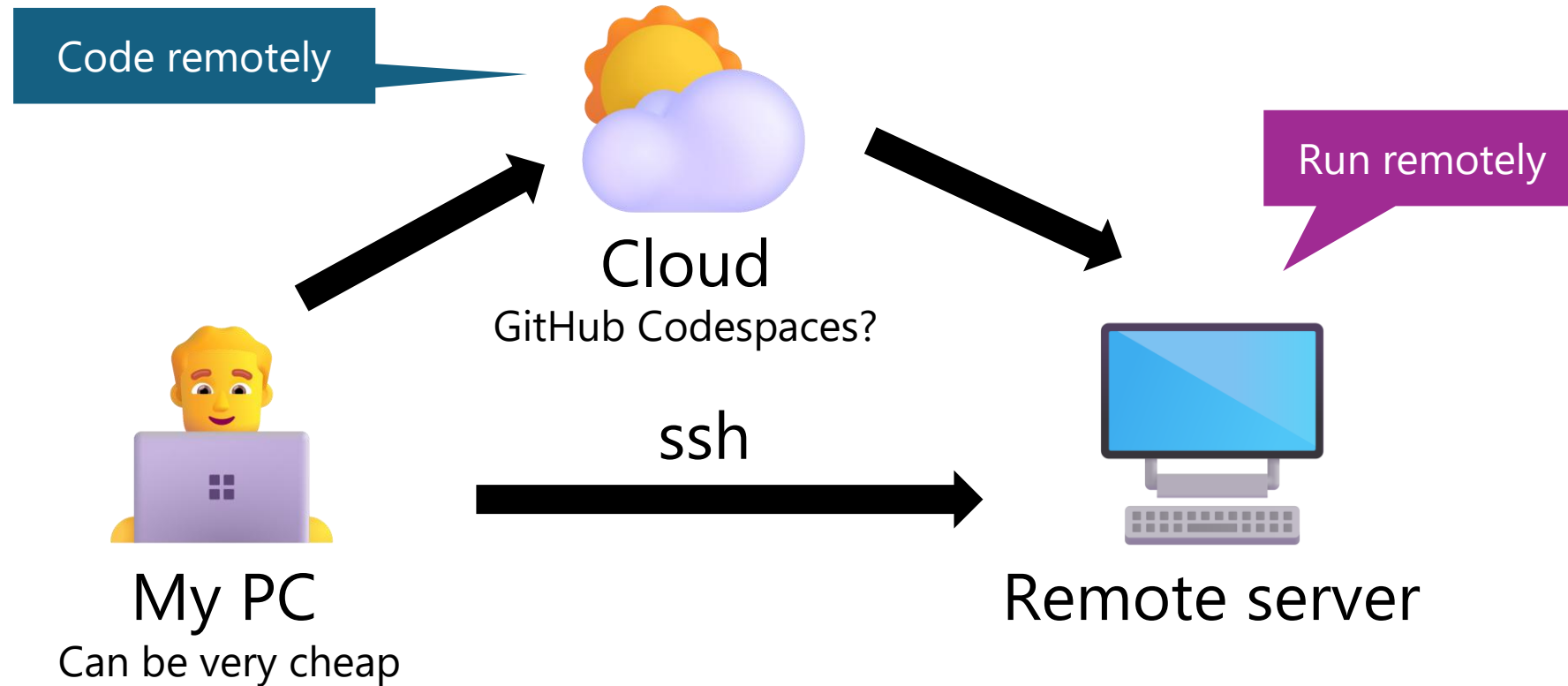
Mental model for computation

- Various mental models for computation
- It's important to carefully consider
 - ✓ which model to use
 - ✓ what tools to employ



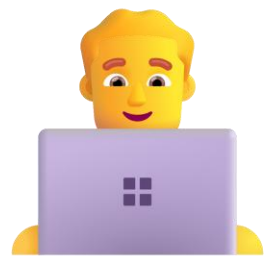
Mental model for computation

- Various mental models for computation
- It's important to carefully consider
 - ✓ which model to use
 - ✓ what tools to employ



Mental model for computation

- Various mental models for computation
- It's important to carefully consider
 - ✓ which model to use
 - ✓ what tools to employ



My PC

ssh



Remote server

submit jobs



HPC Clusters

Code remotely

via vscode-remote-ssh?

Run remotely

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- **SSH + server coding**
- UTokyo services



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

SSH + server coding

- You should be able to
 - ✓ code via GUI over ssh (vscode-remote-ssh, etc)
 - ✓ ssh, then code on terminal (🕶️ emacs, vim, nano, etc)

OVERVIEW
SETUP
GET STARTED
USER GUIDE
SOURCE CONTROL
TERMINAL
GITHUB COPILOT
LANGUAGES
NODE.JS / JAVASCRIPT
TYPESCRIPT
PYTHON
JAVA
C++
C#
DOCKER
DATA SCIENCE
AZURE
REMOTE
Overview
SSH

Remote Development using SSH

The **Visual Studio Code Remote - SSH** extension allows you to open a remote folder on any remote machine, virtual machine, or container with a running SSH server and take full advantage of VS Code's feature set. Once connected to a server, you can interact with files and folders anywhere on the remote filesystem.

No source code needs to be on your local machine to gain these benefits since the extension runs commands and other extensions directly on the remote machine.

This lets VS Code provide a **local-quality development experience** - including full IntelliSense (completions), code navigation, and debugging - **regardless of where your code is hosted**.

IN THIS ARTICLE

- Getting started
- Managing extensions
- Forwarding a port / creating SSH tunnel
- Opening a terminal on a remote host
- Debugging on the SSH host
- SSH host-specific settings
- Working with local tools
- Known limitations
- Common questions

Subscribe
Ask questions
Follow @code
Request features
Report issues
Watch videos

<https://code.visualstudio.com/docs/remote/ssh>

Agenda

- Git / GitHub (basic usage)
- Make
- Docker / Singularity
- uv/pixi
- Markdown (and structured text description)
- Notebook environment
- LaTeX / Overleaf
- Mental model for computation
- SSH + server coding
- **UTokyo services**



- You should be familiar with them
- If you don't know any of them, study on your own. There are several online resources available
- I will not cover them in detail in class.

UTokyo services

- Several services are available. Check them!
- utelecon: <https://utelecon.adm.u-tokyo.ac.jp/>

UTokyo Microsoft License

What is UTokyo Microsoft License?

Table of Contents

"UTokyo Microsoft License" provides M

The main features available are listed b

- Microsoft Office applications such
- Office for the web (web version of
- **OneDrive**: File storage
- **Microsoft Forms**: Form creation s


By signing in as a member of UTokyo, y
possible with a regular Microsoft accou
Microsoft account need to set up an ac

Initial Setup Procedure


UTokyo Account is required to use this
check "How to start using UTokyo Acco

UTokyo MATLAB Campus-Wide License

Introduction

The University of Tokyo has introduced the software "MATLAB  " provided by MathWorks, and supports its use on individual terminals, on shared terminals for education and research, and on the [supercomputer system !\[\]\(31846d3a5bf9ab1fd9d1cab2b92a526a_img.jpg\)](#) (in Japanese) provided by the Information Technology Center.

About MATLAB

MATLAB  is a programming language and a numerical computation / computer algebra software developed for scientific and technical computing. MATLAB can be used for various purposes, including mathematical processing of algebra, geometry and analysis, machine learning, statistical analysis, data visualization, control simulation, and its implementation on hardware. Since the software can be operated using GUI without writing a code, it can be easily introduced not only for research use in specialized fields such as engineering and science, but also for basic education for beginners in programming, and data analysis in the humanities and social sciences.

The University of Tokyo has a comprehensive license agreement that allows **all students and all faculty and staff members employed by the university** to use the following functions from their individual devices without additional costs. The number of functions provided may increase or decrease depending on the university's situation and the terms of the contract.

- Installing and using MATLAB
- Unlimited use of MATLAB Online

UTokyo Microsoft Azure Dev Tools for Teaching

Overview

"Azure Dev Tools for Teaching" is software for developers offered by Microsoft for educational institutions. It is available at the University of Tokyo based on the contract of [UTokyo Microsoft License](#). Users are required to comply with the terms by Microsoft.

Scope of use

- The software can be used only for **development purposes in education and research**.
 - **It cannot be used for administrative or business purposes**. Even for educational/research activities, use for non-development purposes, such as for general PC use, is not permitted.
- Even for educational/research activities, **it is not allowed if the use is considered as commercial purpose**.
- **Students of the University of Tokyo** are eligible to use this service. **Faculty and staff members of the University of Tokyo** who are directly engaged in educational and research activities are also eligible to use this service.
 - No matter which faculty or graduate school you belong to.

Initial Procedure for Use

Table of Contents

- **Overview**
- Scope of use
- **Initial Procedure for Use**
- Others

Table of Contents

- I
- A
- C
- F
- I
- I
- M
- F
- M
- H

- Installing the software version
- Using the online version
 - Access to MATLAB Online
 - Access to MATLAB Drive
- Learning how to operate

Schedule

Date (2026)	Contents	Presented by
Week 1, Apr 15	Introduction. Review of fundamental concepts	Yusuke, Koya, Yuki, Jun
Week 2, Apr 22	Equations and pseudo-codes	Yusuke Matsui
Week 3, May 13	Slides	Koya Narumi
Week 4, May 20	Research community	Jun Kato
Week 5, May 27	Figures	Koya Narumi
Week 6, June 1	3DCG illustrations	Yuki Koyama
Week 7, June 10	Invited Talk 1: Research tips in the private sector: From topic selection to social implementation (temp.)	Dr. Antonio Tejero de Pablos (CyberAgent)
Week 8, June 17	Tables and plots	Yusuke Matsui
Week 9, June 24	Invited Talk 2: Portable and Reproducible Research Environments in the Age of AI Agents	Dr. Mai Nishimura (OSX)
Week 10, July 1	DevOps for research	Jun Kato
Week 11, July 8	Videos	Koya Narumi
Week 12, July 15	Final presentations	YOU

