Train the trainer workshop

August/September 2024 CodeRefinery train the trainer workshop

Do you teach the use of computers and computational tools? Are you curious about scaling your training and learn about tested and proven practices from CodeRefinery workshops and other trainers? Join us for the CodeRefinery train the trainer workshop: four self-contained sessions on tools and techniques for computational training offer a great chance to enhance your teaching skills and learn about new tools and practices. What you will learn is also used a lot outside CodeRefinery, whenever good beginner friendly training is needed.

Learning objectives:

- Learn about tools and techniques and practice to create engaging online teaching experiences (screenshare, audio, etc.).
- Become mentally ready to be an instructor in a collaborative interactive online workshop (not only large workshops but in general).
- Learn how to design and develop lesson material collaboratively.

Target audience:

- Everyone teaching online workshops about computational topics or interested in teaching.
- Previous and future instructors and helpers of CodeRefinery workshops.

Prerequisites: An interest in teaching.

Workshop structure: We aim for a mix of modular presentations and work in small groups. The latter will be done in breakoutrooms - be prepared to switch on your camera and have a fruitful exchange with fellow participants!

Organizers and instructors:

The workshop is organized by partner organizations of the CodeRefinery project.

Facilitators and instructors:

- Radovan Bast
- Richard Darst
- Bjørn Lindi
- Dhanya Pushpadas
- Janne Rantaharju
- Stephan Smuts
- Stefanie Tollaksen
- Samantha Wittke

Content and timing:

The workshop consists of four sessions every Tuesday between August 13th and September 3rd, 2024, as:

- Session 1: About teaching & cool things we all would like to share (Aug 27)
- Session 2: Workshop streaming practices and post-workshop tasks (Sep 3)

You can join all sessions, or the just the ones that interest you. More details on each session will be announced later.

The workshop is free of charge for everyone, please register below to get the Zoom link and other useful information for the workshop.
The workshop is over and registration is closed now.

If you have any questions, please write to support@coderefinery.org.

- Session 1: About lesson design, delivery and iterative improvement (Aug 13)
- Session 2: Tools and techniques adopted in CodeRefinery workshops (Aug 20)

- Session 3: About teaching & cool things we all would like to share (Aug 27)
- Session 4: Workshop streaming practices and post-workshop tasks (Sep 3)

You can join all sessions, or the just the ones that interest you. More details on each session will be shared later.

Content and timing.

The workshop is **free of charge for everyone**, please register below to get the Zoom link and other useful information for the workshop.
The workshop consists of four sessions every Tuesday between August 13th and September 3rd 2024, and

The workshop is over and registration is closed now.

If you have any questions, please write to support@coderefinery.org.

- Session 1: About lesson design, deployment and iterative improvement (Aug 13)
- Session 2: Tools and techniques adopted in CodeRefinery workshops (Aug 20)

Lesson design and development

Objectives

- We share our design processes for teaching material and presentations.
- Learn how to design lessons “backwards”, starting from learning objectives and learner personas.
- Learn good practices for improving existing material based on feedback.

Instructor note

- Discussion: 20 min
- Exercises: 35 min

Exercise: How do you design your teaching material?

✍ We collect notes using a shared document (5 min)

- When you start preparing a new lesson or training material, where do you start?
- What tricks help you with “writer’s block” or the empty page problem?
- Maybe you haven’t designed training material yet. But how do you start when creating a new presentation?
- If your design process has changed over time, please describe what you used to do and what you do now instead.
- What do you know now about preparing lessons/training/presentations that you wish you knew earlier?

Creating new teaching material

Typical problems

- Someone creates a lesson, but they think about what is interesting to them, not what is important for the learners.
- “I want to show a number of things which I think are cool about tool X - how do I press these into 90 minutes?”
- Write down material you want to cover and then sprinkle in some exercises.
- Thinking about how I work, not how the learners work.
- Define learner personas.
- Trying to bring learners to their level/setup, not trying to meet the learners where they are.
- It may be an advantage to share an imperfect lesson with others early to collect feedback and suggestions before the lesson “solidifies” too much. Draft it and collect feedback. The result will probably be better than working in isolation towards a “perfect” lesson.
- Not really knowing the learning objectives or the learner personas.

Better approach

Good questions to ask and discuss with a group of colleagues **from diverse backgrounds**:

- What is the expected educational level of my audience?
- Have they been already exposed to the technologies I am planning to teach?
- What tools do they already use?
- What are the main issues they are currently experiencing?
- What do they need to remember/understand/apply/analyze/evaluate/create (Bloom’s taxonomy)?

The process of designing a lesson “backwards”

As described in “A lesson design process” in the book Teaching Tech Together:

- What is the expected educational level of my audience?
- Have they been already exposed to the technologies I am planning to teach?
- What tools do they already use?
- What are the main issues they are currently experiencing?
- What do they need to remember/understand/apply/analyze/evaluate/create (Bloom's taxonomy)?
- Define learner personas.
- It may be an advantage to share an imperfect lesson with others early to collect feedback and suggestions before the lesson "solidifies" too much. Draft it and collect feedback. The result will probably be better than working in isolation towards a "perfect" lesson.

Good questions to ask and discuss with a group of colleagues **from diverse backgrounds**:

1. Understand your learners.
2. Brainstorm rough ideas.
3. Create an summative assessment to know your overall goal.

   [Think of the things your learners will be able to do at the end of the lesson]

4. Create formative assessments to go from the starting point to this.

   [Think of some engaging and active exercises]

5. Order the formative assessments (exercises) into a reasonable order.
6. Write just enough material to get from one assessment (exercise) to another.
7. Describe the course so the learners know if it is relevant to them.

Improving existing lessons

💬 All CodeRefinery lessons are on GitHub

- Overview: https://coderefinery.org/lessons/
- All are shared under CC-BY license and we encourage reuse and modification.
- Sources are all on GitHub: https://github.com/coderefinery
- Web pages are generated from Markdown using Sphinx (more about that in the episode Lessons with version control).
- We track ideas and problems in GitHub issues.

Collect feedback during the workshop:

- Collect feedback from learners and instructors (Example from a past workshop).
- Convert feedback about lessons and suggestions for improvements into issues so that these don't get lost and stay close to the lesson material.

Collect feedback before you start a big rewrite:

- First open an issue and describe your idea and collect feedback before you start with an extensive rewrite.
- For things still under construction, open a draft pull/merge request to collect feedback and to signal to others what you are working on.

As an example to demonstrate the process of designing and improving lessons, we will have a look at one of our own lessons: Introduction to version control with Git.
Small picture changes vs. big picture changes:

- Lesson changes should be accompanied with instructor guide changes (it's like a documentation for the lesson material).
- Instructor guide is essential for new instructors.
- Before making larger changes, talk with somebody and discuss these changes.
- Some time in 2014-2015 attended Carpentries instructor training.
- 2016: CodeRefinery started.
- 2017: Started a new repository based on the Carpentries lesson template (at the time

- Someone creates a lesson, but they think about what is interesting to them, not what is important for the learners.
- "I want to show a number of things which I think are cool about tool X - how do I press these into 90 minutes?"
- Write down material you want to cover and then sprinkle in some exercises.
- Thinking about how I work, not how the learners work.
- Trying to bring learners to their level/setup, not trying to meet the learners where they are.
- Not really knowing the learning objectives or the learner personas.

The process of designing a lesson "backwards"

Better approach

As described in "A lesson design process" in the book Teaching Tech Together:

Use case: our lessons

- Format: slides and live coding.
- Exercises were separate, during afternoon sessions.

- extensive rewrite.
- For things still under construction, open a draft pull/merge request to collect feedback and to signal to others what you are working on.

Use case: our lessons

As an example to demonstrate the process of designing and improving lessons, we will have a look at one of our own lessons: Introduction to version control with Git.
Small picture changes vs. big picture changes:

- Initial 2014-2016 version
  - https://github.com/sesam/toolbox-talks and https://toolbox.readthedocs.io/
  - Amazingly they are still findable!
  - Format: slides and live coding.
  - Exercises were separate, during afternoon sessions.
- Some time in 2014-2015 attended Carpentries instructor training.
- 2016: CodeRefinery started.
- 2017: Started a new repository based on the Carpentries lesson template (at the time using Jekyll).
  - Exercises become part of the lesson.
  - We start in the **command line** and only later move to GitHub.
- 2019: A lot more thought about learning objectives and personas.
  - Also license change to CC-BY.
- 2022: Convert lesson from Jekyll to Sphinx.
  - Using the tools that we teach/advocate.
  - We can have tabs and better code highlighting/emphasis.
  - Easier local preview (Python environment instead of Ruby environment which we were not used to in our daily work).
- 2024: Big redesign. We move the lesson closer to where learners are.
  - Start from GitHub instead of on the command line.
  - Start from an existing repository instead of with an empty one.
  - Offer several tracks to participate in the lesson (GitHub, VS Code, and command line) and learners can choose which one they want to follow.
  - Blog post: We have completely changed our Git lessons. Hopefully to the better.
- Next steps?
  - Making the lesson citable following our blog post.
  - Improvements based on what we learn from this workshop.

Lesson changes should be accompanied with instructor guide changes (it's like a documentation for the lesson material).
Instructor guide is essential for new instructors.
Before making larger changes, talk with somebody and discuss these changes.

The overarching trend was to make the lesson simpler and more accessible - to meet the learners where they are instead of pulling them to the tool choices of the instructors. Looking back, we learned a lot and the learning process is not over yet.

Exercise: Discussion about learning objectives and exercise design

✍️ We work in groups but use the shared document as result (20 min)

1. As a group **pick a lesson topic**. It can be one of the topics listed here but you can also choose something else that your group is interested in, or a topic that you have taught before or would like to teach. Some suggestions:
   - Git: Creating a repository and porting your project to Git and GitHub
   - Git: Basic commands
   - How to release and publish your code
   - Git: Branching and merging
   - How to document and track code dependencies
   - Git: Recovering from typical mistakes
   - Recording environments in containers
   - Code documentation
   - Profiling memory and CPU usage
   - Jupyter Notebooks
   - Strategies for parallelization
   - Collaboration using Git and GitHub/GitLab
   - Conda environments
   - Using GitHub without the command line
   - Data processing using workflow managers
   - Project organization
   - Regular expressions
   - Automated testing
   - Making papers in LaTeX
   - Data transfer
   - Making figures in your favorite programming language
   - Data management and versioning
   - Linux shell basics
   - Code quality and good practices
   - Something non-technical, such as painting a room
   - Modular code development
   - Introduction to high-performance computing

- Git: Basic commands
- How to release and publish your code
- Git: Branching and merging
- How to document and track code dependencies
- Git: Recovering from typical mistakes
- Recording environments in containers
- Code documentation
- Profiling memory and CPU usage
- Jupyter Notebooks
- Strategies for parallelization
- Collaboration using Git and GitHub/GitLab
- Conda environments
- Using GitHub without the command line
- Data processing using workflow managers
- Project organization
- Regular expressions
- Automated testing
- Making papers in LaTeX
- Data transfer
- Making figures in your favorite programming language
- Data management and versioning
- Linux shell basics
- Code quality and good practices
- Something non-technical, such as painting a room
- Modular code development
- Introduction to high-performance computing
- A lesson you always wanted to teach
- ...

2. Try to define 2-3 learning objectives for the lesson and write them down. You can think of these as "three simple enough messages that someone will remember the next day" - **they need to be pretty simple**.
3. Can you come up with one or two engaging exercises that could be used to demonstrate one of those objectives? **They should be simple** enough people can actually do them. Creating simple exercises is not easy. Some standard exercise types:
   - Multiple choice (easy to get feedback via a classroom tool - try to design each wrong answer so that it identifies a specific misconception).
   - Code yourself (traditional programming)
   - Code yourself + multiple choice to see what the answer is (allows you to get feedback)
   - Inverted coding (given code, have to debug)
   - Parsons problems (working solution but lines in random order, learner must only put in proper order)
   - Fill in the blank
   - Discussions, self directed learning exercises

Great resources

- Teaching Tech Together
- Our summary of Teaching Tech Together
- Ten quick tips for creating an effective lesson
- Carpentries Curriculum Development Handbook
- Our manual on lesson design

Lessons with version control

Objectives

- Understand why version control is useful even for teaching material
- Understand how version control managed lessons can be modified.
  - Change history
  - Others can submit contributions
  - Others can make derived versions and sync up later
  - Same workflow as everything else
  - Write it like documentation: probably more reading after than watching it as a presentation.
- Understand how the CodeRefinery lesson template is used to create new lessons

Instructor note:

- Discussion: 25 min
- Exercises or demos: 20 min
- Disadvantages
  - "What you see is what you get" editing is hard
  - Requires knowing version control

Why version control?

Accepting the smallest contribution

Question: if someone wants to make a tiny fix to your material, can they?

- The benefits of version control also extend to lessons:
- Assume that learners probably know what version control is and why it is important.

- Change history
- Others can submit contributions
- Others can make derived versions and sync up later
- Same workflow as everything else
- Write it like documentation: probably more reading after than watching it as a presentation.

- Disadvantages
  - “What you see is what you get” editing is hard
  - Requires knowing version control

- If you are in CodeRefinery TTT, you probably know what version control is and why it is important.
- The benefits of version control also extend to lessons.

- Understand how the CodeRefinery lesson template is used to create new lessons

Instructor note

- Discussion: 25 min
- Exercises or demos: 20 min

Why version control?

💬 Accepting the smallest contribution

Question: if someone wants to make a tiny fix to your material, can they?

Tour of lesson templates options

There are different ways to make lessons with git. Some dedicated to teaching:

- CodeRefinery
  - Example: This lesson itself
  - Based on the Sphinx documentation generator
  - sphinx-lesson is *very* minimal extra functionality
- Carpentries
  - Example: https://carpentries.github.io/lesson-example/
  - Based on R and Rmarkdown

Our philosophy is that anything works: it doesn’t have to be just designed for lessons

- Jupyter Book
  - Example: https://jupyterbook.org/
  - Note: is based on sphinx, many extensions here are used in CR lessons
- Various ways to make slides out of Markdown
- Cicero: GitHub-hosted Markdown to slides easily
  - Demo: Asking for Help with Supercomputers The source
- Whatever your existing documentation is.

We like the CodeRefinery format, but think that you should use whatever fits your needs the most.

Sphinx

- We build all our lesson material with Sphinx
- Generate HTML/PDF/LaTeX from RST and Markdown.
- Many Python projects use Sphinx for documentation but **Sphinx is not limited to Python**.
- Read the docs hosts public Sphinx documentation for free!
- Also hostable anywhere else, like Github pages, like our lesson material
- For Python selling point for Sphinx is that also API documentation is possible.

Sphinx is a doc generator, not HTML generator. It can:

- Markdown, Jupyter, and ReST (and more…) inputs. Executable inputs.
  - jupyter-book is Sphinx, so anything it can do we can do. This was one of the inspirations for using Sphinx
- Good support for inline code. Much more than static code display, if you want to look at extensions.
- Generate different output formats (html, single-page html, pdf, epub, etc.)
- Strong cross-referencing within and between projects

CodeRefinery lesson template

It’s “just a normal Sphinx project” – with extensions:

- Sphinx-lesson extension
  - adds various directives (boxes) tuned to lesson purposes
  - provides a sample organization and template repo you can use so that lessons look consistent
- Sphinx gives us other nice features for free
  - Tabs: they are very important for us and allow us to customize in-place instead of copying everything and fragmenting the lesson
  - Emphasize lines: they make it easier to spot what has changed in longer code snippets
  - Various input formats

CodeRefinery lesson template

It's "just a normal Sphinx project" with extensions:

- Sphinx lesson extension
  - adds its various directives (boxes) tuned to lesson purposes
  - provides a sample organization and template repo you can use so that lessons look consistent

- For Code, selling point for Sphinx is that also API documentation is possible.

Sphinx is a doc generator, not HTML generator. It can:

- Markdown, Jupyter, and ReST (and more...) inputs. Executable inputs.
  - jupyter-book is Sphinx, so anything it can do we can do. This was one of the inspirations for using Sphinx
- Good support for inline code. Much more than static code display, if you want to look at extensions.
- Generate different output formats (html, single-page html, pdf, epub, etc)
- Strong cross-referencing within and between projects

- Sphinx gives us other nice features for free
  - Tabs: they are very important for us and allow us to customize in-place instead of copying everything and fragmenting lesson
  - Emphasize lines: they make it easier to spot what has changed in longer code snippets
  - Various input formats
    - Markdown (via the MyST-parser), ReStructured text, Jupyter Notebooks.
  - Many other features designed for presenting and interacting with code
- It's fine if you use some other static site generator or git-based lesson method.

👀 Instructors go through the building and contributing process

Depending on the course, instructors will demo what is roughly exercise 4 below. Or a course might go straight to exercises.

- Instructors decide what change they would want to make
- Instructors clone the repository
- **Instructors make the change**
- **Instructors set up the build environment**
- **Instructors build and preview**
- Instructors command and send upstream

Exercises

Some exercises have prerequisites (Git or Github accounts). Most instances of this course will have you do **1 and 2** below.

✍️ Lesson-VCS-1: Present and discuss your own lesson formats

We don't want to push everyone towards one format, but as long as you use Git, it's easy to share and reuse.

- Discuss what formats you do use
- Within your team, show examples of the lessons formats you use now. Discuss what is good and to-be-improved about them.
- Look at how they source is managed and how easy it might be to edit.

✍️ Lesson-VCS-2: Tour a CodeRefinery or Carpentries lesson on Github

- Look at either a CodeRefinery or Carpentries lesson
  - CodeRefinery Git-Intro: Lesson, Github repo
  - Carpentries Linux shell: Lesson, Github repo
- Can you find
  - Where is the content of the lessons?
  - What recent change proposals (pull requests) have been made?
  - What are the open issues?
  - How you would contribute something?
  - How would you cite this yourself?

✍️ Lesson-VCS-3: Modify a CodeRefinery example lesson on Github

In this, you will make a change to a lesson on Github, using only the Github interface. (Github account required, and we assume some knowledge of Github. Ask for help in your team if it is new to you!)

- Navigate to the example lesson we have set up: repo, web
- Go to some page and follow the link to go to the respective page on Github. (Alternatively, you can find the page from the Github repo directly).
- Follow the Github flow to make a change, and open a Pull Request with the change proposal:
  - Click on the pencil icon
  - Make a change
  - Follow the flow to make a commit and change. You'll fork the repository to make your own copy, add a message, and open a pull request.

✍ Lesson-VCS-3: Modify a CodeRefinery example lesson on Github

In this, you will make a change to a lesson on Github, using only the Github interface. (Github account required, and we assume some knowledge of Github. Ask for help in your team if it is new to you.)

- Look at either a CodeRefinery or Carpentries lesson
  - CodeRefinery Git-Intro: Lesson, Github repo
  - Carpentries Linux shell: Lesson, Github repo
- Navigate to the example lesson we have set up: repo, web
- Go to some page and follow the lesson to go to the respective page on Github. (Alternatively, you can find the page from the Github repo directly).
- Follow the Github flow to make a change, and open a Pull Request with the change proposal:
  - Click on the pencil icon
  - Make a change
  - Follow the flow to make a commit and change. You'll fork the repository to make your own copy, add a message, and open a pull request.

We will look at these together and accept some changes.

✍ Lesson-VCS-4: Clone and build a CodeRefinery lesson locally

In this exercise, you will use Git to clone one a CodeRefinery lesson and try to preview it locally. It assumes installation and knowledge of Git.

- Use this sample repository: git-intro (or whatever else you would like)
- Clone the repository to your own computer
- Create a virtual environment using the `requirements.txt` contained within the repository.
- Build the lesson.
  - Most people will probably run: `sphinx-build content/ _build/`
  - If you have `make` installed you can `make html`
  - Look in the `_build` directory for the built HTML files.

Overall, this is pretty easy and straightforward, if you have a Python environment set up. The CodeRefinery documentation lesson teaches this for every operating system.

This same tool can be used to build documentation for other software projects and is pretty standard.

✍ Lesson-VCS-5: (advanced) Create your own lesson using the CodeRefinery format

In this lesson, you'll copy the CodeRefinery template and do basic modifications for your own lesson.

- Clone the lesson template: https://github.com/coderefinery/sphinx-lesson-template
- Attempt to build it as it is (see the previous exercise)
- How can you do tabs?
- How can you highlight lines in code boxes?
- How can you change colors, logo and fonts?
- What directives are available?

How we collect feedback and measure impact

❗ Objectives

- Discuss how one can collect feedback from learners ("what can we improve?").
- Discuss how we convert feedback into actionable items.
- Discuss how we measure the impact of teaching ("did we achieve our goals?").
- Discuss the "why".
- Get to know the reasons and sources of inspiration behind major lesson and workshop updates.

Summary

Version control takes teaching materials to the next level: shareable and easy to contribute.

There are different formats that use version control, but we like Sphinx with a sphinx-lesson extension.

Instructor note

- How can you highlight lines in code boxes?
- How can you change color, logo and fonts?
- What directives are available?

Objectives

- Discuss how one can collect feedback from learners (“what can we improve?”).
- Discuss how we convert feedback into actionable items.
- Discuss how we measure the impact of teaching (“did we achieve our goals?”).
- Discuss the “why”.
- Get to know the reasons and sources of inspiration behind major lesson and workshop updates.

Instructor note

- Discussion: 20 min
- Exercises: 10 min

Asking questions before the workshop

- Motivation: Know your audience.
- Until 2021 we had a pre-workshop survey:
  - Data, questions, and notebook: https://github.com/coderefinery/pre-workshop-survey/
  - Zenodo/DOI: https://doi.org/10.5281/zenodo.2671578
- After 2021 we incorporated some of the questions into the registration form.
  - Easier registration experience for participants.
  - After 5 years of running workshops, we had a good idea of what to expect.

Collecting feedback as we teach

- Each day we ask for feedback in the collaborative notes.
  - One good thing about today.
  - One thing to improve for next time.
  - Any other comments?
- During the workshop we sometimes check in and ask about the pace, example:

```
How is the speed so far? (add an "o")
- Too fast:   oooooo
- Too slow:   ooo
- Just right: oooooooooooooooooo
```

- We publish all questions, answers, and feedback. Example: https://coderefinery.github.io/2024-03-12-workshop/questions/
- How we follow up:
  - Some problems we can fix already before the next workshop day.
  - We convert feedback/problems into GitHub issues and track these close to the lesson material.
  - When reporting to funders.
  - When planning future workshops and bigger picture changes.

Trying to measure impact with longer-term surveys

- 2021 version:
  - Data, questions, notebook, and figures: https://github.com/coderefinery/post-workshop-survey
  - Zenodo/DOI: https://doi.org/10.5281/zenodo.2671576
- Motivation: Understand the long-term impact of our workshops. Have something to show to funders.
- 2024 post-workshop survey:
  - Blog post: https://coderefinery.org/blog/2024/09/10/post-workshop-survey/
  - Questions, notebook, and figures: https://github.com/coderefinery/2024-post-workshop-survey
  - Zenodo/DOI: https://doi.org/10.5281/zenodo.13292363

Lessons learned

- Think about how to measure impact/success from the beginning.
- Make the feedback and survey results public.
- Make the results persistent and citable.
- Have a mechanism to follow up on feedback.

How we collect feedback and measure impact

Summary

Version control takes teaching materials to the next level: shareable and easy to contribute.

There are different formats that use version control, but we like Sphinx with a sphinx-lesson extension.

- 2021 version:
  - Data, questions, notebook, and figures: https://github.com/coderefinery/post-workshop-survey
  - Zenodo/DOI: https://doi.org/10.5281/zenodo.2671576
- How we use the results:
  - When reporting to funders.
  - When planning future workshops and bigger picture changes.
- 2024 post-workshop survey:
  - Blog post: https://coderefinery.org/blog/2024/08/10/post-workshop-survey/
  - Questions, notebook, and figures: https://github.com/coderefinery/2024-post-workshop-survey
  - Zenodo/DOI: https://doi.org/10.5281/zenodo.13292363

- Some problems we can fix already before the next workshop day.
- We convert feedback/problems into GitHub issues and track these close to the lesson material.

Trying to measure impact with longer-term surveys

- Motivation: Understand the long-term impact of our workshops. Have something to show to funders.

Lessons learned

- Think about how to measure impact/success from the beginning.
- Make the feedback and survey results public.
- Make the results persistent and citable.
- Have a mechanism to follow-up on feedback.
- Anonymization is more than just removing or dissociating names.
- Take time designing your survey and collect feedback on the survey itself.

Take time designing your survey

We are not experts in survey design but we reached out to RISE Research Institutes of Sweden to get feedback on our survey questions. They provided us with a lot of valuable feedback and suggestions for improvements. Below are few examples.

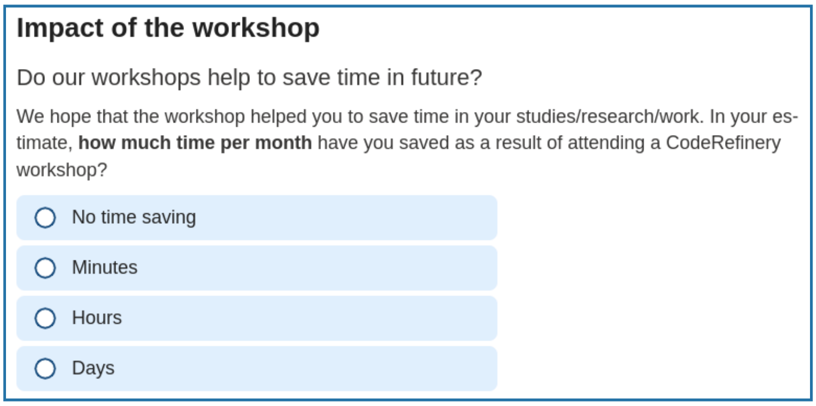
Example 1

First version of our survey question about impact:

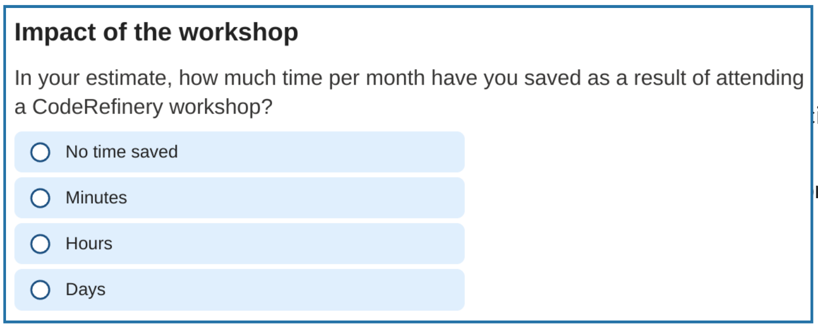
- "Do our workshops help to save time in future?" Please quantify in hours saved: …

Feedback:

- Difficult to answer.
- Since when? Until all eternity?



Earlier version of the survey question about impact.



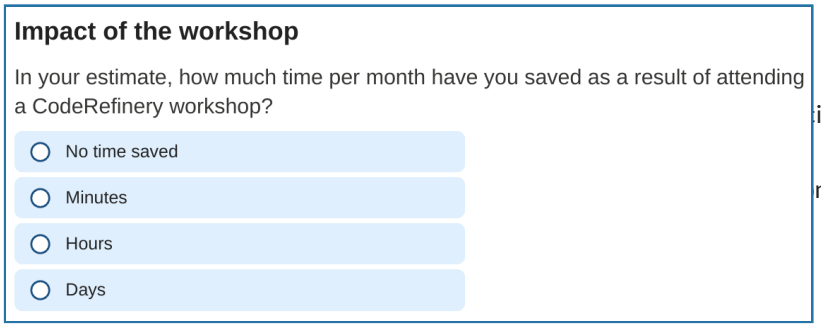
ified, unnecessary,

n.

Later version of the survey question about impact.

- The wording "have you saved" now matches "No time saved".

Example 2



ified, unnecessary,

n.

Later version of the survey question about impact.

- The wording “have you saved” now matches “No time saved”.

Example 2

- Earlier version:

```
Would you judge your code to be better reusable/reproducible/modular/documented
as a result of attending the workshop?

- More reusable
- More reproducible
- More modular
- Better documented
- None of the above
```

- Feedback: The question is not neutrally formulated and risks leading to over-reporting of yes answers. Consider balancing so that the questions are formulated more neutrally.
- Reformulated to:

```
After attending the workshop, would you judge your code to be
more reusable or not more reusable?

- My code is more reusable
- My code is not more reusable
- Not sure
```

Example 3

- Early version:

```
What else has changed in how you write code since attending the workshop?

[free-form text field]
```

- Feedback: Leading and assumes that something has changed.
- Reformulated to:

```
Has it become easier for you to collaborate on software development with your
colleagues and collaborators?

- Yes
- Not sure
- No
```

Example 4

- Feedback:
- Earlier version:
  - Leading question.
  - Avoid “Yes” and “No” response options because respondents tend to answer “Yes” if that option is available, leading to a risk of measurement error (acquiescence bias).
  - Do not place “Not sure” in the middle of the scale because it captures participants who actually don’t know and should therefore be placed at the bottom instead of in the middle of the scale.

```
Has it become easier for you to collaborate on software development with your
colleagues and collaborators?

- Yes
- Not sure
- No
```

Example 4

- Feedback:
- Earlier version:
  - Leading question.
  - Avoid “Yes” and “No” response options because respondents tend to answer “Yes” if that option is available, leading to a risk of measurement error (acquiescence bias).
  - Do not place “Not sure” in the middle of the scale because it captures participants who actually don’t know and should therefore be placed at the bottom instead of in the middle of the scale.
- Reformulated to:

```
After attending the workshop, has it become easier or not for you to
collaborate on software development with your colleagues and collaborators?

- Collaboration is easier
- Collaboration is not easier
- Not sure
```

Exercise: Group discussion (10 min)

✍️ Group discussion using the collaborative notes

- What tricks/techniques have you tried in your teaching or seen in someone else’s teaching that you think have been particularly effective in collecting feedback from learners?
- Can you give tips or share your experiences about how to convert feedback into actionable items?
- How do you measure the impact of your teaching? Any tips or experiences about what you have tried or seen other courses do?
- Anybody knows of good resources on survey design? Please link them in the collaborative notes.

About the CodeRefinery project and CodeRefinery workshops in general

❗ Objectives

- Discuss what CodeRefinery is and how we got here
- Understand about the challenges to define our target audience

CodeRefinery is a Nordic e-Infrastructure Collaboration (NeIC) project that has started in October 2016 and is currently funded until February 2025. We are working on the continuation plans.

- We have started by porting own lessons to the Carpentries teaching style and format, and collaboratively and iteratively grew and improved the material to its present form.

The funding from 2022-2025 is designed to keep this project active beyond 2025 by forming a support network and building a community of instructors and contributors.

Goals

💬 History

- The CodeRefinery project idea grew out of two SeSE courses given at KTH Stockholm in 2014 and 2015. The project proposal itself was submitted to NeIC in 2015, accepted in 2015, and started in 2016.
- Provide a [code repository hosting service](#) that is open and free for all researchers based in universities and research institutes from Nordic countries.
- Provide **training opportunities** in the Nordics using (Carpentries and) CodeRefinery training materials.
- Our material addresses all academic disciplines and is not tied to any programming language-dependent topics possible.

October 2016 and is currently funded until February 2025. We are working on the continuation plans.

Goals

- Develop and maintain **training material on good enough software development practices** for researchers that write code/scripts/notebooks.
- The material addresses all academic disciplines and tries to be as programming language-independent as possible.
- Provide **training opportunities** in the Nordics using (Carpentries and) CodeRefinery training materials.
- Evolve the project towards a **community-driven project** with a network of instructors and contributors.

Community

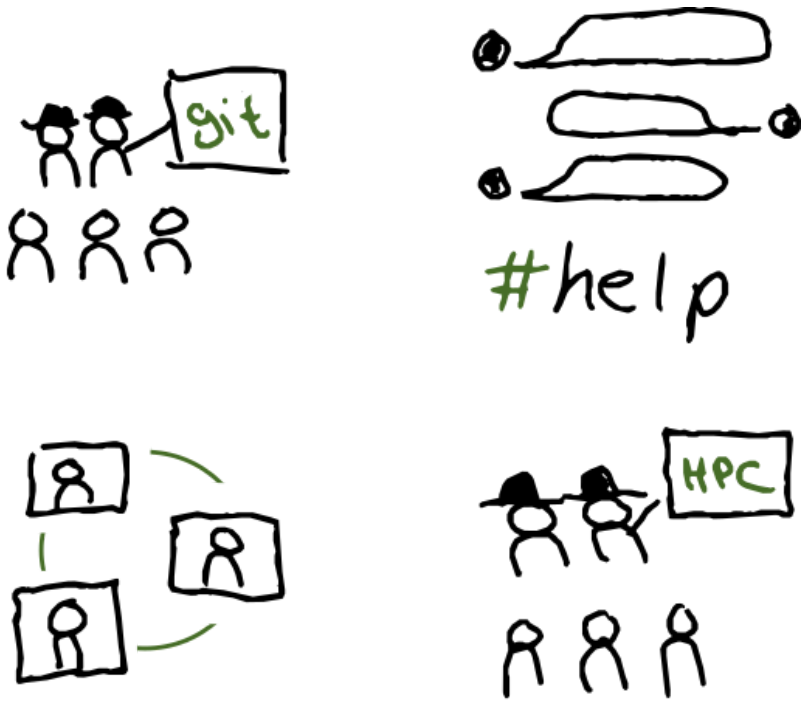


Image showing the key areas of the CodeRefinery community: Workshops, chat inlcuding help channel, online meetings and co-working, other collaborative training, eg on High Performance Computing topics.

CodeRefinery is not just workshops, we are community and want you to be part of it!

There are many different levels of involvement, from the occasional commenter in chat, CodeRefinery ambassador (people who like the project and workshops and help us spreading the word) or lesson issue creators and lesson contributors to local host, co-instructor or co-organizer.

Best **first step** in any case is to join the CodeRefinery Zulip chat or let us know about your interest at support@coderefinery.org.

Target audience

By showing learners how to solve specific problems with specific tools and providing hands-on practice, learners develops confidence for future learning.

One common question we get is how do we relate to the Carpentries. This section describes how we see it:

Carpentries audience

We often qualify Carpentries learners as **novices**: they do not know what they need to learn yet. A typical example is the usage of version control: the Carpentries lesson Version

💬 History

- We have started by porting own lessons to the Carpentries teaching style and format, and collaboratively and iteratively grew and improved the material to its present form.

The funding from 2022-2025 is designed to keep this project active beyond 2025 by forming a support network and building a community of instructors and contributors.

- TQC material addressed academic disciplines and tries to be as programming language-independent as possible.
- The CodeRefinery project idea grew out of two Software Carpentry courses given at CSC Finland in 2014 and 2015. The project proposal itself was submitted to NeIC in 2015, accepted in 2015, and started in 2016.
- Provide a [training community hosting service](#) that is open and free for all researchers based in universities and research institutes from Nordic countries.

The Carpentries aims to teach computational **competence** to learners through an applied approach, avoiding the theoretical and general in favor of the practical and specific.

Mostly, learners do not need to have any prior experience in programming. One major goal of a Carpentries workshop is to raise awareness on the tools researchers can learn/use to speed up their research.

❗ Novices

The Carpentries aims to teach computational **competence** to learners through an applied approach, avoiding the theoretical and general in favor of the practical and specific.

Mostly, learners do not need to have any prior experience in programming. One major goal of a Carpentries workshop is to raise awareness on the tools researchers can learn/use to speed up their research.

Target audience

One common question we get is how do we relate to the Carpentries. This section describes how we see it:

Carpentries audience

We often qualify Carpentries learners as **novices**: they do not know what they need to learn yet. A typical example is the usage of version control: the Carpentries lesson Version Control with Git aims to give a very high level conceptual overview of Git but it does not explain how it can be used in research projects.

CodeRefinery audience

In that sense, CodeRefinery workshops differ from Carpentries workshops as we assume our audience already writes code and scripts and we aim at teaching them **best software practices**.

Our learners usually do not have a good overview of **best software practices** but are aware of the need to learn them. Very often, they know of the tools (Git, Jupyter, etc.) we are teaching but have difficulties to make the best use of them in their software development workflow.

Whenever we can, we direct learners that do not have sufficient coding experience to Carpentries workshops.

Competent practitioners

We often qualify CodeRefinery learners as **competent practitioners** because they already have an understanding of their needs.

Challenges related to defining our target audience

We often get the feedback "I wish I would have known X earlier!" *Competent practitioners* have run into issues with **not** caring (or not fully understanding) about version control, documentation, modularity, reproducibility before, so they are easily motivated to learn more.

For a *novice* these topics may seem unnecessary and "too much" and the workshop may feel too difficult to follow. However, the materials are designed so that one can always revisit a topic, when needed. The important part is that you know that "X" exists, and where to find more information, which is also beneficial for novices.

- We want more people to work with us, and to work with more people

Collaborative notes

Keypoints: CodeRefinery
Objectives

- Teaches intermediate-level software development tool lessons
- Be able to provide a highly interactive online workshop environment with
- It is difficult to define "best practices", we try to teach **"good enough" practices** collaborative documents
- Training network for other lessons
- Publicly-funded discrete projects (3 projects actually) transitioning towards an open community project

Instructor note

- We have online material, teaching, and exercise sessions
- Teaching: 10 min
- Our main target audience are competent practitioners, but also novices and experts
- Exercise: 15 min can get something out of the workshops
- Questions & Answers: 5 min

where to find more information, which is also beneficial for novices.

- We want more people to work with us, and to work with more people

Collaborative notes

Keypoints: CodeRefinery
Objectives

- Teaches intermediate-level software development tool lessons
- Be able to provide a highly interactive online workshop environment with
- It is difficult to define "best practices", we try to teach **"good enough" practices** collaborative documents
- Training network for other lessons
- Publicly-funded discrete projects (3 projects actually) transitioning towards an open community project

Instructor note

- We have online material, teaching, and exercise sessions
- Teaching: 10 min
- Our main target audience are competent practitioners, but also novices and experts
- Exercise: 15 min
- can get something out of the workshops
- Questions & Answers: 5 min

Introduction

The Collaborative document is how you interact with the participants. The participants can ask questions and give feedback through the collaborative document. During a CodeRefinery session there can be a large a volume of questions. A dedicated person, a Collaborative document-manager, is needed to answer and edit the collaborative document. Let us see how the collaborative document is used, then discuss the role of the editor or collaborative document-manager.

Collaborative document mechanics and controls

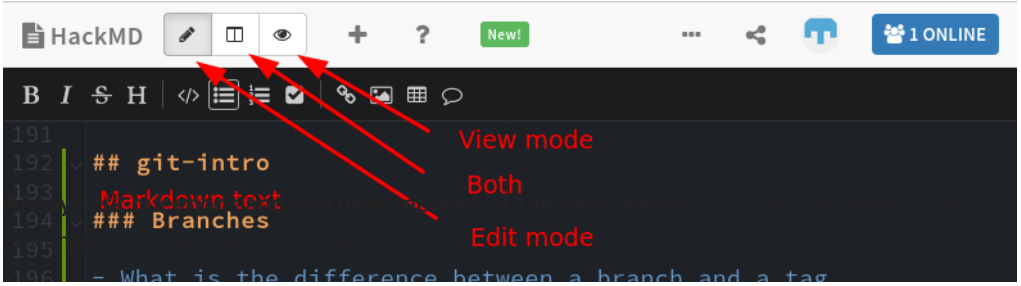
Technologies that can be used as a collaborative document are Hackmd, HedgeDoc, or Google Docs

Hackmd or HedgeDoc are real-time text editor online. We use it to:

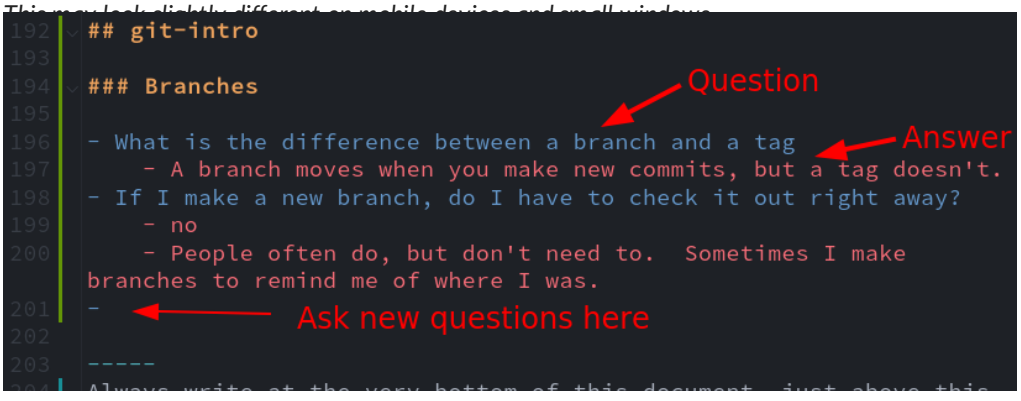
- As a threaded chat, to **answer questions and provide other information** without interrupting the main flow of the room.
- provide everyone with a **more equal opportunity to ask questions**.
- **create notes** which will be archived, for your later reference.

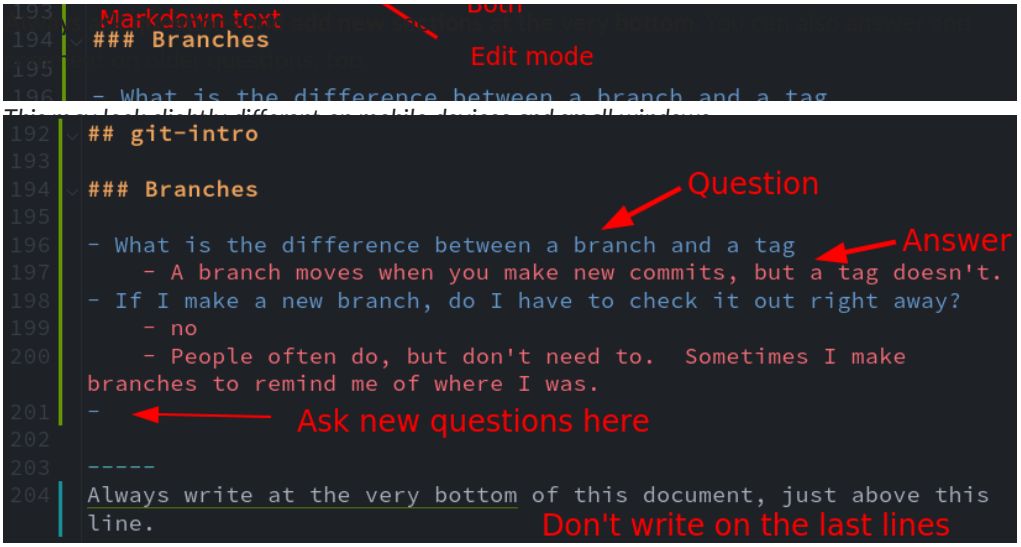
You do not need to login/create an account to be able to edit the document.

Basic controls



This may look slightly different on mobile devices and small windows





Questions and answers in bullet points

Since we plan to publish the questions and answers later as part of the workshop page, we recommend to not use any names. You can indicate your own name to make it easier to discuss more during the workshop but then always use this form: `[name=Myname]`. This makes it easier for us to automatically remove all names before publishing the notes.

Other hints:

- Use `+1` to agree with a statement or question (we are more likely to comment on it).
- Please leave some blank lines at the bottom
- NOTE: Please don't "select all", it highlights for everyone and adds a risk of losing data (there are periodic backups, but not instant).
- It can be quite demanding to follow the collaborative document closely. Keep an eye on it, but consider how distracted you may get from the course. For things beyond the scope of the course, we may come back and answer later.

Don't get overwhelmed

There can be a flood of information on the collaborative document. Scan for what is important, then if you would like come back later. But it is important to keep checking it.

Privacy

- Assume the collaborative document is **public and published: you never need to put your name there**.
- The collaborative document will be **published on the website afterwards**. We will remove all non-instructors names, but it's easier if you don't add it there in the first place.
- Please keep the link private during the workshop, since since security is "editable by those who have the link".
- You can use `[name=YOURNAME]`, to name yourself. We *will* remove all names (but not the comments) before archiving the notes (use this format to make it easy for us).

Collaborative Document Manager Exercise

✍ Discuss how to collaborate and handle questions (15 min)

- Write down your conclusions in the shared document. Items to discuss are:
- What is your experience with questions and discussions while teaching?
- How do you deal with them?
- What kind of technologies do you prefer: chat, shared document, or voices and discussion based during instruction? And why?

Below (*) = important

Before the workshop

- Create a new collaborative document for the workshop
- make sure that **editing is enabled for anyone without login**

Collaborative Document Manager Exercise

✍️ Discuss how to collaborate and handle questions (15 min)

- Write down your conclusions in the shared document. Items to discuss are:
- What is your experience with questions and discussions while teaching?
- How do you deal with them?
- What kind of technologies do you prefer: chat, shared document, or voices and discussion takes during instruction? And why?

Before the workshop

- Create a new collaborative document for the workshop
- make sure that **editing is enabled for anyone without login**
- Add workshop information, links to the workshop page and material and an example question and answer to the top of the collaborative document(see below)

Most things to edit (everyone)

Make it easy to post after the course and consistent to follow:

- Tag all names with `[name=XXX]` (so they can be removed later), remove other personal data or make it obvious.
- Add in information on exercises (new section for them, link, end time, what to accomplish)
- Make a logical section structure (`#` for title, `##` for sections, `###` for episodes, etc. - or what makes sense)

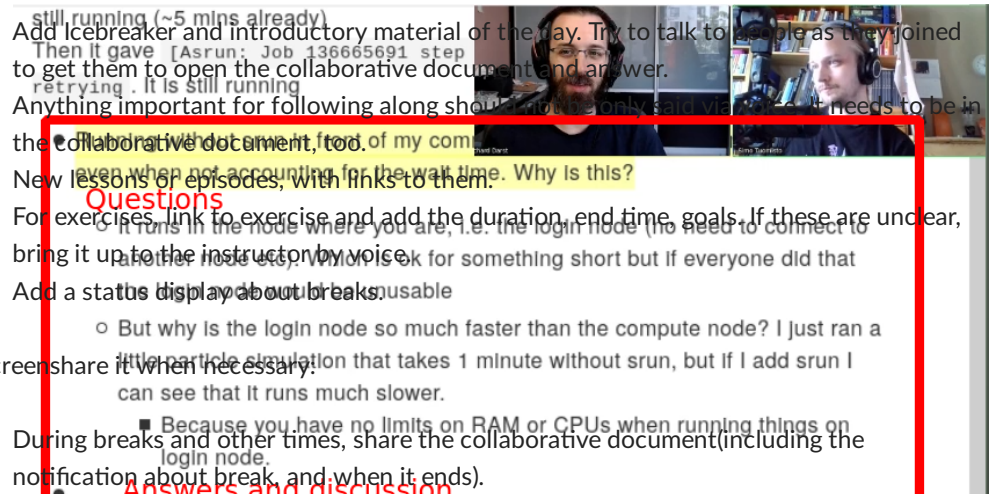
General Collaborative Document practices

Keep it formatted well:

- (*) Tag names you see with `[name=XXX]` so that we can remove it later.
- Heading level `#` is only the page title
- Add a new `##` heading when a new *lesson* or similar thing is started (introduction, icebreaker, break between lessons, etc)
- Add a new `###` heading when a new *episode*, *exercise*, *break* (within exercise session)
- Ensure people are asking questions at the bottom, direct them there if they aren't.
- (*) Ensure each question is a bullet point. Each answer or follow-up should be a bullet point below.
  - Should you use more deeply nested bullet points, or have only one level below the initial question? It depends on the context, but if a conversation goes on too long, try not to let it go too deep.

Update with meta-talk, so that learners can follow along easily:

- Add Icebreaker and introductory material of the day. Try to talk to people as they joined to get them to open the collaborative document and answer.
- Anything important for following along should be only said via voice. It needs to be in the collaborative document, too.
- New lessons or episodes, with links to them.
- For exercises, link to exercise and add the duration, end time, goals. If these are unclear, bring it up to the instructor by voice.
- Add a status display about breaks.



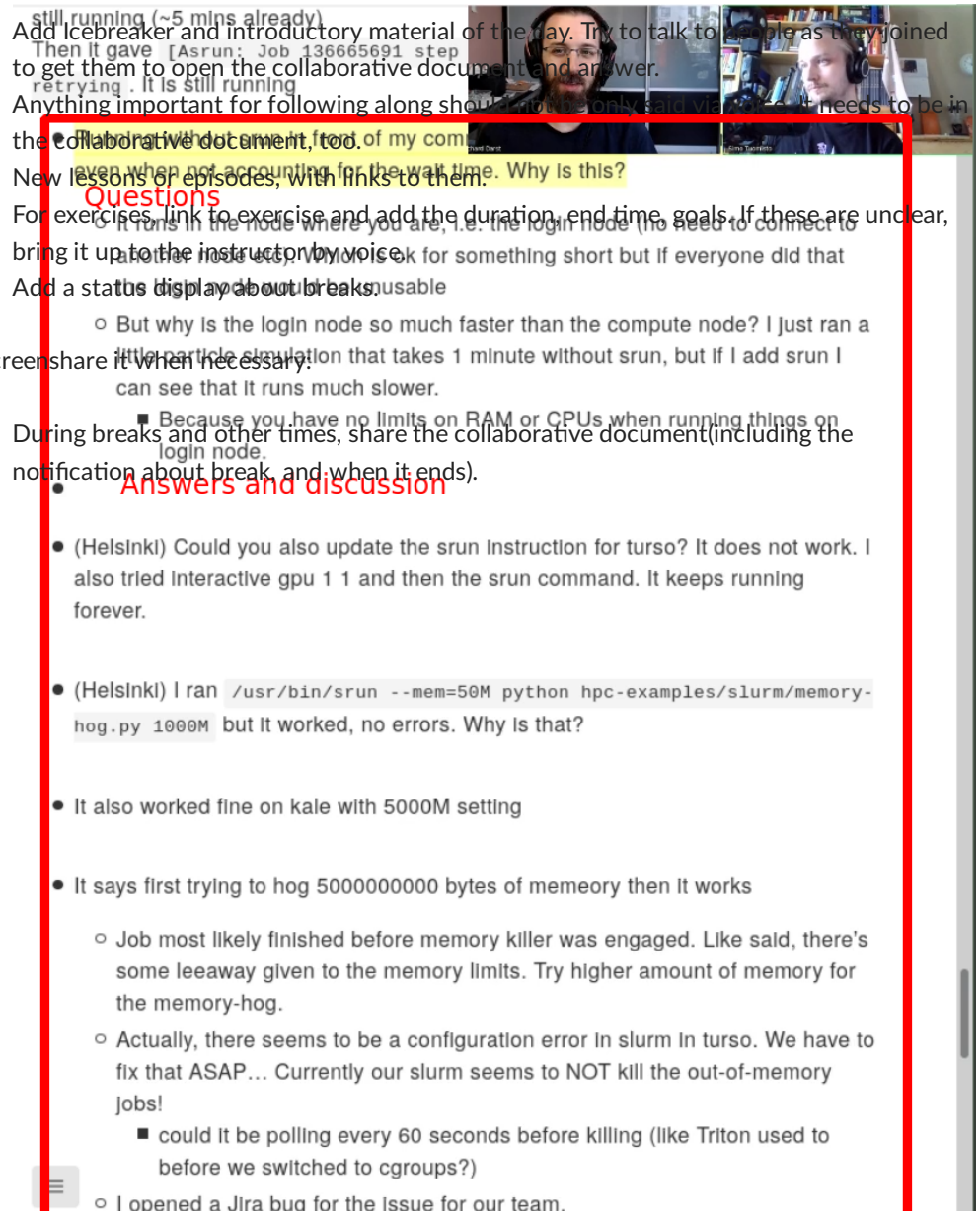
Screenshare it when necessary.

- During breaks and other times, share the collaborative document(including the notification about break, and when it ends).

- Add icebreaker and introductory material of the day. Try to talk to people as they joined to get them to open the collaborative document and answer.
- Anything important for following along should not be only said via voice, it needs to be in the collaborative document, too.
- New lessons or episodes, with links to them.
- For exercises, link to exercise and add the duration, end time, goals. If these are unclear, bring it up to the instructor by voice.
- Add a status display about breaks.

Screenshare it when necessary.

- During breaks and other times, share the collaborative document(including the notification about break, and when it ends).



A live demo of a Collaborative Document during a Q&A time. The two instructors are discussing some of the import answers. Multiple learners have asked questions, multiple answers, and some remaining to be answered

- It is nice if the arrangement allows some of the latest questions to be seen, so people are reminded to ask there.
- Someone else may do this, but should make sure it happens.

Answer questions

- If there is an question that should be answered by the instructor by voice, bring it up (by voice) to the instructor immediately.
  - Declare it advanced and that you will come back later.
- How soon do you answer questions? Two points of view:
  - Answer questions right away: can be really intense to follow.

Ensure it can be posted quickly:

  - Wait some so that we don't overload learners: reduces the info flow. But then do people need to check back more often.
  - You need to find your own balance. Maybe a quick answer right away, and more detailed later. Or delay answers during the most important parts of the lecture.
- Avoid wall-of-text answers. If reading an answer takes too long, it puts the person (and other people who even try to read it) behind even more by taking up valuable mental energy. If an answer needs a wall of text, consider these alternatives:
  - Progressive bullet points getting more detailed (first ones useful alone for basic cases)
  - There may be other private URLs at the top or bottom.
  - Don't be worried to say "don't worry about this now, let's talk later."
  - Figure out the root problem instead of answering every possible interpretation (though others can do this, too).

- The collaborative document gets posted to the workshop webpage. For this, it needs some minimal amount of formatting (it doesn't need to be perfect, just not horrible).
- All names and private information needs to be stripped. This is why you should rigorously tag all names with `[name=XXX]` so they can be removed (see above).
  - Learner names can be completely removed, CR staff names can be `[name=CR]` or something similar.
- If possible, send the PR adding the collaborative document to the workshop webpage (though others can do this, too).

voice) to the instructor immediately.
    - Declare it advanced and that you will come back later.
- How soon do you answer questions? Two points of view:
    - Answer questions right away: can be really intense to follow.
Ensure it can be posted quickly:
    - Wait some so that we don't overload learners: reduces the info flow. But then do people need to check back more often.
- The collaborative document gets posted to the workshop webpage. For this, it needs
    - You need to find your own balance. Maybe a quick answer right away, and more some minimal amount of formatting (it doesn't need to be perfect, just not horrible). detailed later. Or delay answers during the most important parts of the lecture.
- All names and private information needs to be stripped. This is why you should rigorously
- Avoid wall-of-text answers. If reading an answer takes too long, it puts the person (and tag all names with `[name=XXX]` so they can be removed (see above).
other people who even try to read it) behind even more by taking up valuable mental
    - Learner names can be completely removed, CR staff names can be `[name=CR]` or energy. If an answer needs a wall of text, consider these alternatives: something similar.
    - Progressive bullet points getting more detailed (first ones useful alone for basic cases)
    - There may be other private URLs at the top or bottom.
    - Don't be worried to say "don't worry about this now, let's talk later."
- If possible, send the PR adding the collaborative document to the workshop webpage
    - Figure out the root problem instead of answering every possible interpretation (though others can do this, too).

Collaborative document format example

```
# Workshop, day 1


## Lesson name
https://coderefinery.github.io/lesson/

### Episode name
https://coderefinery.github.io/01-episode/

- This is a question
  - Anwser
  - More detailed answer
- question
  - answer

### Exercises:
https://link-to-exercise/.../.../#section
20 minutes, until xx:45
Try to accomplish all of points 1-3.  Parts 4-5 are optional.

Breakout room status:
- room 2, need help with Linux permissions
- room 5, done

### Break
:::danger
We are on a 10 minute break until xx:10
:::


## Lesson 2
https://coderefinery.github.io/lesson-2/
```

Posting the collaborative document to the website

The collaborative document should be posted sooner rather than later, and hopefully the steps above will make it easy to do so quickly. You could wait a few hours, to allow any

```
## Feedback, day N

:::info
### News for day N+1
- .
- .
:::

### Today was (multi-answer):
- too fast:
- just right:
- too slow:
- too easy:
- right level:
- too advanced:
- I would recommend this course to others:
- Exercises were good:
- I would recommend today to others:
```

```
## Feedback, day N

:::info
### News for day N+1
- .
- .
:::

### Today was (multi-answer):
- too fast:
- just right:
- too slow:
- too easy:
- right level:
- too advanced:
- I would recommend this course to others:
- Exercises were good:
- I would recommend today to others:
- I wouldn't recommend today:

### One good thing about today:
- ...
- ...

### One thing to be improved for next time:
- ...
- ...

### Any other comments:
- ...
- ...
```

Keypoints

- Having a collaborative document improves communication and interaction.
- Answering questions requires a dedicated person - A Collaborative Document Manager.
- The collaborative document should be posted on the web site as soon as possible.

A workshop seen from different perspectives

Objectives

- Understand the general structure of CodeRefinery workshops and why it is the way it is
- Get to know the different roles of a workshop and which ones are the most essential ones
- Understand the importance of installation instructions and how they contribute to learners success
- Understand the importance of onboarding and a welcoming community for volunteers in a workshop

Instructor note

- Teaching: 20 min
- Exercises: 15 min
- Roles journeys can be shortened to looking only at instructor role.

Discussion

💬 Discussion

In breakoutrooms: choose one or two questions to talk about your own experiences; summary in collaborative notes:

- What kind of roles have you been in yourself regarding workshops?
- How were you prepared for your role?
- What are the things you would like to know before a workshop?
- How are you preparing your participants for your trainings?
- What was the best workshop experience for you as learner, helper or instructor? What made it great?

Note

This episode focuses on the large scale streamed workshop setup. Some of the features can also be applied to smaller scale workshops though.

Discussion

- Teaching: 20 min
- Exercises: 15 min
- Roles journeys can be shortened to looking only at instructor role.

Discussion

In breakoutrooms: choose one or two questions to talk about your own experiences; summary in collaborative notes:

- What kind of roles have you been in yourself regarding workshops?
- How were you prepared for your role?
- What are the things you would like to know before a workshop?
- How are you preparing your participants for your trainings?
- What was the best workshop experience for you as learner, helper or instructor? What made it great?

Note

This episode focuses on the large scale streamed workshop setup. Some of the features can also be applied to smaller scale workshops though.

One workshop - many parts

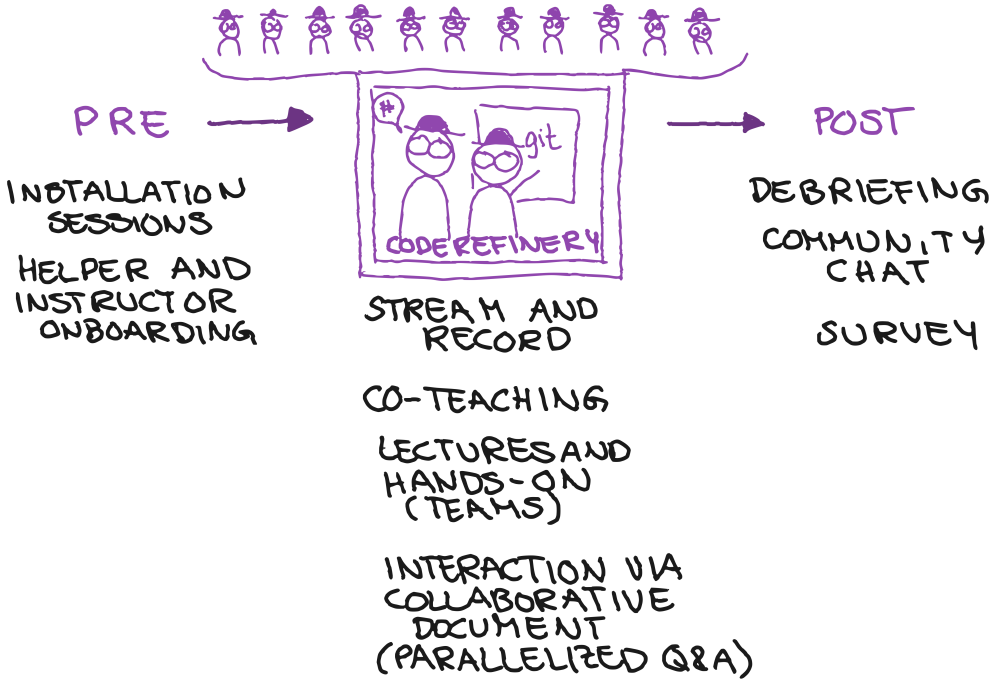


Image showing the different phases of a CodeRefinery workshop described below. Also showing that while there is only two people teaching, many people work behind the scenes.

Before the workshop

- During winter/summer: “Someone” takes the coordinator role for the next workshop
- Coordinator fills other roles:
  - Coordinator: collects necessary lesson updates, supports other coordinators
  - Registration coordinator: Sets up web page, starts and manages registration
  - Instructor coordinator: Finds instructors and onboards them
  - Advertising coordinator: Prepares advertizement texts and finds people to distribute them
  - Team coordinator: Gathers local organizers/teams
  - Bring your own code session coordinator: If available, offer BYOC session after main workshop
- Instructor: Onboarding with (instructor-) coordinator and previous instructor of lesson; update of lesson materials (urgent issues and personal touches)
- Local organizer/Team lead: Group onboarding ~ week/or two before workshop
- Learner: Gets installation instructions, invited to installation help session, workshop info from event page and summary via e-mail
- Collaborative notes manager sets up the notes document

Note

Onboarding manuals

- Outline of what will happen during the workshop
- Discussion of different strategies to handle the exercise sessions
- Lowering the barrier to ask for support by meeting some organizers
- Q&A

Note

Installation instructions

- Instructions for all operating systems

- Registration coordinator: Sets up web page, starts and manages registration
- Instructor coordinator: Finds instructors and onboards them
- Advertising coordinator: Prepares advertizement texts and finds people to distribute them
- Team coordinator: Gathers local organizers/teams
- Bring your own code session coordinator: If available, offer BYOC session after main workshop
- Q&A

Note

Onboarding manuals

- **Outline of what will happen during the workshop**
- Discussion of different strategies to handle the exercise sessions
- Lowering the barrier to ask for support by meeting some organizers

- Instructor: Onboarding with (instructor-) coordinator and previous instructor of lesson; update of lesson materials (urgent issues and personal touches)
- Local organizer/Team lead: Group onboarding ~ week/or two before workshop
- Learner: Gets installation instructions, invited to installation help session, workshop info from event page and summary via e-mail
- Collaborative notes manager sets up the notes document

Note

Installation instructions

- Instructions for all operating systems
- Goal: Learners leave the workshop and are ready to apply the learned tools and techniques to their own work
- Support session for installation challenges

During the workshop

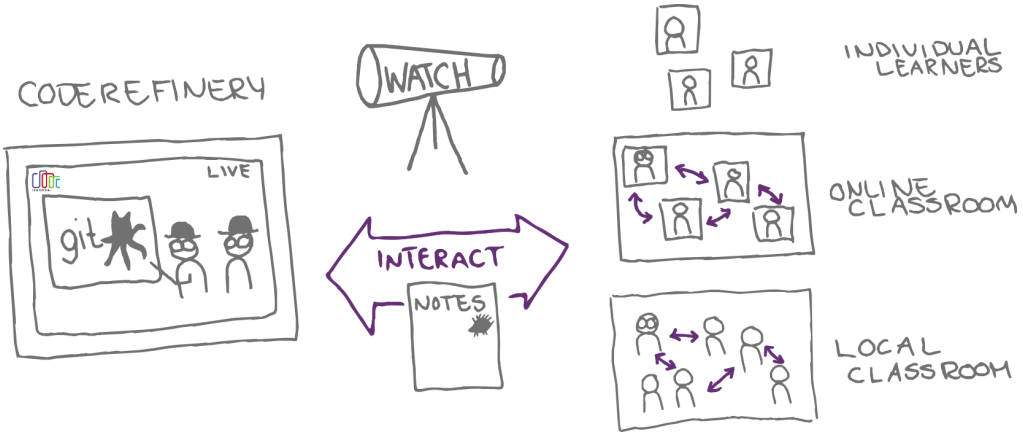


Image showing the different ways learners can do the exercises. No matter which way they choose, they always watch the stream and interact with instructors and organizers via collaborative notes. Individual learners do the exercises alone, people in online or local classrooms in addition get to discuss with their peers.

- Everyone watches stream
- Co-instructors - Present content - Answer questions in collaborative notes
- Collaborative notes manager
  - Keeps the collaborative notes clean
  - Helps answering questions
  - Adds sections
  - Archives the document after each day
- Learners do exercises individually or in team
- Local organizer / Team lead
  - Guides learners through exercises
  - Facilitates discussion
- Broadcaster prepares and shares recording (see session 4)
- (Registration-) coordinator sends out e-mails to participants after each day (summary + preparation)
- Director/Broadcaster manages the streaming (see session 4)

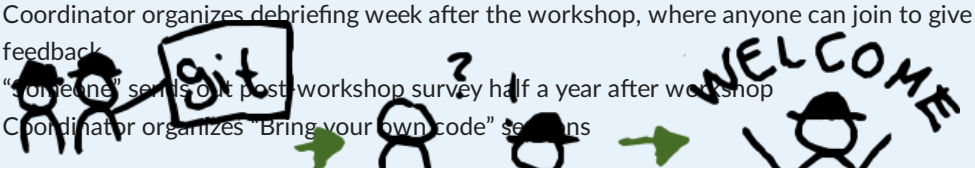
Note

Bring your own code sessions

We want to support learners to apply what they have learned in the workshop to their own work by offering one/two sessions where they can drop in, and discuss challenges with CodeRefinery instructors and helpers.

After the workshop

- Coordinator collects lessons learned based on experience and feedback and turns them into issues
- Coordinator organizes debriefing week after the workshop, where anyone can join to give feedback
- "someone" sends out post-workshop survey half a year after workshop
- Coordinator organizes "Bring your own code" sessions



- Facilitates discussion
- Broadcaster prepares and shares recording (see session 4)
- (Registration-) coordinator sends out e-mails to participants after each day (summary + preparation)
- Director/Broadcaster manages the streaming (see session 4)

Note

Bring your own code sessions

We want to support learners to apply what they have learned in the workshop to their own work by offering one/two sessions where they can drop in, and discuss challenges with CodeRefinery instructors and helpers.

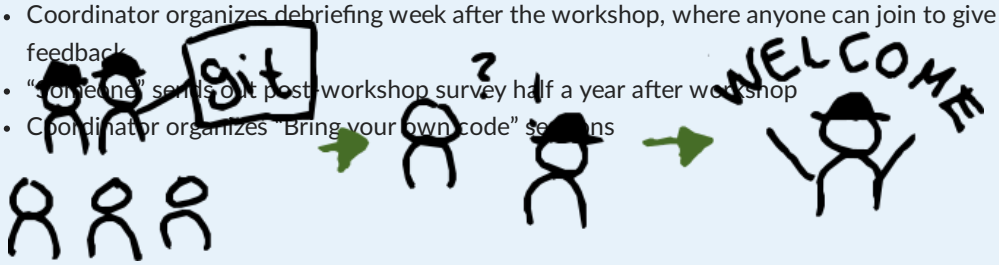


Image showing a classroom learning about git leading to a discussion between student and teacher to start a welcoming environment for asking for help.

After the workshop

- Coordinator collects lessons learned based on experience and feedback and turns them into issues
- Coordinator organizes debriefing week after the workshop, where anyone can join to give feedback
- "someone" sends out post-workshop survey half a year after workshop
- Coordinator organizes "Bring your own code" sessions

This sounds like a lot of people and time investment!

- Many roles can be combined or adjusted as needed
- Some roles can be taken on by multiple people

So how many people are needed for this kind of workshop?

- Recommended minimum: 2-3 people
- Optimum: ~ 4-5 people The more people you involve, the more time goes into coordination work.

How did we get to this setup?

- **Collaborative** in-person workshops since 2016 around the Nordics
- Moved online in 2020
- Started with "traditional zoom" workshops, breakoutrooms, volunteer helpers
- Thought about scaling and ease of joining -> current setup (More about this in session 4)
- Continuous development

We also still do smaller scale local workshops, if instructors are available. **You can offer your own CodeRefinery workshop** by inviting other CodeRefinery instructors. You can also reuse single lessons and integrate them into your own teaching.

Workshop roles and their journeys

Individual learner journey

- Registration
- Installations
- Lesson material updates
- Workshop
  - Watch stream
  - Q&A in notes
- Debrief

Team lead / Local host journey

- Registration
- Post workshop survey

Instructor journey

- Indicate interest in CodeRefinery chat
- Onboarding
- (if applicable: invited to local meetup)
- Teaching
- (if wished: supports answering questions in notes)
- Debrief
  - Daily feedback
  - Debrief feedback session
- (if applicable: run own registration)
- Onboarding
- Workshop
  - Watch stream

- Indicate interest in CodeRefinery chat
- Registration
- Onboarding
- Installations
- Lesson material updates
- (if applicable: invited to local meetup)
- Teaching
- Workshop
- (if wished: supports answering questions in notes)
- Watch stream
- Debrief
- Q&A in notes

Team lead / Local host journey

- Onboarding
- Lesson material updates
- Daily feedback
- Registration
- Debrief / feedback session
- (if applicable: run own registration)
- Post workshop survey

Instructor journey

- Onboarding
- Workshop
  - Watch stream
  - Facilitate exercises/discussions
  - Daily feedback
- Debrief / feedback / survey

Other roles

Director and broadcaster roles will be discussed in Session 4. For more in-depth descriptions and other roles (host, instructor, registration coordinator, etc), see also our CodeRefinery manuals.

Different roles as stepping stones for community involvement

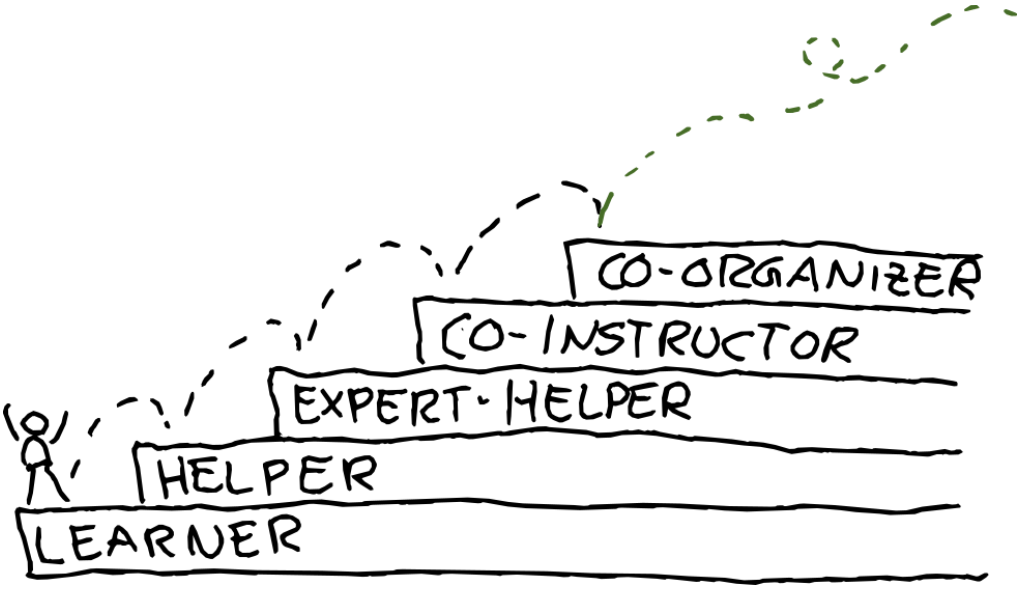


Image showing a person jumping from step to step following learner, helper, expert helper, co-instructor, co-organizer.

There is no “one way” to get involved. Do what feels right. Interested in teaching with us? Let us know and we can discuss what is a suitable path **for you**!

Instructor note

Keypoints

- Teaching: 10 min
- CodeRefinery workshops are a collaborative effort with many different roles
- Exercises: 10 min
- Onboarding the different roles is one key aspect of our workshops
- Everyone has their own path

The importance of audio

Sound

- Pleasing audio quality makes events much more enjoyable
- Audio is one of the most important things you can control
- Things that can go wrong:
  - Too quiet
  - Too loud
  - Instructors’ volumes imbalanced

Objectives

- Understand that sound quality is very important
- Evaluate your and others sound quality and know how to improve it
- Test tips and tricks for achieving good sound quality

Instructor note

Keypoints

- Teaching: 10 min
- Exercises: 10 min

Objectives

- Understand that sound quality is very important
- Evaluate your and others sound quality and know how to improve it
- Test tips and tricks for achieving good sound quality

The importance of audio

Sound

- Pleasing audio quality makes events much more enjoyable
- Audio is one of the most important things you can control
- Things that can go wrong:
  - Too quiet
  - Too loud
  - Instructors' volumes imbalanced
  - Background noise
  - Low-quality/breaking up audio hard to hear.
  - "Ducking" (first words lower volume, or lower volume other times)

Tips for good sound quality

- Have a headset with mounted microphone
  - Even if you have a professional external microphone, it doesn't matter if your room has bad acoustics.
  - The close pickup can't be beat with normal tools.
  - As long as it's headset mounted, price doesn't seem to matter *that* much.
- Don't use Bluetooth
  - Bluetooth can have too much latency (300-500ms)
  - This may seem small but for interactive work, it's a lot
  - Use a wired headset, or wireless with a non-Bluetooth USB plug (like gaming headsets have). These have much lower latency.
  - Bluetooth 5 can have much lower latency, but you probably shouldn't count on that without testing.
  - It can also have lower sound quality on some devices due to bandwidth limitations.
- Once you have a headset, turn input noise cancellation to low (wherever it might be: headphone, meeting software, etc.).

Balancing and dynamic adjustment

- It's important that instructors volumes match.
- An exercise will go over a systematic procedure for matching volumes.
  - Practice so that you can do this quickly.
- May need re-adjusting when instructors swap out or start getting excited.
- An exercise below will demonstrate our procedure.

Speak up when there are problems

- If you notice someone with audio issues, let them know right away (voice if bad, or chat/notes if less urgent).
- Be polite and practical.
- Make a culture of *speaking up, helping, and not suffering*.

Exercises

✍ Sound-1: Evaluate sound quality

It's important to be able to discuss with others the quality of their audio. They can't hear themselves, after all.

Recommendations

- Within the teams, discuss each person's audio quality and what kind of setup they have.
- Procure some reasonable headset.
- Low/medium-priced gaming-type headsets have worked well for us.
  - (gaming headsets usually aren't Bluetooth, because gaming needs low latency.)
- Be respectful and constructive (and realize that people came prepared to listen, not teach).
- Show this page to your workplace (if you have one) and call a good headset work equipment.
- Consider, for example
  - Volume
  - Clarity

Exercises

✍️ Sound-1: Evaluate sound quality

It's important to be able to discuss with others the quality of their audio. They can't hear themselves, after all.

Recommendations

- Within the teams, discuss each person's audio quality and what kind of setup they have.
- Procure some reasonable headset.
- Low/medium-priced gaming-type headsets have worked well for us.
  - (gaming headsets usually aren't Bluetooth, because gaming needs low latency.)
- Show this page to your workplace (if you have one) and call a good headset work equipment.
- Consider, for example
  - Volume
  - Clarity
  - Background noise
  - Noise cancellation artifacts
  - "Ducking": first words at lower volume or missing
- Discuss how to bring this up during other courses and meetings.

✍️ Sound-2: Adjust volume up and down

In addition to individual quality, it is important that sound is balanced among all instructors. Have you ever been to a event when one person is too quiet and they can't make themselves any louder? And they can't adjust it?

You should ensure that you have a range of volumes available to you. You might have to look into several different locations in order to make the full adjustment ().

- Go to your sound settings
- One by one, each person
  - Adjusts their microphone volume so quiet that they can't be heard.
  - Adjusts their microphone volume so that it is extremely loud (this may require going beyond 100% if possible).
- Basically, make sure you aren't so close to either end that you have no potential to make an adjustment in that direction.
- Everyone tries to set the volume to something reasonable, in preparation for the next exercise.

Once you know where these settings are , you won't be panicked when the volume is too low or high during a course.

✍️ Sound-3: Do a balance check

It's important that instructor audio is balanced (the same volume). But how can you do this?

- Pick a leader.
- The leader decides the order ("I am first, then [name-1] and [name-2]")
- The leader says "one". Everyone else says "one" in the order specified.
- The leader says "two". Everyone else says "two" in the order.
- The leader asks for opinions on who they think is louder or softer. If there are more than three people, you can figure it out yourselves. With less than two, you have to ask someone in the audience.
- Leader: How did that sound to everyone?
- [someone else]: Leader and BBB were pretty similar but AAA is a bit lower.

Example:

- Leader: Let's do a sound check. I am first, then AAA and BBB.
- Leader: One
- AAA: One
- BBB: One
- Leader: Two
- AAA: Two
- BBB: Two
- Leader: Three
- AAA: Three
- BBB: Three

Summary

❗ Keypoints

- AAA: One
- BBB: One
- Leader: Two
- AAA: Two
- BBB: Two
- Leader: Three
- AAA: Three
- BBB: Three
- Leader: How did that sound to everyone?
- [someone else]: Leader and BBB were pretty similar but AAA is a bit lower.

Example:

- Leader: Let's do a sound check. I am first, then AAA and BBB.
- Leader: One

Pick a leader.

The leader decides the order ("I am first, then [name-1] and [name-2]")

The leader says "one". Everyone else says "one" in the order specified.

The leader says "two". Everyone else says "two" in the order.

The leader asks for opinions on who they think is louder or softer. If there are more than three people, you can figure it out yourselves. With less than two, you have to ask someone in the audience.

Summary

Keypoints

- Audio quality is important and one of the most notable parts of the workshop.
- Improving audio isn't hard or expensive, but does require preparation.

How to prepare a quality screen-share

Objectives

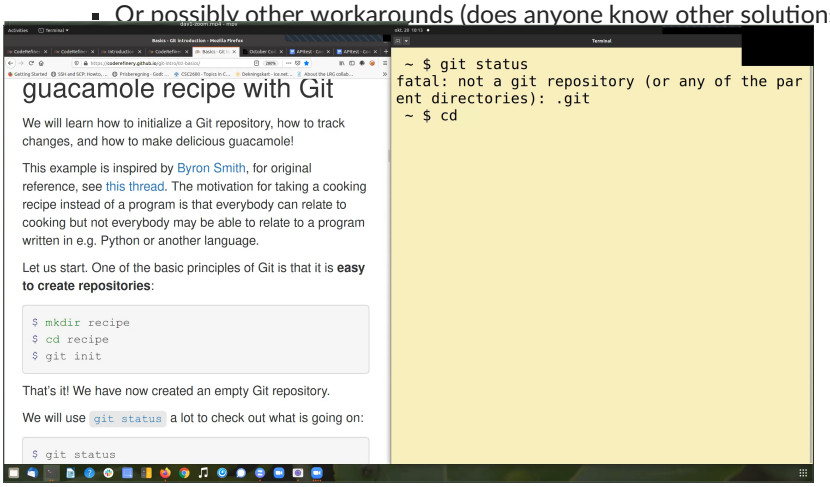
- Discuss the importance of a well planned screen-share.
- Learn how to prepare and how to test your screen-share setup.
- Know about typical pitfalls and habits to avoid.

Instructor note

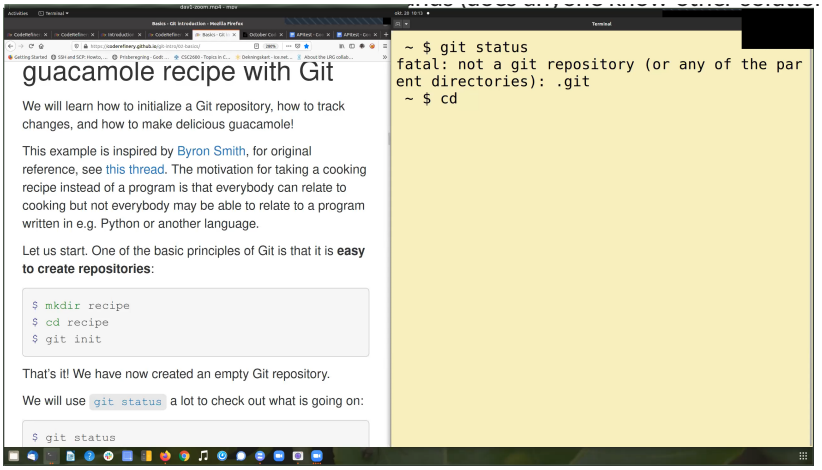
- Discussion: 15 min
- Exercises: 15 min

Share portrait layout instead of sharing entire screen when teaching online

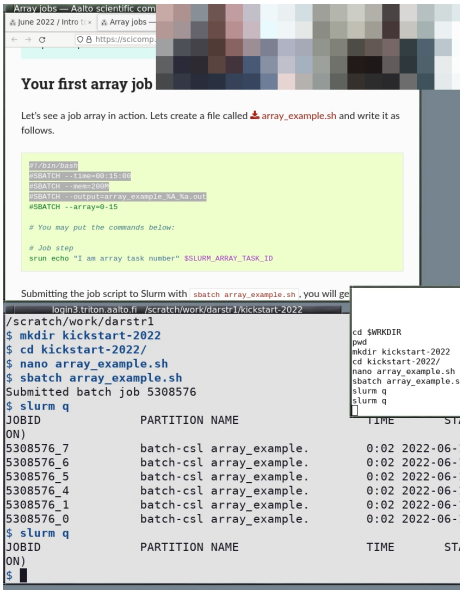
- Many learners will have a smaller screen than you.
- You should plan for learners with **only one small screen**.
- A learner will **need to focus on both your screen share and their work**.
- Share a **portrait/vertical half of your screen** (840 × 1080 is our standard and your maximum).
- Zoom provides a "Share a part of screen" that is good for this.
  - Our latest streaming setup (day 4) can take the portrait part for you so you can share landscape.
  - Zoom + Linux + Wayland display manager doesn't have "Share a portion of the screen"
    - You can share a single window portrait.
    - Or you can start your desktop session in "X11" or "Xorg" legacy mode.
    - Or possibly other workarounds (does anyone know other solutions?)



A FullHD 1920x1080 screen shared. Learners have to make this really small if they want to have



A FullHD 1920x1080 screen shared. Learners have to make this really small if they want to have something else open.

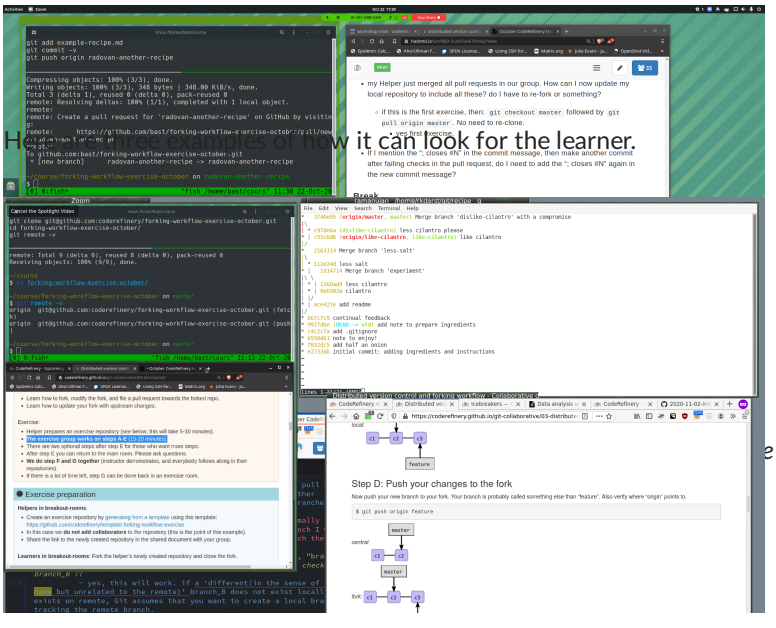


Portrait layout. Allows learners to have something else open in the other half.

Motivation for portrait layout:

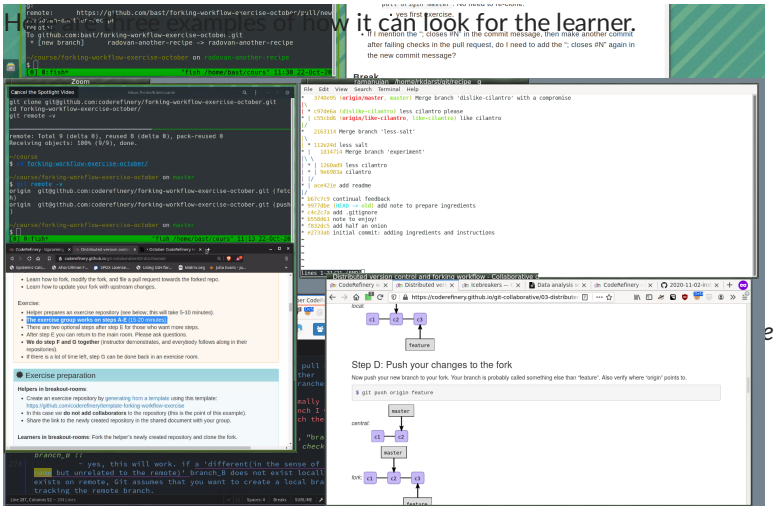
- This makes it easier for you to look at some other notes or to coordinate with other instructors at the same time without distracting with too much information.
- This makes it possible for participants to have something else open in the other screen half: terminal or browser or notebook.

Instructor perspective

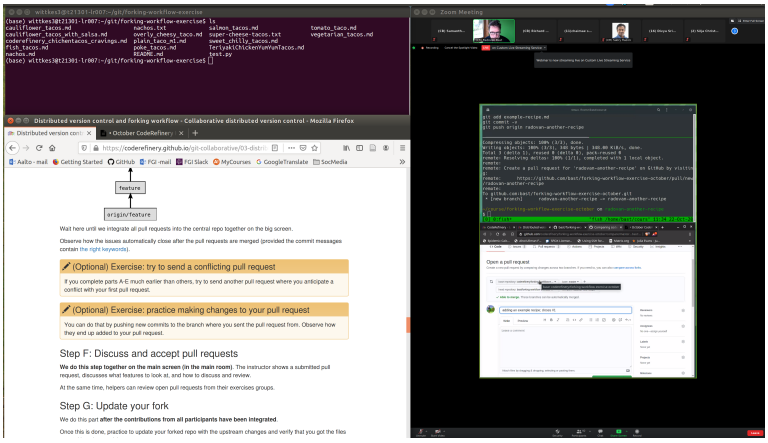


Here are three examples of how it can look for the learner.

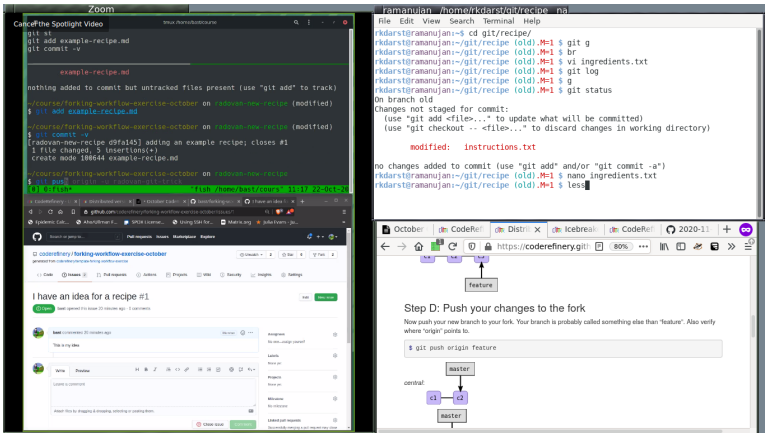
e left side, the right side



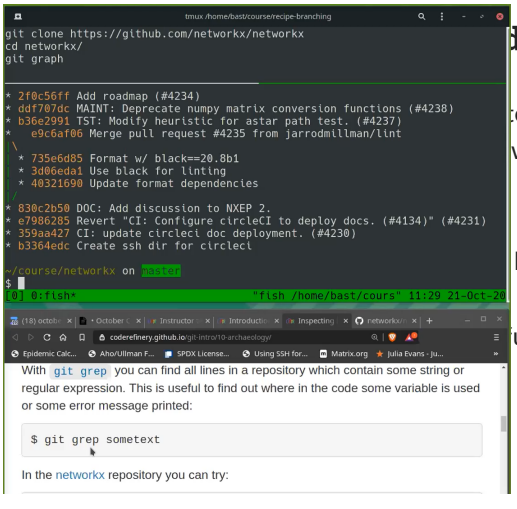
***L1**: Learner with a large screen, Zoom in dual-monitor mode so that the instructor pictures are not shown. Screen-share is on the left side, collaborative notes at bottom left, terminal and web browser on the right.*

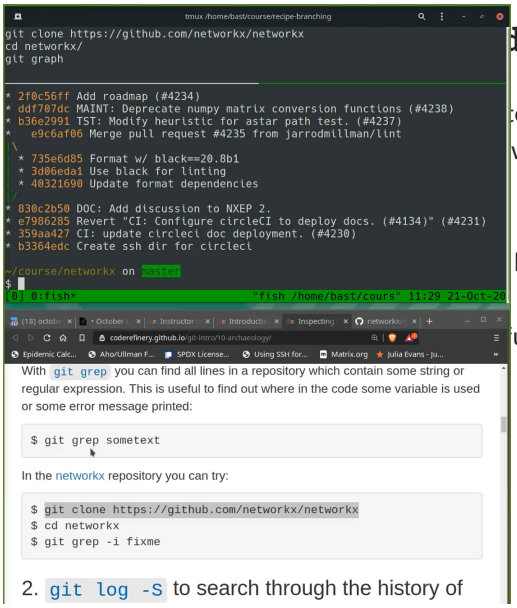


***L2**: A learner with a single large screen (Zoom in "single monitor mode"). Instructor screen share at right, learner stuff at left.*

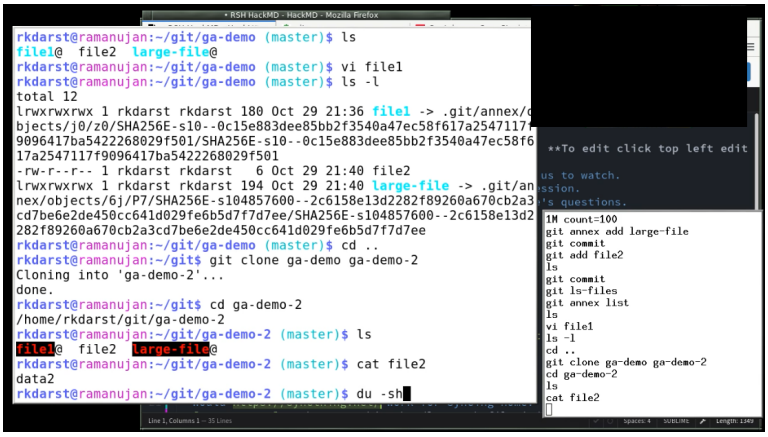


***L3**: A learner with a particularly small screen. Instructor screen-share at left, your windows at right.*

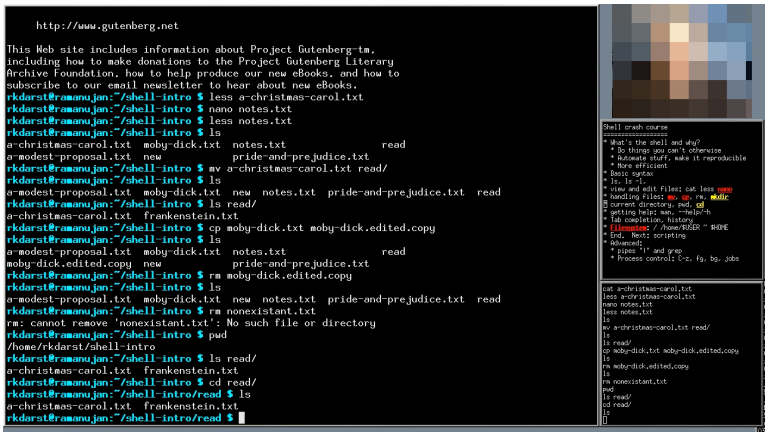


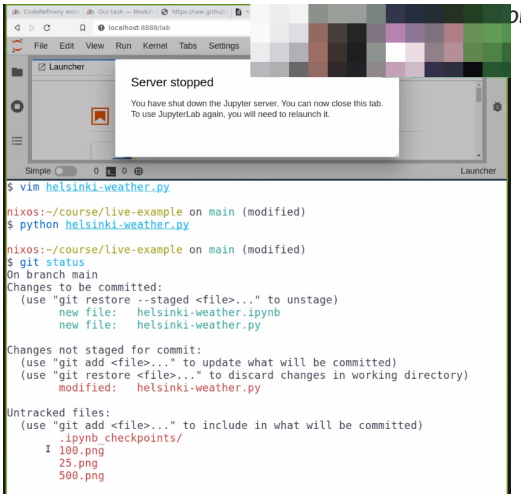


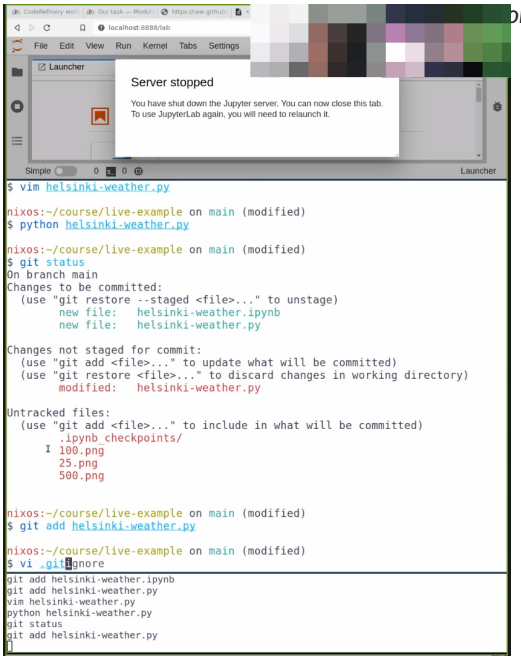
***H1**: A vertical screen layout shared. Note the extra shell history at the top. The web browser is at the bottom, because the Zoom toolbar can cover the bottom bit.*



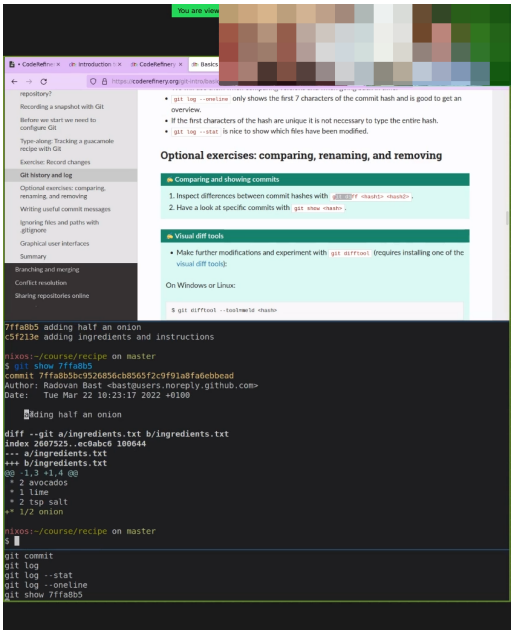
***H2**: This isn't a screen-share from CodeRefinery, but may be instructive. Note the horizontal layout and shell history at the bottom right.*



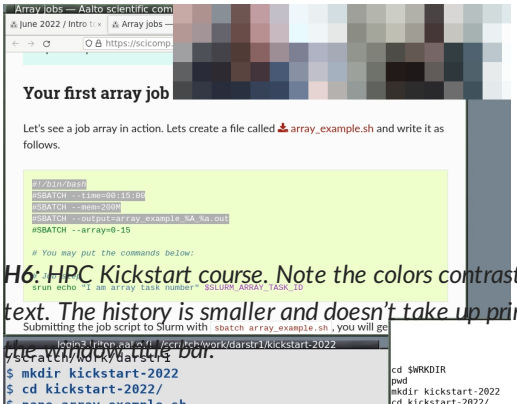




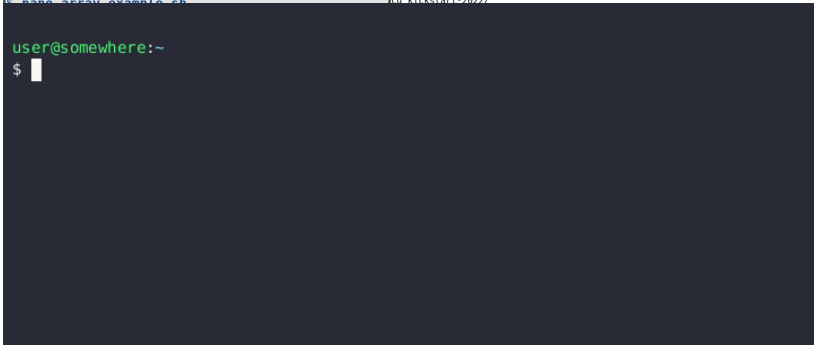
***H4**: Jupyter + terminal, including the `fish` shell and the terminal history.*

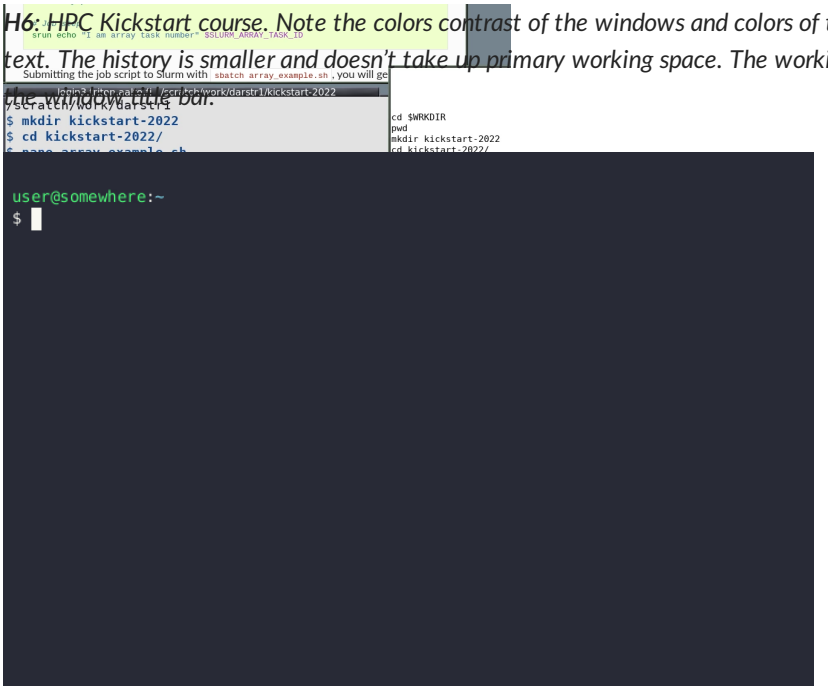


***H5**: Lesson + terminal, `tmux` plus terminal history and dark background.*



***H6**: HPC Kickstart course. Note the colors contrast of the windows and colors of the prompt and text. The history is smaller and doesn't take up primary working space. The working directory is in the window title bar.*





H6: HPC Kickstart course. Note the colors contrast of the windows and colors of the prompt and text. The history is smaller and doesn't take up primary working space. The working directory is in the window title bar.

H7: Show command history "picture-in-picture", in the same terminal window.

How to configure history sharing

You need to find a way to show the recent commands you have entered, so that learners can see the recent commands. Below are many solutions. Try them out and see what works for you.

- prompt-log: It adds a interesting idea that the command you enter is in color and also provides terminal history before the command returns.
- **Simple**: The simple way is `PROMPT_COMMAND="history -a"` and then `tail -f -n0 ~/.bash_history`, but this doesn't capture ssh, sub-shells, and only shows the command after it is completed.
- **Better yet still simple**: Many Software Carpentry instructors use this script, which sets the prompt, splits the terminal window using tmux and displays command history in the upper panel. Requirement: tmux
- **Better (bash)**: This prints the output before the command is run, instead of after. Tail with `tail -f ~/demos.out`.

```
BASH_LOG=~/demos.out
bash_log_commands () {
    # https://superuser.com/questions/175799
    [ -n "$COMP_LINE" ] && return  # do nothing if completing
    [[ "$PROMPT_COMMAND" =~ "$BASH_COMMAND" ]] && return # don't cause a preexec
for $PROMPT_COMMAND
    local this_command=`HISTTIMEFORMAT= history 1 | sed -e "s/^[ ]*[0-9]*[ ]*//"`;
    echo "$this_command" >> "$BASH_LOG"
}
trap 'bash_log_commands' DEBUG
```

- **Better (tmuxp)**: This will save some typing. TmuxP is a Python program (`pip install tmuxp`) that gives you programmable `tmux` sessions. One configuration that works (in this
- **Better (zsh)**: This works like above, with zsh. Tail with `tail -f ~/demos.out` .

```
preexec() { echo $1 >> ~/demos.out }
```

```
session_name: demo
windows:
  - window_name: demo
    layout: main-horizontal
    options:
      main-pane-height: 7
    panes:
      - shell_command:
          - touch /tmp/demo.history
          - tail -f /tmp/demo.history
      - shell_command:
          - function cmd_log --on-event fish_preexec ; echo "$argv" >>
/tmp/demo.history  ; end
```

- **Better (tmuxp)**: This will save some typing. TmuxP is a Python program (`pip install tmuxp`) that gives you programmable `tmux` sessions. One configuration that works (in this
- **Better (zsh)**: This works like above, with zsh. Tail with `tail -f ~/demos.out` .

```
preexec() { echo $1 >> ~/demos.out }
```

```
session_name: demo
windows:
  - window_name: demo
    layout: main-horizontal
    options:
      main-pane-height: 7
    panes:
      - shell_command:
          - touch /tmp/demo.history
          - tail -f /tmp/demo.history
      - shell_command:
          - function cmd_log --on-event fish_preexec ; echo "$argv" >>
/tmp/demo.history  ; end
```

- **Windows PowerShell**: In Windows Terminal, a split can be made by pressing `CTRL+SHIFT+=`. Then, in one of the splits, the following PowerShell command will start tracking the shell history:

```
Get-Content (Get-PSReadlineOption).HistorySavePath -Wait
```

  Unfortunately, this only shows commands after they have been executed.

- Tavatar: shell history mirroring teaching tool can copy recent history to a remote server.
- history-window: Show command history "picture-in-picture" when teaching command line. Requires Bash.

Font, colors, and prompt

Terminal color schemes

- Dark text on light background, *not* dark theme. Research and our experience says that dark-text-on-light is better in some cases and similar in others.
- You might want to make the background light grey, to avoid over-saturating people's eyes and provide some contrast to the pure white web browser. (this was an accessibility recommendation when looking for ideal color schemes)
- Do you have any yellows or reds in your prompt or program outputs? Adjust colors if possible.

Font size

- Font should be large (a separate history terminal can have a smaller font).
- Be prepared to resize the terminal and font as needed. Find out the keyboard shortcuts to do this since you will need it.

Prompt

- Add colors only if it simplifies the reading of the prompt.

At the beginning of the workshop your goal is to have a shell **as easy to follow as possible** Later in the workshop or in more advanced lessons:
and **as close to what learners will see on their screens**:

- Using other shells and being more adventurous is OK - learners will know what is
- Your prompt should be minimal: few distractions, and not take up many columns of text. essential to the terminal and what is extra for your environment.
- prompt-log does this for you.
- The minimum to do is is `export PS1='\$ '` .

Try to find a good balance between:

- Blank line between entries: `export PS1='\n\$ '` .
- Have a space after the `$` or `%` or whatever prompt character you use.
- Showing a simple setup and showing a more realistic setup.
- Strongly consider the Bash shell. This is what most new people will use, and Bash will be
- Showing a consistent setup among all instructors and showing a variety of setups. less confusing to them.

Habits we need to un-learn

- Eliminate menu bars and any other decoration that uses valuable screen space.

Prompt

- Add colors only if it simplifies the reading of the prompt.

At the beginning of the workshop your goal is to have a shell **as easy to follow as possible** and **as close to what learners will see on their screens**:

- Your prompt should be minimal: few distractions, and not take up many columns of text.
- prompt-log does this for you.
- The minimum to do is is `export PS1='\$ '`.

Try to find a good balance between:

- Blank line between entries: `export PS1='\n\$ '`.
- Have a space after the `$` or `%` or whatever prompt character you use.
- Showing a simple setup and showing a more realistic setup.
- Showing a consistent setup among all instructors and showing a variety of setups.
- Strongly consider the Bash shell. This is what most new people will use, and Bash will be less confusing to them.

Later in the workshop or in more advanced lessons:

- Using other shells and being more adventurous is OK - learners will know what is essential to the terminal and what is extra for your environment.

Habits we need to un-learn

- Eliminate menu bars and any other decoration that uses valuable screen space.
- **Do not clear the terminal**. Un-learn CTRL-L or `clear` if possible. More people will wonder what just got lost than are helped by seeing a blank screen. Push `ENTER` a few times instead to add some white space.
- **Do not rapidly switch between windows** or navigate quickly between multiple terminals, browser tabs, etc. This is useful during your own work when nobody is watching, but it is very hard to follow for learners.
- Avoid using **aliases** or **shortcuts** that are not part of the standard setup. Learners probably don't have them, and they will fail if they try to follow your typing. Consider even to rename corresponding files (`.bashrc`, `.gitconfig`, `.ssh/config`, `.ssh/authorized_keys`, `.conda/*`).
- Be careful about using **tab completion** or **reverse history search** if these haven't been introduced yet.

Desktop environment and browser

- Try to remove window title bars if they take up lots of space without adding value to the learner.
- Can you easily resize your windows for adjusting during teaching?
- Does your web browser have a way to reduce its menu bars and other decoration size?
  - Firefox-based browsers: go to `about:config` and set `layout.css.devPixelsPerPx` to a value slightly smaller than one, like `0.75`. Be careful you don't set it too small or large since it might be hard to recover! When you set it to something smaller than 1, all window decorations become smaller, and you compensate by zooming in on the website more (you can set the default zoom to be greater than 100% to compensate). Overall, you get more information and less distraction.

How to switch between teaching setup and work setup?

- Make a dedicated "demos" profile in your terminal emulator, if relevant. Or use a different terminal emulator just for demos.
- Same idea for the browser: Consider using a different browser profile for teaching/demos.
- Another idea is to containerize the setup for teaching. We might demonstrate this during the Sharing teaching gems session later.

Exercises

✍ Evaluate screen captures (20 min)

Evaluate screenshots on this page. Discuss the trade-offs of each one. Which one do you prefer? Which are useful in each situation?

- Share **portrait layout** instead of sharing entire screen
- **Adjust your prompt** to make commands easy to read
- **Readability** and beauty is important: adjust your setup for your audience
- **Share the history** of your commands

Please take notes in the collaborative document.

✍ Set up your own environment (20 min)

Set up your screen to teach something. Get some feedback from another learner or your exercise group.

- Get set up a few days in advance and get feedback from someone else. Feedback and time to improve is very important to make things clear and accessible. 10 minutes before the session starts is typically too late.

Exercises

the Sharing teaching gems session later.

✍ Evaluate screen captures (20 min)

Evaluate screenshots on this page. Discuss the trade-offs of each one. Which one do you prefer? Which are useful in each situation?

- Share **portrait layout** instead of sharing entire screen
- **Adjust your prompt** to make commands easy to read
- **Readability** and beauty is important: adjust your setup for your audience
- **Share the history** of your commands

Please take notes in the collaborative document.

✍ Set up your own environment (20 min)

Set up your screen to teach something. Get some feedback from another learner or your exercise group.

Other resources

- https://coderefinery.github.io/manuals/instructor-tech-setup/
- https://coderefinery.github.io/manuals/instructor-tech-online/

Computational thinking

Objectives

- Explain what is computational thinking
- Get to know how the theory of computational thinking can be used in teaching
- Short exercise

Instructor note

- Teaching: 20 min
- Exercises: 10 min

Materials available as slides.

Keypoints

- Computational Thinking consists of 4 main parts: decomposition, pattern recognition, abstraction and algorithmic design.
- How can this be a useful framework for solving problems?
- How can this be used practically?

CodeRefinery teaching philosophies

Objectives

✍ Ice-breaker in groups (20 minutes)

- Share your approach to teaching and your teaching philosophy with your group.
- Please share your tricks and solutions in the live document for others.

Additional ice-breaker questions:

- What is your motivation for taking this training?
- How structured or informal are your own teaching needs?
- What difference do you notice between the teaching what we (also Carpentries) do and traditional academic teaching?
- What other skills need to be taught, but academic teaching isn't the right setting?

Introduction

During this episode we split into breakoutrooms and discuss own teaching philosophies. Collect your teaching philosophies in the collaborative document. We will be going through these and the end of the lesson.

Instructor views

✍ Ice-breaker in groups (20 minutes)

- Share your approach to teaching and your teaching philosophy with your group.
- Please share your tricks and solutions in the live document for others.

Additional ice-breaker questions:

- What is your motivation for taking this training?
- How structured or informal are your own teaching needs?
- What difference do you notice between the teaching what we (also Carpentries) do and traditional academic teaching?
- What other skills need to be taught, but academic teaching isn't the right setting?

Introduction

During this episode we split into breakoutrooms and discuss own teaching philosophies.
Collect your teaching philosophies in the collaborative document. We will be going through these and the end of the lesson.

Instructor views

Here CodeRefinery instructors share their training philosophy to show that we all have different teaching styles and how these differences are beneficial to CodeRefinery workshops.

It is important to explain how much we value individuals and that there is not one way to teach but as many ways as individuals. We want to help each other to find the way that is best for each of us.

⚙ Video recordings

We have recorded some of the below as videos: https://www.youtube.com/playlist?list=PLpLblYHCzJAAHF89P-GCjEXWC8CF-7nhX

✍ Bjørn Lindi

My teaching style has changed a bit since I started with CodeRefinery. In the beginning I had this "BLOB" (Binary Large OBject) of knowledge and experience that I wanted to to convey to the participants. With experience and some help from the Carpentries Instructor training, I have realized I need to serialize the "BLOB", to be able to share it with others.

In a similar fashion as you would do with a binary large object which you intend to send over the wire, you will need stop signals, check-sums and re-transmissions, when you give a lecture. I have come to appreciate the natural periods/breaks the lessons offers, the questions raised, the errors that appear during type-along and the re-transmission. Co-instructors are good to use for re-transmission or broadening a specific topic.

When I started with CodeRefinery my inclination was to give a lecture. Today I am trying to be a guide during a learning experience, a learning experience which includes me as well. That may sound a bit self-centric, but is in fact the opposite, as I have to be more sensitive to what is going on in the room. The more conscious I am of being a guide, the better lesson.

Tools that I find useful in preparing a lesson is concept maps and Learner Personas (though I have developed too few of them):

- Concept Maps
- Learner Personas

✍ Radovan Bast

My teaching changed by 180 degrees after taking the Carpentries instructor training.
Before that I used slides, 45-minute lecture blocks, and separate exercise sessions. After the Carpentries instructor training I embraced the interaction, exercises, demos, and

better lesson.

My goal for a lesson is to spark curiosity to try things after the lesson, both for the novices (“This looks like a useful tool, I want to try using it after the workshop.”) and the more experienced participants (“Aha - I did not know you could do this. I wonder whether I can make it work with X.”). I like to start lessons with a question because this makes participants look up from their browsers.

Keeping both the novices and the experts engaged during a lesson can be difficult and offering additional exercises seems to be a good compromise.

For me it is a good sign if there are many questions. I like to encourage questions by asking questions to the participants. But I also try not to go into a rabbit hole when I get a question where only experts will appreciate the answer.

I try to avoid jargon and “war stories” from the professional developers’ perspective or the business world. Most researchers may not relate to them. For examples I always try to use the research context. Avoid “customer”, “production”, also a lot of Agile jargon is hard to relate to.

Less and clear is better than more and unclear. Simple examples are better than complicated examples. Almost never I have felt or got the feedback that something was too simple. I am repeating in my head to not use the words “simply”, “just”, “easy”. If participants take home one or two points from a lesson, that’s for me success.

I prepare for the lesson by reading the instructor guide and all issues and open pull requests. I might not be able to solve issues, but I don’t want to be surprised by known issues. I learn the material to a point where I know precisely what comes next and am never surprised by the next episode or slide. This allows me to skip and distill the essence and not read bullet point by bullet point.

I try to never deviate from the script and if I do, be very explicit about it.

A great exercise I can recommend is to watch a tutorial on a new programming language/tool you have never used. It can feel very overwhelming and fast to get all these new concepts and tools thrown at self. This can prepare me for how a participant might feel listening to me.

I very much like the co-teaching approach where the other person helps detecting when I go too fast or become too confusing. I like when two instructors complement each other during a lesson.

✍ Sabry Razick

My approach is to show it is fun to demystify concepts. Once a concept is not a mystery anymore, the learners will understand is what it means, where it is coming from, why it is in place and what it could it offer for their future. I try to relate concepts to real life with a twist of humour whenever possible if the outcome is certain not be offensive to any one. I use diagrams whenever possible. I have spent weeks creating diagrams that is sometime three or four sentences. That effort I consider worthwhile as my intention is not to teach, but to demystify. Once that is achieved, learners will learn the nitty gritty on their own easily and with confidence, when they have the use-case.

When teaching, I like lots of audience questions and don’t mind going off-script a bit (even though I know it should be minimized). I find that sufficient audience interest allows any lesson to be a success - you don’t have to know everything perfectly, just show how you’d approach a problem.

Like many people, I’ve often been teaching, but rarely a teacher. I tend to teach like I am doing an informal mentorship. I’ve realized long ago that my most important lessons weren’t learned in classes, but by a combination of seeing things done by friends and

✍ Stepas Toliautas

My main job is supporting computing infrastructure, so my teaching is very grounded in real-world problems. I often start at the very basics, because this is what I see missing most often.

✍ Richard Darst

Tools that I find useful in preparing a lesson is concept maps and Learner Personas (though I have developed too few of them).

- Concept Maps
- Learner Personas

✍ Radovan Bast

My teaching changed by 180 degrees after taking the Carpentries instructor training. Before that I used slides, 45 minute lecture blocks, and separate exercise sessions. After the Carpentries instructor training I embraced the interaction, exercises, demos, and independent study after that. I’ve realized that teaching (the things I teach) is trying to correct these differences in backgrounds.

anymore, the learners will understand is what it means, where it is coming from, why it is in place and what it could it offer for their future. I try to relate concepts to real life with a twist of humour whenever possible if the outcome is certain not be offensive to any one. I use diagrams whenever possible. I have spent weeks creating diagrams that is sometimes three or four sentences. That effort I consider worthwhile as my intention is not to teach, but to demystify. Once that is achieved, learners will learn the nitty gritty on their own easily and with confidence, when they have the use-case.

When teaching, I like lots of audience questions and don't mind going off-script a bit (even though I know it should be minimized). I find that sufficient audience interest allows any lesson to be a success - you don't have to know everything perfectly, just show how you'd approach a problem.

✍ Richard Darst

Like many people, I've often been teaching, but rarely a teacher. I tend to teach like I am doing an informal mentorship. I've realized long ago that my most important lessons weren't learned in classes, but by a combination of seeing things done by friends and independent study after that. I've realized that teaching (the things I teach) is trying to correct these differences in backgrounds.

My main job is supporting computing infrastructure, so my teaching is very grounded in real-world problems. I'm often start at the very basics, because this is what I see missing most often.

✍ Stepas Toliautas

I aim for my learners to understand things (concepts, techniques...), instead of just memorizing them. The phrase I use from time to time when teaching information technology topics (be it hardware or software) is "there is no magic": everything can be explained if necessary. While CodeRefinery also emphasizes usefulness/ immediate applicability of the content, often having the correct notions actually helps to start using them sooner and with less mistakes.

I try to guide my audience through the presented material with occasional questions related to previous topics and common knowledge – to help them link the concepts already during the lesson. I'm also fully aware that waiting for someone to answer is quite uncomfortable in-person and doubly so in an online environment, especially if no one uses their cameras :)

And if I get the question I don't have the answer for (or material to show it) at the moment, in university classes I prepare and come back to it next time. While with one-off workshops there might not be a "next time", open-source material used by CodeRefinery allows to have the same outcome: the latest and greatest version stays available for self-study afterwards.

Summary

Keypoints

- People have different viewpoints on teaching.

Co-teaching

Objectives

- Get to know the principle of co-teaching: How we do it and how you can too.
- Learn the team teaching concept and how to tailor it to your situation.

Instructor note

- Teaching: 15 min
- Exercises: 10 min
- Discussion: 5 min

Co-teaching can be used in various forms, some of which are present in our workshops:

Overview

- **Teaching + support**, e.g. one of the teachers leading instruction while the other watches over and maintains the collaborative document (Hack MD / Etherpad / ...).
- Another similar example is **remote learning groups** that watch the streamed CodeRefinery lessons guided by the local instructors.
- Having open-source material and planning jointly allows **multiple instances** of a lesson to be held by multiple teachers:

CodeRefinery lessons benefit from this application of **Co-teaching**.

Co-teaching

Co-teaching can be defined as “the division of labor between educators to plan, organize, instruct and make assessments on the same group of students, generally in a common classroom, and often with a strong focus on those teaching as a team complementing one another’s particular skills or other strengths”.

Co-teaching can be used in various forms, some of which are present in our workshops:

Overview

- **Teaching + support**, e.g. one of the teachers leading instruction while the other watches over and maintains the collaborative application (Hackmd/etherpad etc.).
- Another similar example is **remote learning groups** that watch the streamed CodeRefinery lessons guided by the local instructors.
- Having open-source material and planning jointly allows **multiple instances** of a lesson to be held by multiple teachers:
  - *parallel teaching*, to different audiences at the same time,
  - *alternative teaching*, to different audiences at the same or different time, with potential content adjustments (for example, different installation procedures).
- **Team teaching**, where the lesson is presented by multiple (in most cases, two) teachers who actively engage in conversation with each other. The team-teaching concept is explained in more detail in the CodeRefinery manual.

In reality, different forms are very often mixed or fused together, even within a single lesson.

Co-teaching is not an online-only concept. However, it is very practical in online teaching due to larger number of instructors and learners potentially available to join a teaching session.

Co-teaching and team teaching benefits

- It **saves preparation time**. Co-teachers can rely on each other’s strengths while creating/ revising the material as well as in unexpected situations during the lesson.
- It **helps with onboarding new instructors**. One of the co-teachers can be learning at the same time, either the subtleties of the material taught (in this case literally being the “voice of the audience”) or the teaching process itself.
- Team teaching **looks more interactive and engaging** to the audience in many cases, without forcing the learners to speak up if they can’t or don’t want to do so.
- It also **ensures responsive feedback and less workload** by having more active minds.

Are there any downsides?

Not every learner and not every instructor might like the team-teaching approach.

- It might seem **less structured**, unprepared, and chaotic, even with preparation.
  - It might create situations where instructors accidentally talk over each other or “interrupt” and change the flow of the lesson.
  - For some instructors it can be stressful to not know in advance what questions they get asked from the co-instructor.
  - Sometimes when an unexpected question is asked that throws the other instructor off, it can add to the feeling of chaos and unpreparedness.
- It can be interactive and engaging but it can also end up awkward if the co-teachers don’t have a good synergy.
  - Can sound awkward: Main instructor talking all the time and at the end asking co-instructor whether everything is clear and co-instructor only saying “yes”.
  - Possibly more engaging: Co-instructor asking questions which help with the flow and a common understanding of the material.
- Experienced solo teacher might have a habit to keep talking (lecturing), while the co-teacher might not want to “interrupt”. Therefore, it is important for the leading presenter to anticipate and **allow for remarks/ questions**, and this can be different from one’s previous teaching style at first.

Team teaching specifics

For successful team teaching, additional **coordination** is needed, first of all to agree on the teaching model (see below) and the person in control (the **director**) for the lesson or its parts.

- It’s useful to keep track of the **lecture plan**. The discussion is a good way to make lesson more interactive and adjust to the audience, but deviating too much will become disorienting (for example, if someone dropped their attention for a minute and now is trying to catch-up by reading lecture notes).

Team teaching specifics

- Sometimes when an unexpected question is asked that throws the other instructor off, it can add to the feeling of chaos and unpreparedness.
- It can be interactive and engaging but it can also end up awkward if the co-teachers don't have a good synergy.
  - Can sound awkward: Main instructor talking all the time and at the end asking co-instructor whether everything is clear and co-instructor only saying "yes".
  - Possibly more engaging: Co-instructor asking questions which help with the flow and a common understanding of the material.

For successful team teaching, additional **coordination** is needed, first of all to agree on the teaching model (see below) and the person in control (the **director**) for the lesson or its parts.

- It's useful to keep track of the **lecture plan**. The discussion is a good way to make lesson more interactive and adjust to the audience, but deviating too much will become disorienting (for example if someone dropped their attention for a minute and now is trying to catch-up by reading lecture notes).
- Experienced solo teacher might have a habit to keep talking (lecturing), while the co-teacher might not want to "interrupt". Therefore, it is important for the leading presenter to anticipate and **allow for remarks/ questions**, and this can be different from one's previous teaching style at first.

Team teaching models

We propose two basic models, but of course there is a constant continuum.

Guide and demo-giver

One person serves the role of **guide**, explaining the big picture and context of the examples.

Another, the **demo-giver**,

- shows the typing and does the examples,
- might take the role of a learner who is asking about what is going on, to actually explain the details, or to comment occasionally.

Hands-on demos and exercises work especially well like this.

Presenter and interviewer

In this case, one is the **presenter** who is mostly explaining (including demos or examples), and trying to move the forward through the material.

Another, the **interviewer**,

- serves as a learner or spotter,
- fills in gaps by asking relevant questions,
- tries to comment to the presenter when things are going off track.

This can be seen as closer to classical teaching, but with a dedicated and prepared "voice of the audience".

Exercise

Summary

✍ Discuss the models of team teaching (10 min)

❗ Keypoints

While in breakout rooms, discuss one of the basic team-teaching models presented here:

- Co-teaching focuses on complementing individual skills and strengths in teaching process.
- Co-teaching may save time, reduce teachers' workload and make lessons more interactive/ engaging.
- Team teaching requires some adjustments in lesson preparation and delivery.

Have you already tried this or similar model in your teaching?
Does it seem natural to apply this model in your subject area (tell what it is)? How could it be adapted to fit best?
Write your comments in the collaborative document.

Sharing teaching gems

❗ Objectives

Summary

✍ Discuss the models of team teaching (10 min)

Keypoints

While in breakout rooms, discuss one of the basic team-teaching models presented here:

- Co-teaching focuses on complementing individual skills and strengths in teaching process.
- Co-teaching may save time, reduce teachers' workload and make lessons more interactive/engaging.
- Team teaching requires some adjustments in lesson preparation and delivery.

Write your comments in the collaborative document.

Sharing teaching gems

Objectives

- Our goal is to share our teaching tricks and tools and demonstrate them in very short presentations/discussions.

Instructor note

- Demonstrations: 35 min

Here we encourage participants to take the screen share for 3 minutes and demonstrate either some tool or trick they found useful for their teaching or something from a course you liked.

Session 4 intro

This session covers streaming and technical production. Some introductory notes:

- (As of 2024) This is the first and only comprehensive introduction to our online streaming.
- These practices are new and well-refined internally, but a *different* kind of refinement is needed to teach and reuse them.
- The lessons have outlines of what to talk about, but it's just an outline. It is *not* refined, since thees things are *new*. Many things will have to be figured out as we talk.
- This session is a demo of lots of basics.
  - Ask questions - otherwise it will be boring
  - If you want to use this in real life: you will need *mentoring sessions* and active help. Contact us to do that.

Warning

Audio and video weirdness

We are setting up video and audio recording, but also capturing that for re-sharing it. There may be anomalies as video or audio feedback. Please be prepared.

- Exercises: 0 min

Why we stream

Objectives

This is a general discussion of the topics of the day, focused on the history of why we stream and record, how we got here, and how people feel about it. We won't focus on how anything is done.

- Learn the general history of CodeRefinery streaming.
- Discuss the benefits of streaming and recording
- Discuss the downsides and difficulties

Icebreaker questions

Instructor note

- What is the most people you have taught for?
- What are the distinct parts of teaching, that can be separated? (teaching, helping, etc)
- Discussion: 10 min
- Q&A: 5 min

What is streaming and recording?

- Exercises: 0 min

Why we stream

Objectives

- Learn the general history of CodeRefinery streaming.
- Discuss the benefits of streaming and recording
- Discuss the downsides and difficulties

This is a general discussion of the topics of the day, focused on the history of why we stream and record, how we got here, and how people feel about it. We won't focus on how anything is done.

Icebreaker questions

Instructor note

- What is the most people you have taught for?
- What are the distinct parts of teaching, that can be separated? (teaching, helping, etc)
  - Discussion: 10 min
  - Q&A: 5 min

What is streaming and recording?

- Streaming is mass communication: one to many
  - Interaction audience→presenters is more limited (but different)
- Using consumer-grade tools, normal people can reach huge audiences by Twitch/YouTube/etc.
- This isn't actually that hard: people with much less training than us do it all the time.
- They reach huge audiences and maintain engagement for long events.

Recording and rapid video editing is useful even without streaming.

History

- In-person workshops
  - 3 × full day, required travel, infrequent, one-shot
- Covid and remote teaching
  - Traditional "Zoom" teaching several times
- Mega-CodeRefinery workshop
  - 100-person Zoom teaching
  - Emphasis on teams
- Research Software Hour
  - Livestream free-form discussions on Twitch
- Streamed "HPC Kickstart" courses

Benefits and disadvantages

Benefits:

- Larger size gives more (but different) interaction possibility
  - "Notes" for async Q&A
- Recording (with no privacy risk) allows instant reviews
- Stream-scale allows for many of the things you have learned about in days 1-3.

Disadvantages:

- Requires training for using the tools
- Requires a certain scale to be worth it
- Coordination is much harder for big events

Future prospects (briefly)

- Streaming probably stays as a CodeRefinery tool
- We *can* scale our courses much larger than they are now. Why don't we, together with others?
- These tools are useful in other places too
  - I've used them to record my own talks to make videos of my single-person presentations or record conference talks nicer than Zoom.

When would I recommend it?

- No: For small courses
- Questionable: Medium sized courses with no videos
- Yes: When you want the largest audience
- Yes: When you want without registration required
- Yes: When you want good reusability/fast videos

Q&A

Q&A from audience

Disadvantages:

Future prospects (briefly)

- Requires training for using the tools
- Streaming probably stays as a CodeRefinery tool
- Requires a certain scale to be worth it
- We *can* scale our courses much larger than they are now. Why don't we, together with others?
- Coordination is much harder for big events
- These tools are useful in other places too
  - I've used them to record my own talks to make videos of my single-person presentations or record conference talks nicer than Zoom.

When would I recommend it?

- No: For small courses
- Questionable: Medium sized courses with no videos
- Yes: When you want the largest audience
- Yes: When you want without registration required
- Yes: When you want good reusability/fast videos

Q&A

Q&A from audience

Keypoints

- Streaming optimizes a course for different things
- The disadvantages can be compensated for
- There are benefits and disadvantages

Behind the stream

Objectives

- Take a first look at the broadcaster's view.
- Get to know what happens "behind the stream" of a workshop
- See what the "broadcaster" sees and what they need to do.
- Not yet: learn details of how to do this.

Instructor note

- Teaching: 20 min
- Q&A 10 min

In this episode, you'll see an end-to-end view of streaming from the broadcaster's point of view. It's a tour but not an explanation or tutorial.

Who does what

We have certain role definitions:

- **Broadcaster**: Our term for the person who manages the streaming.
- **Director**: Person who is guiding the instructors to their sessions, changing the scenes, calling the breaks, etc.
  - Could be the same as broadcaster.
- **Instructor**: One who is teaching. They don't have to know anything else about how streaming works.

CodeRefinery control panel

This lesson describes what the Broadcaster/Director sees.

- A custom application that controls scenes
- Based on OBS-websocket (remote control connection for OBS - we'll learn about this later)
- Can also work remotely, so that you can have a remote director

Window layouts

What does the broadcaster see on their screen?

- What are the main windows you see?
- What do each of them do?

How scenes are controlled

What has to be done during a course?

- Which ones do you need to focus on?
- How do you keep all this straight in your head?
- How do you start the stream?
- How do you change the view?

streaming works

CodeRefinery control panel

This lesson describes what the Broadcaster/Director sees.

- A custom application that controls scenes
- Based on OBS-websocket (remote control connection for OBS - we'll learn about this later)
- Can also work remotely, so that you can have a remote director

Window layouts

What does the broadcaster see on their screen?

- What are the main windows you see?
- What do each of them do?
- Which ones do you need to focus on?
- How do you keep all this straight in your head?

How scenes are controlled

What has to be done during a course?

- How do you start the stream?
- How do you change the view?
- How do you adjust things based on what the instructors share?
- How do you coordinate with the instructors?
- How do you know when to change the view?

Getting it set up

- How hard was it to figure this out?
- How hard is it to set it up for each new workshop?

What can go wrong

- What's the worst that has happened?
- What if you need to walk away for a bit?
- Someone broadcasts something unexpectedly

Alternatives

- Youtube vs Twitch
- Zoom stream directly to YouTube/Twitch
- Direct streaming platform, e.g. streamyard

Q&A

Q&A from audience

Keypoints

- The broadcaster's view shouldn't be so scary.
- There is a lot to manage, but each individual part isn't that hard.

Video editing

Objectives

- Get to know ways of quick video editing to be able to provide accessible videos
- Learn how video editing can be distributed and done the same day.

Instructor note

- Teaching: 20 min
- Exercises: 20 min

Primary articles

- Video editor role description: https://coderefinery.github.io/manuals/video-editor/
- ffmpeg-editlist: the primary tool: https://github.com/coderefinery/ffmpeg-editlist
- Example YAML edit list: https://github.com/AaltoSciComp/video-editing

How this relates to streaming

Video recordings should be made available within a few hours of the workshop. CodeRefinery does this, and you can too.

Hypothesis: videos must be processed the same evening as they were recorded, otherwise (it may never happen) or (it's too late to be useful). To do that, we have to make processing *good enough* (not perfect) and *fast* and the work *distributeable*.

Examples will be useful later, but also are (perhaps more) useful for **immediate review and catching up after missing a day in a workshop**. For this, they need to be released immediately.

Objectives

- Get to know ways of quick video editing to be able to provide accessible videos
- Learn how video editing can be distributed and done the same day.

Instructor note

- Teaching: 20 min
- Exercises: 20 min

Primary articles

- Video editor role description: https://coderefinery.github.io/manuals/video-editor/
- ffmpeg-editlist: the primary tool: https://github.com/coderefinery/ffmpeg-editlist
- Example YAML edit list: https://github.com/AaltoSciComp/course-editlists

Video recordings are useful for watching later, but also are (perhaps more useful for **immediate review and catching up after missing a day in a workshop**. For this, they need to be released immediately, within a few hours of the workshop. CodeRefinery does this, and you can too.

Hypothesis: videos must be processed the same evening as they were recorded, otherwise (it may never happen) or (it's too late to be useful). To do that, we have to make processing *good enough* (not perfect) and *fast* and the work *distributeable*.

How this relates to streaming

- If you stream, then the audience *can not* appear in the recorded videos
- This allows you to release videos *very quickly* if you have the right tools.
- When you have a large audience, the videos start helping more (review a missed day, catch up later, review later)
- Thus
  - If you would never want videos, there may never be a benefit to streaming
  - If you want videos, it gives motivation to stream.

Summary

- Basic principle: privacy is more important than any other factor. If we can't guarantee privacy, we can't release videos at all.
  - Disclaimers such as "if you don't want to appear in a recording, leave your video off and don't say anything", since a) accidents happen especially when coming back from breakout rooms. b) it creates an incentive to not interact or participate in the course.
- Livestreaming is important here: by separating the instruction from the audience audio/video, there is no privacy risk in the raw recording. They could be released or shared unprocessed.
- Our overall priorities
  1. No learner (or anyone not staff) video, audio, names, etc. are present in the recordings.
  2. Good descriptions.
  3. Removing breaks and other dead time.
  4. Splitting videos into useful chunks (e.g. per-episode), perhaps equal with the next one:
  5. Good Table of Contents information so learners can jump to the right spots (this also helps with "good description".)
- ffmpeg-editlist allows us to define an edit in a text file (crowdsourceable on Github), and then generate videos very quickly.

How we do it

The full explanation is in the form of the exercises below. As a summary:

✍ Editing-1: Get your sample video

- Record raw video (if from a stream, audience can't possibly be in it)
- Run Whisper to get good-enough subtitles. Distribute to someone for checking and improving.
- Define the editing steps (which segments become which videos and their descriptions) in a YAML file.
- Run ffmpeg-editlist, which takes combines the previous three steps into final videos.

Exercises

Exercise A

These exercises will take you through the whole sequence.

✍ Editing-2: Run Whisper to generate raw subtitles and test video.

First off, install Whisper and generate the base subtitles, based on the. Since this is probably too much to expect for a short lesson, they are provided for you (above), but

Download a sample video:

- Video (raw): http://users.aalto.fi/~darstr1/sample-video/ffmpeg-editlist-demo-kickstart-2023.mkv
- Whisper subtitles (of raw video): http://users.aalto.fi/~darstr1/sample-video/ffmpeg-editlist-demo-kickstart-2023.srt
- Schedule of workshop (day 1, 11:35–12:25) - used for making the descriptions. :::::

✍️ Editing-1: Get your sample video

- Record raw video (if from a stream, audience can't possibly be in it)
- Run Whisper to get good-enough subtitles. Distribute to someone for checking and improving.
- Define the editing steps (which segments become which videos and their descriptions) in a YAML file.
- Run ffmpeg-editlist, which takes combines the previous three steps into final videos.

Download a sample video:

- Video (raw): http://users.aalto.fi/~darstr1/sample-video/ffmpeg-editlist-demo-kickstart-2023.mkv
- Whisper subtitles (of raw video): http://users.aalto.fi/~darstr1/sample-video/ffmpeg-editlist-demo-kickstart-2023.srt
- Schedule of workshop (day 1, 11:35–12:25) - used for making the descriptions. :::::

Exercises

Exercise A

These exercises will take you through the whole sequence, probably too much to expect for a short lesson, they are provided for you (above), but if you want you can try using Whisper, or generating the subtitles some other way.

✍️ Editing-2: Run Whisper to generate raw subtitles and test video.

First off, install Whisper and generate the base subtitles. Since this is

You can start generating subtitles now, while you do the next steps, so that they are ready by the time you are ready to apply the editlist. ffmpeg-editlist can also slice up the subtitles from the main video to make subtitles for each segment you cut out.

Whisper is left as an exercise to the reader.

✔ Solution

Example Whisper command:

```
$ whisper --device cuda --output_format srt --initial_prompt="Welcome to
CodeRefinery day four." --lang en --condition_on_previous_text False
INPUT.mkv
```

An initial prompt like this make Whisper more likely to output full sentences, instead of a stream of words with no punctuation.

✍️ Editing-3: Create the basic editlist.yaml file

Install ffmpeg-editlist and try to follow its instructions, to create an edit with these features:

- The input definition.
- Two output sections: the "Intro to the course", and "From data storage to your science" talks (Remember it said the recording started at 11:35... look at the schedule for hints on when it might start!). This should have a start/end timestamp from the *original* video.

A basic example:

```
- input: day1-raw.mkv

# This is the output from one section.  Your result should have two of these
sections.
  output: part1.mkv
  - input: day1-obs.mkv

  - output: day1-intro.mkv
    title: 1.2 Introduction
    description: >-
      General introduction to the workshop.

      https://scicomp.aalto.fi/training/kickstart/intro/

    editlist:
```

A basic example:

```
- input: day1-raw.mkv

# This is the output from one section.  Your result should have two of these
sections.
  output: part1.mkv
  - input: day1-obs.mkv

  - output: day1-intro.mkv
    title: 1.2 Introduction
    description: >-
      General introduction to the workshop.

      https://scicomp.aalto.fi/training/kickstart/intro/

    editlist:
    - start: 00:24:10
    - end: 00:37:31


- output: day1-from-data-storage-to-your-science.mkv
  title: "1.3 From data storage to your science"
  description: >-
    Data is how most computational work starts, whether it is
    externally collected, simulation code, or generated. And these
    days, you can work on data even remotely, and these workflows
    aren't obvious. We discuss how data storage choices lead to
    computational workflows.

    https://hackmd.io/@AaltoSciComp/SciCompIntro

  editlist:
  - start: 00:37:43
  - end: 00:50:05
```

💬 Discussion: what makes a video easy to edit?

- Clear speaking and have high audio quality.
- For subtitle generation: Separate sentences cleanly, otherwise it gets in a “stream of words” instead of “punctuated sentences” mode.
- Clearly screen-sharing the place you are at, including section name.
- Clear transitions, “OK, now let’s move on to the next lesson, LESSON-NAME. Going back to the main page, we see it here.”
- Clearly indicate where the transitions are
- Hover mouse cursor over the area you are currently talking about.
- Scroll screen when you move on to a new topic.
- Accurate course webpage and sticking to the schedule

All of these are also good for learners. By editing videos, you become an advocate for good teaching overall.

✍️ Editing-4: Run ffmpeg-editlist

Install ffmpeg-editlist: `pip install ffmpeg-editlist` (you may want to use a virtual environment, but these are very normal dependencies).

The `ffmpeg` command line tool must be available in your `PATH`.

Look at the `.info.txt` files that come out.

It can be run with (where `.` is the directory containing the input files):

```
$ ffmpeg-editlist editlist.yaml .
```

✍️ Editing-5: Add more features

- Just ffmpeg-editlist is quick and works, but the stream may be garbled for a few seconds (because it’s missing a key frame). A future exercise will go over fixing this. Basically, add the `--reencode` option, which re-encodes the video (this is **slow**). Don’t do it yet.

✍ Editing-4: Run ffmpeg-editlist

Install ffmpeg-editlist (`pip install ffmpeg-editlist`) (you may want to use a virtual environment). (because it's missing a key feature). A future exercise will go over fixing this. Basically, add the `--reencode` option, which re-encodes the video (this is **slow**). Don't do it yet.

The `ffmpeg` command line tool must be available in your `PATH`.

Look at the `.info.txt` files that come out.

It can be run with (where `.` is the directory containing the input files):

```
$ ffmpeg-editlist editlist.yaml .
```

✍ Editing-5: Add more features

- generated). Video chapter definitions are timestamps of the *original* video, that get translated to timestamps of the *output* video.

  ```
  - output: part1.mkv
    editlist:
    - start: 10:00
    - -: Introduction    #  <-- New, `-` means "at start time"
    - 10:45: Part 1      #  <-- New
    - 15:00: Part 2      #  <-- New
    - end: 20:00
  ```

  Look at the `.info.txt` files that come out now. What is new in it?

- Add in "workshop title", "workshop description", and see the `.info.txt` files that come out now. This is ready for copy-pasting into a YouTube description (first line is the title, rest is the description).

  Look at the new `.info.txt` files. What is new?

✔ Solution

- This course actually didn't have chapters for the first day sessions, but you can see chapters for day 2 here, for example.
- Example of the workshop description for this course
- Example info.txt file for the general introduction to the course. The part after the `-----` is the workshop description.

  ```
  1.2 Introduction - HPC/SciComp Kickstart summer 2023

  General introduction to the workshop.

  https://scicomp.aalto.fi/training/kickstart/intro/

  00:00 Begin introduction       <-- Invented for the exercise demo, not real
  03:25 Ways to attend           <-- Invented for the exercise demo, not real
  07:12 What if you get lost     <-- Invented for the exercise demo, not real

  -----

  This is part of the Aalto Scientific Computing "Getting
  started with Scientific Computing and HPC Kickstart" 2023
  workshop.  The videos are available to everyone, but may be
  most useful to the people who attended the workshop and want
  to review later.

  Playlist:
  https://www.youtube.com/playlist?list=PLZLVmS9rf3nMKR2jMglaN4su3ojWtWMVw

  Workshop webpage:
  https://scicomp.aalto.fi/training/scip/kickstart-2023/

  Aalto Scientific Computing: https://scicomp.aalto.fi/
  ```

✍ Ed

Re-ru
`ffmpe`
out n

Use s
mista
some other tool.

```
-----

This is part of the Aalto Scientific Computing "Getting
started with Scientific Computing and HPC Kickstart" 2023
workshop.  The videos are available to everyone, but may be
most useful to the people who attended the workshop and want
to review later.

Playlist:
https://www.youtube.com/playlist?list=PLZLVmS9rf3nMKR2jMglaN4su3ojWtWMVw

Workshop webpage:
https://scicomp.aalto.fi/training/scip/kickstart-2023/

Aalto Scientific Computing: https://scicomp.aalto.fi/
```

✍ Ed

Re-ru
`ffmpe`
out n

Use s
mista
some other tool.

What do you learn from editing the subtitles?

✔ Solution

```
$ ffmpeg-editlist --srt editlist.yaml
```

There should now be a `.srt` file also generated. It generated by finding the `.srt` of the original video, and cutting it the same way it cuts the video. Look and you see it aligns with the original.

This means that someone could have been working on fixing the Whisper subtitles while someone else was doing the yaml-editing.

✍ Editing-6: Subtitles

Re-run ffmpeg-editlist with the `--srt` option (you have to install it with `pip install ffmpeg-editlist[srt]` to pull in the necessary dependency). Notice how `.srt` files come out now.

Use some subtitle editor to edit the *original* subtitle file, to fix up any transcription mistakes you may find. You could edit directly, use `subtitle-editor` on Linux, or find some other tool.

What do you learn from editing the subtitles?

✍ Editing-7: Generate the final output file.

- Run ffmpeg-editlist with the `--reencode` option: this re-encodes the video and makes sure that there is no black point at the start.
- If you re-run with `--check`, it won't output a new video file, but it *will* re-output the `.info.txt` and `.srt` files. This is useful when you adjust descriptions or chapters.

This is a more limited (and older) version of the above exercise, using an synthetic example video.

✍ Use ffmpeg-editlist to edit this sample video

Prerequisites: `ffmpeg` must be installed on your computer outside of Python. Be able to install ffmpeg-editlist. This is simple in a Python virtual environment, but if not the only dependency is `PyYAML`.

How hard was this editing? Was it worth it?

- Download the sample video: http://users.aalto.fi/~darstr1/sample-video/sample-video-to-edit.raw.mkv
- Copy a sample editlist YAML
- Modify it to cut out the dead time at the beginning and the end.

✍ Discussion: how to distribute this?

Create a flowc
parallel! Don't forget things that you might need to do before the workshop starts.

Exercise B

- If you re-run with `--check`, it won't output a new video file, but it will re-output the `.info.txt` and `.srt` files. This is useful when you adjust descriptions or chapters.

This is a more limited (and older) version of the above exercise, using an synthetic example video.

Use ffmpeg-editlist to edit this sample video

Prerequisites: `ffmpeg` must be installed on your computer outside of Python. Be able to install ffmpeg-editlist. This is simple in a Python virtual environment, but if not the only dependency is `PyYAML`.

How hard was this editing? Was it worth it?

- Download the sample video: http://users.aalto.fi/~darstr1/sample-video/sample-video-to-edit.raw.mkv
- Copy a sample editlist YAML
- Modify it to cut out the dead time at the beginning and the end.
- If desired, add a description and table-of-contents to the video.
- Run ffmpeg-editlist to produce a processed video.

✔ Solution

```
- input: sample-video-to-edit.raw.mkv
- output: sample-video-to-edit.processed.mkv
  description: >
  editlist:
    - start: 00:16
    - 00:15: demonstration
    - 00:20: discussion
    - stop: 00:25
```

```
$ ffmpeg-editlist editlist.yaml video/ -o video/
```

Along with the processed video, we get `sample-video-to-edit.processed.mkv.info.txt`::

```
This is a sample video


00:00 Demonstration
00:04 Discussion
```

Discussion: How to distribute this?

Create a flowc[illegible] if the parts that need to be done, and which parts can be done in parallel? Don't forget things that you might need to do before the workshop starts.

Exercise B

- Understand the basic controls and features of OBS.

See also

- ffmpeg-editlist demo: https://www.youtube.com/watch?v=thvMNTBJg2Y
- Full demo of producing videos (everything in these exercises): https://www.youtube.com/watch?v=_CoBNe-n2Ak
- Example YAML editlists: https://github.com/AaltoSciComp/video-editlists-asc

Keypoints

Instructor note

- Teaching: 15 min
- Q&A: 5 min

Open Broadcaster Software (OBS) introduction

- OBS theory in CodeRefinery manuals
- The next episode: Open Broadcaster Software (OBS) setup

Objectives

In this episode, you'll get more familiar with the OBS portion of the streaming setup. You'll see how it's used, but not yet how to configure it from scratch. You'll learn how to be a "director".

- Understand that OBS is a video mixer.
- Video editing is very useful for learning
- Set your time budget and make it good enough in that time
- Reviewing videos improves your teaching, too.

- Example YAML editlists: https://github.com/AaltoSciComp/video-editlists-asc
- Understand the basic controls and features of OBS.

Keypoints

- Video editing is very useful for learning
- Set your time budget and make it good enough in that time
- Reviewing videos improves your teaching, too.

Instructor note

- Teaching: 15 min
- Q&A: 5 min

Open Broadcaster Software (OBS) introduction

- OBS theory in CodeRefinery manuals
- The next episode: Open Broadcaster Software (OBS) setup

Objectives

- Understand that OBS is a video mixer.

In this episode, you'll get more familiar with the OBS portion of the streaming setup. You'll see how it's used, but not yet how to configure it from scratch. You'll learn how to be a "director".

What is OBS?

- Formally "OBS Studio"
- Most commonly known as a livestreaming application.
- Open source, free.
- Cross-platform, easy to use screencasting and streaming application.
- Real-time video mixer.

OBS user interface

- We'll click through each view.
- What does each view do?
- Let's click through the buttons.
- Let's see the important config options.

OBS during a course

- What management is needed.
- The control panel.
- Audio.
- Adjusting windows and so on.

Hardware requirements

- Reasonably powerful broadcast computer
  - CodeRefinerys streaming computer is 8 CPU AMD, 64GB memory
  - That is way overkill: a powerful laptop can probably do this.
- Large second monitor for laying out the windows you capture
- Stable internet connection (wired preferable)
  - CodeRefinery's broadcast is the slowest purchasable: 100 Mbit down / 25Mbit up

Q&A

- OBS may seem complicated, but it's a graphical application and most pieces make sense once you know how it works.

Questions and answers

Open Broadcaster Software (OBS) setup

Objectives

- See how to configure OBS using the pre-made CodeRefinery scene collections.
- Modify the collections to suit your needs.

Setup

- The next episode: Open Broadcaster Software (OBS) setup which is about configuring OBS.

Instructor note

- Teaching: ?? min
- Hands-on: ?? min

- OBS may seem complicated, but it's a graphical application and most pieces make sense once you know how it works.

Q&A

We'll answer audience questions.

Open Broadcaster Software (OBS) setup

Objectives

- See how to configure OBS using the pre-made CodeRefinery scene collections.
- Modify the collections to suit your needs.

Keypoints

Instructor note

- Teaching: ?? min
- Hands-on: ?? min
- Q&A: ?? min

- The previous episode Open Broadcaster Software (OBS) introduction
- OBS theory in CodeRefinery manuals

In this lesson, we'll see how to configure OBS from scratch for your purposes. We'll do this by deleting the instructor's configuration and trying to recreate it

This section is short, since it has never been done before: we'll just give it a short and update the lesson later.

CodeRefinery OBS configs

- CodeRefinery configs are shared in a repository: https://github.com/coderefinery/obs-config
- These can be imported to pre-configure some things
- There are two types of configs:
  - Profiles
    - Servers, resolutions, audio, video, etc.
  - Scene collections
    - The graphical layouts.

Installing the OBS config

- Clone the git repository.
- Import the profile and scene collections under their respective menus.

Initial setup

- Click through each menu and change anything that is needed
- Set the streaming server
- There are more options but let's not cover them yet... and leave this for a hands-on session.

The instructor will go through the setup.

Setup before each course

- Reset configuration: `mv .config/obs-studio/ .config/obs-studio-old/`
- Import `profiles/TeachingStreamingv3/`
- Import `scenes/TeachingStreamingZoomv3.json`
- Re-confirm audio
- Re-confirm each scene
- Test everything
- rkdarst has a rather long checklist, but each individual step is short.

Set up the remote control

- Create a Python environment and install it: `pip install https://github.com/coderefinery/obs-cr/archive/refs/heads/master.zip`
- Run it: `python obs_cr/control.py localhost:4445 TOKEN --broadcaster`

- There are more options but let's not cover them yet... and leave this for a hands-on session.

The instructor will go through the setup.

Setup before each course

- Reset configuration: `mv .config/obs-studio/ .config/obs-studio-old/`
- Import `profiles/TeachingStreamingv3/`
- Import `scenes/TeachingStreamingZoomv3.json`
- Re-confirm audio
- Re-confirm each scene
- Test everything
- rkdarst has a rather long checklist, but each individual step is short.

Set up the remote control

- Create a Python environment and install it: `pip install https://github.com/coderefinery/obs-cr/archive/refs/heads/master.zip`
- Run it: `python obs_cr/control.py localhost:4445 TOKEN --broadcaster`

Keypoints

- Most of our configuration has been OBS may seem complicated, but it's a graphical application and most pieces make sense once you know how it works.

What's next?

Objectives

- Know next steps if you want to do streaming

Instructor note

- Teaching: 5 min
- Q&A 5 min

Q&A

We'll answer audience questions.

What comes next?

- We talked a lot about theory, and gave demonstrations.
- Hands-on is very different. We recommend working with someone to put it in practice.
-
- Work with someone who can show you the way
- Use it for smaller courses with a backup plan

See also

Keypoints

- These lessons about streaming have been the theoretical part of streaming training.

Collaborative notes archives from workshops

August/September 2024

Day1: Session 1 (13.08.24) - About lesson design, deployment and iterative improvement

:calendar: Schedule

Time (CEST)	Title	EEST (UTC+3)	BST (UTC+1)
10.00 - 10.15	Break	11.00 - 11.15	9.00 - 9.15
10.15 - 11.00	Lessons with version control	11.15 - 12.00	9.15 - 10.00
11.00 - 11.15	Break	12.00 - 12.15	10.00 - 10.15
11.15 - 12.00	How we collect feedback and measure impact	12.15 - 13.00	10.15 - 11.00
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 10.00	Lesson design and development	10.15 - 11.00	8.15 - 9.00

:icecream: Icebreaker

Check-in

August/September 2024

Day1. Session 1 (13.08.24) - About lesson design, deployment and iterative improvement

:calendar: Schedule

Time (CEST)	Title	EEST (UTC+3)	BST (UTC+1)
10.00 - 10.15	Break	11.00 - 11.15	9.00 - 9.15
10.15 - 11.00	Lessons with version control	11.15 - 12.00	9.15 - 10.00
11.00 - 11.15	Break	12.00 - 12.15	10.00 - 10.15
11.15 - 12.00	How we collect feedback and measure impact	12.15 - 13.00	10.15 - 11.00
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 10.00	Lesson design and development	10.15 - 11.00	8.15 - 9.00

:icecream: Icebreaker

Check-in

Which emoji best describes your current state of mind? Emoji cheat sheet

- :Decaffed ... and going strong ...
- :coffee: 🤔
- :100: (ironically!) :coffee: intake happening
- :smile:
- :tired_face:
- :coffee: :party:
- :coffee: :happy:
- 😪
- just got zoom installed / before my coffee
- :smil-
- :sweat_smile: (+)
- :yawning_face:
- :is it morning already?
- :smile:
- :yawning_face:
- :nerded_face:
- 🥵
- :smile:
- :sleeping:
- :sleepy: :coffee: :smiley_cat:
- 🥳
- :coffee:
- :coffee:

Introduction in breakoutrooms.

- Name / Affiliation / Location

About teaching

What is the hardest thing about teaching for you?

- Preparation of the session material and estimating the right amount of time for each section
- Getting learners to take the first step: sign up for and attend a workshop, when they don't think programming is a skill they can learn
- Getting learners to take the second step: translating what they've learned in a workshop into something they can apply
- Managing groups with vastly different academic backgrounds
- Engaging with the students which was much easier for me when I used to coach
- Finding the right depth for an unknown audience for my topic
- General preparation time, how to fit it into the regular work schedule, and estimate how much time is needed for prep.
- How to reduce too much text into just the right amount

Knowing how "it's going", whether learners are happy or not (especially when teaching online) :+1:
The preparation the night before. It is always much more that I would hope, no matter how prepared I want it to be :+1: :+1:
preparation and time management
Teaching wide spectrum of skill level at the same time :+1: :+1: :+1: :+1:
Time management! (Knowing how much can fit in a sessions) :+1: :+1:
education background of participants and their learning objectives

What is the best thing about teaching for you?

What is the hardest thing about teaching for you?

- Preparation of the session material and estimating the right amount of time for each section
- Getting learners to take the first step: sign up for and attend a workshop, when they don't think programming is a skill they can learn
- Getting learners to take the *second* step: translating what they've learned in a workshop into something they can apply
- Managing groups with vastly different academic backgrounds
- Engaging with the students which was much easier for me when I used to coach
- Finding the right depth for an unknown audience for my topic
- General preparation time, how to fit it into the regular work schedule, and estimate how much time is needed for prep.
- How to reduce too much text into just the right amount

What is the best thing about teaching for you?

- Seeing when learner gets interested/curious about something and takes inspiration from the course. :+1:
- Seeing it works and someone can do something new.
- Motivated students grasping new stuff
- Students picking up and running with the matertial and skills I give them and using it for their own work. :+1:
- The "ah-ha!" moment when a student gets it! :+1::100: :+1:
- Learning from students who know about some topic more than you do. :+1:
- Being able to help people reach their goals, by showing them something new.
- Teaching is optimism acted out in the hope of making a difference both ways I suppose.
- The feeling of accomplishment when you see learning grow and implement the learnings :book:
- mutual learning and impact on learners :+1:
- Results and feedbacks / Mutual interests and interesting discussions
- mutual learning and I can also learn lots of things and new ideas from participants

:question: Questions

General / Practicalities

- Will the material be made available?
  - Materials are and will stay available at https://coderefinery.github.io/train-the-trainer/
    - Can I share them with a colleague
      - Yes please :)
    - Can I resuse them in my own teaching?
      - Yes please :)
- Will these sessions be recorded?
  - No, but the materials (see above) should include everything needed for self-learning.
- Can I get a certificate for this workshop?
  - You can get a certificate of attendance after the workshop by sending an e-mail to support@coderefinery.org, telling us a bit about what you have learned and which sessions you attended.
- How often will we be talking in breakout rooms? I'm not in a place where I can talk, so if we're going to talk a lot I'll need to move offices
  - We'll have about one breakoutroom session per episode
  -

Episode 1: Lesson design and development

Materials: https://coderefinery.github.io/train-the-trainer/lesson-development/ **Discussion:**

- When you start preparing a new lesson or training material, where do you start?
  - Start by looking for related resources (documents, videos, courses, etc) to have a base data and better understanding of what have been done around the topic
  - Know your audience
  - Review existing material
  - Think about learning objectives (to keep focus on essential things)
  - Think of three things simple enough that they will be remembered the next day. Design around that.
  - I write down the thoughts that I have and then go back and structure it.
- What tricks help you with "writer's block" or the empty page problem?
  - I look at existing materials on the same topic, but chose/refine topics based on the perceived needs/interests of the presumed audience
  - Write anything down that comes to mind, sometimes draw something, looking out the window :)
  - Outline of structure and Material collection for a new lesson :+1:
  - Do a mind map - what concepts do I want to get across?
  - What material I have already? (what other material is already out there?)
  - Starting small, for example a list of headings and then building around it. :+1:

We're going to talk a lot. I need to move offices
    - Start by looking for related resources (documents, videos, courses, etc) to have a base
    - We'll have about one breakoutroom session per episode
data and better understanding of what have been done around the topic
-
    - Know your audience
    - Review existing material
Episode 1: Lesson design and development
    - Think about learning objectives (to keep focus on essential things)
    - Think of three things simple enough that they will be remembered the next day.
Materials: https://coderefinery.github.io/train-the-trainer/lesson-development/ **Discussion:**
Design around that.
    - I write down the thoughts that I have and then go back and structure it.
- When you start preparing a new lesson or training material, where do you start?
- What tricks help you with "writer's block" or the empty page problem?
    - I look at existing materials on the same topic, but chose/refine topics based on the
    - Write anything down that comes to mind, sometimes draw something, looking out the
perceived needs/interests of the presumed audience
window :)
    - Outline of structure and Material collection for a new lesson :+1:
    - Do a mind map - what concepts do I want to get across?
    - What material I have already? (what other material is already out there?)
    - Starting small, for example a list of headings and then building around it. :+1:
    - What does one know about the target audience and why should they be spending longer than 1 minute listening to what they will eventually get to hear?
    - Get some inspiration from another source.
    - Start with the three things above
    - Start with the plan and the overview design
    - Some kind of outline / main message(s)
    - Start with three things. Three supporting points for each of these.
- Maybe you haven't designed training material yet. But how do you start when creating a new presentation?
    - Think about the learning objectives and try to break them down into steps
    - Title, Objectives, target audience and Plan
    - Some kind of outline / main message(s). Get as many images as I can, instead of words
    - Draft an outline and use chatgpt to fill in details
    - Example of how the teaching content is applied in a real-world context
- If your design process has changed over time, please describe what you used to do and what you do now instead.
    - I look more at existing materials and try to get more information about the audience. Unfortunately getting information about the audience before the event is hard
    - I used to start from the begginning and get from there but that often meant a very polished start and a rushed end. Now I try an overview first, then fil out sparsed details at every section
    - Updating the data and tweak the presentation
    - If I am teaching a small group (or one to one) talk to them before hand - find out what they know already, what they want to learn.
- What do you know now about preparing lessons/training/presentations that you wish you knew earlier?
    - less is more. It's better to have 2-3 main messages rather than trying to show everything in one go :+1: :+1: :+1: :+1:
    - how much practice time the learners need to master what's taught
    - Don't worry about something going wrong. It often makes the lesson (and thus the material) more presistent in memory. :+1:
    - Try and remove everything except what you want the person to learn
        - That's a very tough part for me in the sense that I never know how much of an
    - A defined example person described in terms of the materials you are preparing,
underlying "black box" is still ok...
making up a story about the background and interest of t
    - Designing intermediate materials is hard, and requires putting some "gatekeeping"
    - Answer from chat: A learner persona is a blackbox which has a very simple algorithm
making sure that learners are directed to appropriate courses
inside: GIGO.
    - When I see a cool graphic, concept, slide, etc., download it and save it in my 'new-
    - Answer from chat: I think it's the same as ICP (ideal customer persona) where you
materials' folder to use later on!
describe the learner as a customer
    - I have come to the conclusion that perhaps a more "agile" approach to developing
    - Answer from chat: I'm a big fan of "How Learning Works" by Ambrose et al
materials (try to do design/teaching iterations quickly) might be the best way to go,
https://bookshop.org/p/books/how-learning-works-eight-research-based-principles-
but there are risks with this approach too
for-smart-teaching-michele-dipietro/18640868?ean=9781119861690
    - See also book: Teaching tech together

Questions

CodeRefinery lessons: https://coderefinery.org/lessons/

- From zoom chat: Can you expand on what a learner persona is?

        - That's a very tough part for me in the sense that I never know how much of an underlying "black box" is still ok...
    - A defined example person described in terms of the materials you are preparing, making up a story about the background and interest of t
    - Designing intermediate materials is hard, and requires putting some "gatekeeping" making sure that learners are directed to appropriate courses
    - Answer from chat: A learner persona is a blackbox which has a very simple algorithm inside: GIGO.
    - When I see a cool graphic, concept, slide, etc., download it and save it in my 'new-materials' folder to use later on!
    - Answer from chat: I think it's the same as ICP (Ideal customer persona) where you describe the learner as a customer
    - I have come to the conclusion that perhaps a more "agile" approach to developing materials (try to do design/teaching iterations quickly) might be the best way to go, but there are risks with this approach too
    - Answer from chat: I'm a big fan of "How Learning Works" by Ambrose et al https://bookshop.org/p/books/how-learning-works-eight-research-based-principles-for-smart-teaching-michele-dipietro/18640868?ean=9781119861690
    - See also book: Teaching tech together

Questions

CodeRefinery lessons: https://coderefinery.org/lessons/

- From zoom chat: Can you expand on what a learner persona is?
- From chat: Do you have an example of an instructor guide?
    - One example from our reproducible research lesson
    - Another from git intro: https://coderefinery.github.io/git-intro/guide/
- From zoom chat: Do you have a "measure" of how better are the new lesson compared to the old ones? Does it make sense to talk about measurements here?
    - So far we only have daily feedback that we collect at the end of the day and we can compare that to earlier daily feedbacks. More about that in the last episode of today. But measuring this is indeed not easy and it also takes time to capture the effect during longer-term surveys. Some of the changes were more leap-of-faith than based on data.
    - I guess that sometimes one can use "compelling arguments" instead of data to justify a decision
- Software Carpentry does teach using the command line, and gives a session on the command line before. What do you think about this approach? Are you stating that a 3-hour lesson about CLI is not sufficient to make people comfortable enough to use that for version control?
    - Yes, good question. For CR we consider it a more advanced step: we do think people should learn command line, but it's for a different course. We accept some people might not know it or want to know it so adapt to that.
    - In general, we've found the "you need X, but have to learn A, B, and C first" approach should be avoided if possible: people are busy, try to reach people where they are.

Exercise in breakoutrooms

- Room 1
    - Research data management
    - Objectives
        - Research life cycle
        - FAIR principles (Findable, Accessible, Interoperable, Reusable)
    - Exercise
        - develop a data management plan (DMP) that outlines how their research data will be handled throughout its lifecycle
        - [What are and how to manage typical issues]
- Room 2: What is the difference between eating breakfast and sitting through a lecture? (Tip: Start by listing the similarities...)
- Room 2 : Making papers in LaTeX - What is LaTeX and how is it different from editors like Word - Basic Structures - How to find and use a template - Including figure & table - How to use references and labels - Exercise: to create a new LaTeX document & edit it - Exercise: common error messages: can you find the error in this code? - Exce
- Room 3
    - GPU Programming (1 hour intro)
    - Exercise
    - Learning objectives:
        - What is GPU programming.
        - When is it usable, beneficial to use GPU programming?
        - What are technologies to do GPU programming?
- Room 4
    - Git / Figures / Project management / data cleaning
    - Chose data cleaning
    - Three learning objectives:
        1. What do we mean by "clean"? How to identify it?
        2. Identify some common problems in a dataset
        3. Identifying and handling missing values/fields
    - Discuss problems, find the most common ones
        - In small groups
        - What problems have you run into with datasets you've worked with?
        - What problems can you imagine, or have you heard about from

be handled throughout its lifecycle
    - [What are and how to manage typical issues]
- Room 2: What is the difference between eating breakfast and sitting through a lecture?
- Room 4
  - Git / Figures / Project management / data cleaning
  - Chose data cleaning
  - Three learning objectives:
    1. What do we mean by "clean"? How to identify it?
    2. Identify some common problems in a dataset
    3. Identifying and handling missing values/fields
- Room2 : Making papers in LaTeX - What is LaTex and how is it different from editors like Word - Basic Structures - How to find and use a template - Including figure & table - How to use references and labels - Exercise: to create a new LaTeX document & edit it - Excercise: common error messages: can you find the error in this code? - Exce
- Room 3
  - GPU Programming (1 hour intro)
  - Exercise
  - Learning objectives:
    1. Discuss problems, find the most common ones
       - What is GPU programming.
       - In small groups
       - When is it usable, beneficial to use GPU programming?
       - What problems have you run into with datasets you've worked with?
       - What are technologies to do GPU programming?
       - What problems can you imagine, or have you heard about from colleagues/news/social media?
       - Report back, instructor collects all problems into a list
    2. Have a dataset to clean; a clean and a messy example
       - (identifying) Using a visualisation or overviewing tool?
- Room 5
  - Jupyter Notebooks
  - Learning Objectives
    - Can setup an enviroment you can reuse / can share with others / a project.
    - The order you run the commands in needs to match the order of the code in the Notebook - otherwise you end up in a pickle.
    - Why use a Jupyter Notebook? What are the advantages? Easy to use environment.
  - Exercise
    - Print variable assignments from different cells - show that the order you run the code is important.
    - hand out a domain specific notebook that does some calculations and visualization to run and modify, to show that you can share easily
    - show !pip install to add dependencies/features
- Room 6
  - Using GitHub without the command line
  - Learning Outcomes:
    - Be able to discuss changes before merging changes into the main repository.
    - Publish a personal repository page (I think its called intro repository)
    - Share their script/notebook/code/small data on GitHub
    - Homepage using GitHub pages or the README that becomes the "index page" of the GitHub user account page
  - Learner personas:
    - Check the Personas of the learners
      - Somebody who has seen/heard of GitHub but hasn't used it yet
      - Someone using it for their own work but struggling with collaboration
  - Exercises:
    - Upload/share an example dataset or script
    - Review another collaborators pull request.
    - Take part in the discussion on a pull request.

Materials: https://coderefinery.github.io/train-the-trainer/lessons-with-git/

- Room 7 Linux shell basics: why do we want to teach them? (this took most of the time for the discussion)
  - Learning objectives:
    1. Filesystem: Directory (CLI) - Folder (GUI) analogy
    2. Basic commands (which commands are basic? make a survey of shell histories and get the most common ones)
    3. Basic constructs (e.g., for loops, pipes, while loops... which ones are basic? See above)

Poll options: I want to hear about:

- A: Why use version control for teaching materials
- B: Different template options
- C: How CodeRefinery does it; CodeRefinerys lesson template

Poll vote (multi-select): add a `o` to your answer - A: ooo - B: oooooo - C: oooooooo :ghost:

Note from chat: Zoom timer: https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0068677

Question to audience: if someone wants to make a tiny fix to your material, how hard is it?

- If my material is only in pdf format which is not online, then it is hard.

Episode 2: Lessons with version control

Materials: https://coderefinery.github.io/train-the-trainer/lessons-with-git/

- Room 7 Linux shell basics: why do we want to teach them? (this took most of the time for the discussion)

Poll options: I want to hear about:

- A: Why use version control for teaching materials
- B: Different template options
- C: How CodeRefinery does it, CodeRefinerys lesson template

Poll vote (multi-select): add a `o` to your answer - A: ooo - B: oooooo - C: oooooooo :ghost:

Question to audience: if someone wants to make a tiny fix to your material, how hard is it?

Episode 2: Lessons with version control

- If my material is only in pdf format which is not online, then it is hard.
- I hope it is easy - my material is in GitHub. Nobody has ever done that though!
- Slides: hard to find, easy to edit; Git repos somewhat easy, unless they have to be built locally or with an action; HackMD easiest to edit.
  - if you use something like google slides they are not that difficult to find
  - known permanent address. Google docs et al fail on that/same for many pads.
- For this material today - very easy! e.g. https://github.com/coderefinery/train-the-trainer/pull/128
- Template with an edit button in the HTML pointing to the source-code in the repo.

Questions/Discussion topics

Upcoming CodeRefinery workshop in September: https://coderefinery.github.io/2024-09-10-workshop/ Please share with your networks, and let us know if you would like to co-teach a lesson (to support@coderefinery.org or in our Zulip chat)

- Changes to the material are sometimes not just tiny fixes. For example, one wants to present a slightly different selection of topics, but maybe some topics are “coupled” (perhaps naturally or perhaps just because of how they are presented), or change the general “theme” (e.g., life sciences vs theoretical physics) while keeping the actual topics the same. How could a lesson be developed to avoid having to do a huge rewrite?
  - You can try to modularize your material and keep non topic specific parts non themed. E.g. where you explain the functionality, don’t make reference to the example (not simple), so the example can be changed depending on the audience.
- What are the pros/cons of renaming a course / changing a repo name in GitHub?
  - It should be relatively unproblematic since GitHub will forward to the new name.
    - How long does Github keep the “old” name linked? Is there a max time it’s blocked, or changed as soon as a new one appears with the name?
      - in my experience it forwards “forever” until I create a new repo with the old name which will break the forward
- what should be in the readme for the git repo? (canonical address? contact points? license!)
  - https://coderefinery.github.io/mini-workshop/2/documentation/#often-a-readme-is-enough-checklist
    - I miss the url to the (main) repo in the readme
      - so realistically we need a content folder and some more next to the readme as a best practice. sounds good!
      - Do you mean the link to the rendered (lesson) page?
- Is myst (markdown renderer) enabled by default in sphinx now?
  - to my knowledge no. we add the “myst_parser” extension to conf.py.
  - Link to rendered page: We often have this in the “about” section of the readme (see up right: https://github.com/coderefinery/train-the-trainer)
  - this gets lost in a fork, doesn’t it? so 20 folks forking and or reuploading the repo somewhere means the url gets lost?
    - True, but they might want to have their own version rendered?
    - yes I have had issues finding and contributing to the “master” repo so all benefit. Else it gets cluttered. Both is valid
    - Good point though, the link to rendered page should probably also be included in the readme for completeness; ah and the link to the repository too, yes

Cicero: https://cicero.readthedocs.io/ Sphinx documentation: https://www.sphinx-doc.org/en/master/ Carpentries workbench for creating Carpentries lessons: https://carpentries.github.io/workbench/ This is the testing lesson Richard showed: https://coderefinery.github.io/testing/ And the corresponding repository: https://github.com/coderefinery/testing You can lean back and watch, exercise coming in a bit :) Richard will now edit our train the trainer material repository: https://github.com/coderefinery/train-the-trainer The Sphinx extension for CodeRefinery lessons: https://github.com/coderefinery/sphinx-lesson Empty lesson template to build

enough-checklist
    - so realistically we need a content folder and some more next to the readme as a best practice. sounds good!
  - I miss the url to the (main) repo in the readme
    - Do you mean the link to the rendered (lesson) page?
- Is myst (markdown renderer) enabled by default in sphinx now?
  - to my knowledge no. we add the "myst_parser" extension to conf.py.
  - Link to rendered page: We often have this in the "about" section of the readme (see up right: https://github.com/coderefinery/train-the-trainer)
    - this gets lost in a fork, doesn't it? so 20 folks forking and or reuploading the repo somewhere means the url gets lost?
      - True, but they might want to have their own version rendered?
      - yes I have had issues finding and contributing to the "master" repo so all benefit. Else it gets cluttered. Both is valid
      - Good point though, the link to rendered page should probably also be included in the readme for completeness; ah and the link to the repository too, yes

Cicero: https://cicero.readthedocs.io/ Sphinx documentation: https://www.sphinx-doc.org/en/master/ Carpentries workbench for creating Carpentries lessons: https://carpentries.github.io/workbench/ This is the testing lesson Richard showed: https://coderefinery.github.io/testing/ And the corresponding repository: https://github.com/coderefinery/testing You can lean back and watch, exercise coming in a bit :) Richard will now edit our train the trainer material repository: https://github.com/coderefinery/train-the-trainer The Sphinx extension for CodeRefinery lessons: https://github.com/coderefinery/sphinx-lesson Empty lesson template to build your own lesson: https://github.com/coderefinery/sphinx-lesson-template

More questions

- Does the coderefinery.org page built on sphinx?
  - It's built also from markdown but it uses https://www.getzola.org/ (which is more optimized to render websites, but we could do teaching also with Zola or create the project page using Sphinx)
- Is there a way to build and preview CR lessons without the command line?
  - The GitHub workflow we use creates HTML from all branches. This means that one can push a branch to GitHub, preview it there, and then open a pull request. One thing that would be nice but we don't have yet is if a bot automatically posts the link to the preview to a pull request.
- For VCS-2, for the Carpentries Linux shell example, Lesson, Github repo, to me it seemed as though the Sandpaper setup they have makes a very nice interface, complicates the relationship between source and output. E.g on the soucre page https://github.com/swcarpentry/shell-novice/blob/main/index.md I could only see to 'Prerequisits' and not 'Download files' on the page https://swcarpentry.github.io/shell-novice/. Perhaps I have missed something?
  - I am sorry I don't know the new Sandpaper setup well enough to answer this question.
- Do we have the instructions to build the lessons available somewhere to try out later?
  - https://github.com/coderefinery/sphinx-lesson

Exercise VCS-1 (and VCS-2 (if time)) (https://coderefinery.github.io/train-the-trainer/lessons-with-git/#exercises) in breakoutrooms until xx:05

then short summary in main room and then break.

Use the notes below. Collect a list of lesson formats and discuss what you like about each of them.

- PDF slides by a presentation program
- Jupyter + Nbviewer and custom css (read only) + Binder link (hands on)
- pdf slides with beamer in latex under source control.
- RMarkdown
- Markdown slides via Github
- Quarto Slides
- CodeRefinery template
- Reveal.js - write slides in HTML (also supports markdown)
- Carpentries template
  - Example: https://bl-presentations.erambler.co.uk/2024-02-21-idcc24/
- Google Slides
- Remark.js - write slides in markdown
- git-book
  - Example: https://bl-presentations.erambler.co.uk/2024-02-19-idcc-pids-workshop/
- Jupyter Book - allows creating a structured reference website using RST/Markdown, including Jupyter Notebooks to show result of running code
- mdbook - create simple/minimal website using Markdown
- Binder - allows learners to run their own live version of a github repository with Jupyter Notebooks, with only a web browser
- Jupyter notebooks and jupyterhub platform
  - CodeRefinery template

Room discussions:

- Room 1
  - (https://smc-aau-cph.github.io/SPIS/README.html)

- PDF slides by a presentation program
- Jupyter + Nbviewer and custom css (read only) + Binder link (hands on)
- pdf slides with beamer in latex under source control.
- RMarkdown
- Markdown slides via Github
- Quarto Slides
- CodeRefinery template
- Reveal.js - write slides in HTML (also supports markdown)
- Carpentries template
  - Example: https://bl-presentations.erambler.co.uk/2024-02-21-idcc24/
- Google Slides
- Remark.js - write slides in markdown
- git-book
  - Example: https://bl-presentations.erambler.co.uk/2024-02-19-idcc-pids-workshop/
- Jupyter Book - allows creating a structured reference website using RST/Markdown, including Jupyter Notebooks to show result of running code
- mdbook - create simple/minimal website using Markdown
- Binder - allows learners to run their own live version of a github repository with Jupyter Notebooks, with only a web browser
  - (https://smc-aau-cph.github.io/SPIS/README.html)
- Jupyter notebooks and jupyterhub platform
  - CodeRefinery template
  - Shared notes & blogs

Room discussions:

- Room 1
- Room 2
  - demo the exercise
- Room 3
  - Mix of JupyterLab, Terminal within that, and traditional slides
  - Jupyter + RISE (benefit: slides that are editable and executable live)
  - Challenges
    - present code in an interactive way
    - handle lots of images/ figures without too much additional overhead when setting up the material
- Room 4
  - Material and tools depend on instructors
  - Some slide building tools
    - https://revealjs.com/
    - https://remarkjs.com/
    - https://github.com/rust-lang/mdBook
  - Using flat git repos without fancy styling to make it easier to edit but still version controlled
- Room 5
  - Using R as a GIS: https://github.com/nickbearman/intro-r-spatial-analysis
    - Workbook: RMarkdown, Slides: Quarto
    - No continuous intergration
  - Having a jupyter nbinder link to test the notebook
  - repo lives on codeberg in this example
- Room 7
  - Ex 1:
    - Sphinx, similar to CR
    - Sometimes pdfs generated with beamer package (or from pptx and odt), but blows up repository size if in version control
  - Ex 2:
    - Content
      - CR: Content commonly found int the content folder
      - SC: Mostly under episodes
  - Question that came up:
    - is CR/Sphinx approach mostly for technical topics? What about language.
    - I guess by our nature it's focused there. But probably could be used for others (I guess it it's much easier to use git in a technical audience.)

- What tricks/techniques have you tried in your teaching or seen in someone else's teaching that you think have been particularly effective in collecting feedback from learners?
  - preparing a survey and emailing it to participants as soon as the event ends (usually get <50% of answers but it's a representative enough sample)
  - do engage with the audience, give them the time to get the courage to speak up
  - Be the audience yourself
    - yes! some problems/issues I don't notice as instructor, only as listener
  - Have time (e.g. 5 min) in the session to fill out the feedback form

Episode 3: How we collect feedback and measure impact

Materials: https://coderefinery.github.io/train-the-trainer/feedback-and-impact/

Questions to audience

- If the course/workshop has several days/sections do a couple questions every change, then a full one afterwards
- "Traffic light" feedback (e.g. for pacing or progress on exercises): give each learner two different coloured post-it notes for in-person, or use emoji for online, to indicate a

- What tricks/techniques have you tried in your teaching or seen in someone else's teaching that you think have been particularly effective in collecting feedback from learners?
  - preparing a survey and emailing it to participants as soon as the event ends (usually get <50% of answers but it's a representative enough sample)
  - do engage with the audience, give them the time to get the courage to speak up
  - Be the audience yourself
  - Have time (e.g. 5 min) in the session to fill out the feedback form
  - If the course/workshop has several days/sections do a couple questions every change, then a full one afterwards
  - "Traffic light" feedback (e.g. for pacing or progress on exercises): give each learner two different coloured post-it notes for in-person, or use emoji for online, to indicate a current status
- Can you give tips or share your experiences about how to convert feedback into changes or actionable items?
  - ask questions about things you know you don't know
  - "Don't let the noise of others' opinions drown out your own inner voice." — Steve Jobs
- When people ask questions / get stuck with a specific step. Is there a typo I can change there and then in the workbook?
  - Being able to fix small things on the fly helps sometimes, otherwise open issues if working on github
  - If multiple people have the same question, then this is an indication that I should look at this in more detail
- How do you measure the impact of your teaching? Any tips or experiences about what you have tried or seen other courses do?
  - "How likely are you to recommend this workshop to others? (0/5)"
  - Check the 'Garbage Out' bin, what you find there will be what went across. The rest will be history…
- Anybody knows of good resources on survey design? Please link them here.
  - "Our job is to figure out what they're going to want before they do… Our task is to read things that are not yet on the page." - Steve Jobs

Questions/Discussion points

- Feedback is necessary to design good lessons. Improvement can only be done iteratively. So the faster the iteration (design/teach), the better. But on the other hand, I cannot constantly bomb our community with training event emails. So I would like to try proposing a course many times a year, but I don't want to spam everybody many times a year.
  - :+1:
  - With CR we also have a calendar that can be imported to different calendar applications, where we add all workshops: https://github.com/coderefinery/calendar/
- Does an even split of too fast / too slow mean speed is about right?!
  - I take it to mean "it's roughly where it should be". Would be better to accomdate the sides better though… (we do try to design for a broad audience: some easy stuff, some advanced stuff, so people can make their own path)
  - It could also indicate that the prerequisites/scope are not well defined :+1:
  - I guess it depends how much it matters for the workshop itself. If you have group exercises etc, it probably matters? , lecture style maybe not so much?; important is perhaps the even split between too fast/too slow is ok only if the majority of votes goes for "just right" :+1:
  - Sometimese people that do not meet the prerequisites join to get to "know what they don't know", find out where to find information on topics they might get interested in future etc
- Do you have any suggestions for prerequisites? I try to make them clear, but occasionally people attend who clearly do not meet them. :+1:
  - Ask participants if they use Dev.to, hashnode etc to blog their work or use twitter etc to have a memory of their work.
  - Based on long term surveys we also know though that even though the workshop might have been to fast, people appreciate having "heard things before and know where to find them later"
  - that the participants themselves know that prerequisites exist and if they meet them
- One of the most important disciplines in journalism is to challenge your working premises. (Bill Keller) This applies equally to learning as well.
  - Other suggestions? Viewpoints? :)
  - Thank you - things for me to think about there :)
  - With CR's latest ideas, we try to make the prerequisites clear, but there is no capicity

Episode 3: How we collect feedback and measure impact

Materials: https://coderefinery.github.io/train-the-trainer/feedback-and-impact/

Questions to audience

- SC: Mostly under episodes
- Question that came up:
  - is CR/Sphinx approach mostly for technical topics? What about language.
    - I guess by our nature it's focused there. But probably could be used for others (I guess it it's much easier to use git in a technical audience.)
    - yes! some problems/issues I don't notice as instructor, only as listener

sides better though... (we do try to design for a broad audience: some easy stuff, some advanced stuff, so people can make their own path)
    - Ask participants if they use Dev.to, hashnode etc to blog their work or use twitter etc to have a memory of their work.
    - It could also indicate that the prerequisites/scope are not well defined :+1:
    - Based on long term surveys we also know though that even though the workshop might have been to fast, people appreciate having "heard things before and know where to find them later"
- Do you have any suggestions for prerequisites? I try to make them clear, but occasionally people attend who clearly do not meet them. :+1:
    - I guess it depends how much it matters for the workshop itself. If you have group exercises etc, it probably matters? , lecture style maybe not so much?; important is perhaps the even split between too fast/too slow is ok only if the majority of votes goes for "just right" :+1:
    - Sometimese people that do not meet the prerequisites join to get to "know what they don't know", find out where to find information on topics they might get interested in future etc
- One of the most important disciplines in journalism is to challenge your working premises. (Bill Keller) This applies equally to learning as well.
    - Other suggestions? Viewpoints? :)
    - Thank you - things for me to think about there :+1
    - With CR's latest ideas, we try to make the prerequisites clear, but there is no capicity limit: it's OK if people drop by and are less than prepared: it's livestream, anyone can come. We try to have something for these people, too: learn what you are missing, why you might want to study more in the future.
    - Any thoughts on applying this to a more practical focused course, with a total capacity limit (i.e. not like CodeRefinery).
- Everytime you do a course on CodeRefinery, think of it as trying to bake a new cake with an old recipe with no idea about the individual taste preferences of the one you bake the cake for. Then, you will not experience too many surprises. Fine-tuning the recipe for one cohort still remains the 'old recipe' as far as the next 'new cohort' will be concerned. C'est la vie.
    - nice analogy!
- Question from chat: For examples 2 & 4, would you also consider using something like a Likert scale? E.g. Rate how well you agree with the statement "My code is more reusable as a result of attending a CodeRefinery workshop" (from 1 Strongly disagree to 5 Strongly agree)
    - Thanks! I am new to Likert scale and will read up on it. In my experience numbered answers can be useful for reporting. But I also like questions where we ask for what actually to change.
- Comment from chat: Richard, your point crowns the point of having CodeRefinery, With Radovan's and Samantha's acquiescence assured, I would like to suggest you wear a crown going forward, especially for the CodeRefinery session. Do we have a deal ?
    - I don't quite understand this
    - I *think* it's a tongue-in-cheek-comment (joke) but I am not sure.....
- What is the ideal way to allow people to interact while course is going on in the main online session, i.e. without a breakout room? Taking a pause or discussion in the background?
    - I've had some success with WhatsApp groups for a course - particually ones that run over 4 (or 2) sessions.
    - A quick survey/quiz
    - Question to the audience that we take time to answer but it helps to show the answers

---

Wrapup

- If you want to co-teach with us in the upcoming (or next) CodeRefinery workshop (https://coderefinery.github.io/2024-09-10-workshop/); or just want to learn more about this opportunity: Send your name and e-mail address to support@coderefinery.org; we will contact you!
- Thank you for active participation :)
    - We'll continue answering questions after the workshop and make the cleaned up (no names etc) available from the materials
- Next session **Tuesday August 20, 9 CEST: Tools and techniques adopted in CodeRefinery workshops**

Feedback about todays session

What one thing was the most valuable/useful thing you learned today? Or generally one thing you liked about this session:

- we will talk about on-boarding, install help, roles, screensharing, sound, collaborative notes
- Lesson design approach
- All workshop materials will stay available (after the full workshop also on Zenodo :tools:)
- I liked the backwards lesson design, gives a name to a practice I've already been doing :+1:
- CodeRefinery community: https://coderefinery.zulipchat.com/
- :+1:
- Lessons from code refinery on lesson design. Learning about Sphinx.

- If you want to co-teach with us in the upcoming (or next) CodeRefinery workshop (https://coderefinery.github.io/2024-09-10-workshop/); or just want to learn more about this opportunity: Send your name and e-mail address to support@coderefinery.org; we will contact you!

Wrapup

- Thank you for active participation :)
  - We'll continue answering questions after the workshop and make the cleaned up (no names etc) available from the materials
- Next session **Tuesday August 20, 9 CEST: Tools and techniques adopted in CodeRefinery workshops**
  - we will talk about on-boarding, install help, roles, screensharing, sound, collaborative notes

Feedback about todays session

What one thing was the most valuable/useful thing you learned today? Or generally one thing you liked about this session:

- Lesson design approach
- All workshop materials will stay available (after the full workshop also on Zenodo :tools:)
- I liked the backwards lesson design, gives a name to a practice I've already been doing :+1:
- CodeRefinery community: https://coderefinery.zulipchat.com/
- :+1:
- Lessons from code refinery on lesson design. Learning about Sphinx.
- Collaborative notes and anonymized archive of notes.
- it's nice to have a community to discuss these kinds of problems
- Hearing the variety of tools people use for slides (PDF, Quarto, Beamer etc.)
- Good intro of the tools, the jargon and the pedagogical philosophy of teaching tech.
- The absolute openess in having everything publicly available - materials ofc but also questions, discussions, feedback, ...
- Thinking about all content being available publicly and as a git repo
- Thanks for explaining the HedgeDoc interface, with view and edit options
  - Is there anyway to come back to this document later (e.g. tomorrow) and see what has changed since now? So I can see if any other questions have been asked/answered?
    - There is a "revision" point under "Menu" up right, which shows you different versions of this document
- Modesty: the gentle art of enhancing your charm by pretending not to be aware of it. - Oliver Herford
-

What one thing would you improve about this session?

- Perhaps some more interactivity but can be hard to plan and time
  - We'll see what we can do.
- *Maybe* mix the breakout groups up for each breakout session (although there are advantages to consistency too) :+1:
  - yes, pros and cons! For me, having consistency worked well this time.
  - Likely we will have different participants next week, and a bit of mixing will happen naturally.
- The pace of the presentations could be a little tighter (the pace is good for a discussion though)
- Richard's voice was only 80% audible
  - Sorry to hear that. It seemed fine on my end. Will check better for next session.
- Hands on simulation and collaborative notes
- I found it difficult to follow the presentation and the collaborative notes at the same time
  - We will highlight the collaborative notes more in the next session. Our strategy usually is to say to focus on the "teaching part" and use the notes only when needed. Since you can go back and also check the notes later.
  - Someone speaking and explaining content at the same time as...
  - Chat in a different application at the same time as...
  - Content changing in a termin application.
    - My problem is that I want to follow both since I don't want to miss out on anything. As such, I have both windows (zoom and the notes) open side-by-side.
    - Thank you for the feedback. We will remove the Zoom chat out for the next session and explain the use of the collaborative notes more.
- Make more use of defining and then making sure the learning objectives are met
  - True we did not talk about them much, will pick that up better for next session.
- Heads up to participants on using the break out rooms on voice interaction and screen sharing.
- UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity. - Dennis Ritchie. You need to do a lot to prove that to be wrong, simply put.
  - Do you mean that we should have mentioned it more clearly in the pre workshop email?
- Allocate some time for learners to type in the answers so that we don't have to listen and type at the same time.
- I found it hard to follow with so many things going on at once...
  - A collaborative document being changed at the same time as...

Any other comments?

usually is to say to focus on the "teaching part" and use the notes only when needed.
    - Someone speaking and explaining content at the same time as…
    - Chat in a different application at the same time as…
    - Content changing in a termin application.
        - My problem is that I want to follow both since I don't want to miss out on anything. As such, I have both windows (zoom and the notes) open side-by-side.
        - Thank you for the feedback. We will remove the Zoom chat out for the next session and explain the use of the collaborative notes more.
        - Since there is near-constant real-time typing in the editor, and as such visible movement on screen, I get constantly distracted. This makes following the event quite exhausting.
- Make more use of defining and then making sure the learning objectives are met
    - True we did not talk about them much, will pick that up better for next session.
- Heads up to participants on using the break out rooms on voice interaction and screen sharing.
- UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity. - Dennis Ritchie. You need to do a lot to prove that to be wrong, simply put.
    - Do you mean that we should have mentioned it more clearly in the pre workshop email?
- Allocate some time for learners to type in the answers so that we don't have to listen and type at the same time.
- I found it hard to follow with so many things going on at once…
    - A collaborative document being changed at the same time as…

Any other comments?

- It's great to hear other people's experience
    - Good to hear :)
- Looking to meet you onsite if there are any events and get more emails from you :)
    - We will keep you informed :)
- Thanks for putting this together, I got a lot of inspiration for my own courses
    - Good to hear :)
- I didn't know where to ask questions. It would work best *for me* to do it in zoom chat as that's the application that I'm watching in..
    - Sorry for that, just to figure out why: Did you join a bit late?
        - No. Perhaps I just missed that or it was too quick for me or something?
            - Ok, sorry. Will try to make this more clear in next session.
        - Perhaps also because there were so many things going on and jumping around. For me, the zoom chat is the constant (and obvious) place to ask questions while watching a shared screen, trying to find the right place in a moving collaborative document is difficult. It tears my attention into too many pieces. :+1:
- Had a lot of content that assumed extant knowledge that I didn't have. This wasn't in the pre-requisites.
    - Do you have examples?
        - Yes! The code refinery templates and even just the way of doing that. I was expecting more on "when designing a lesson you should think about these steps" rather than this is how we do it and this is our tool for doing it. I guess I should have read the pre-workshop content better?
        - Ok, thank you for your feedback. You are right that we actually had some content that for sure would have been easier to follow for people with background knowledge about a certain topic.
        - "If you are in CodeRefinery TTT, you probably know what version control is and why it is important." - extant knowledge, why not just add a sentence or two?
        - Please can you update in preparation for the next workshops so that I can decide whether or not they will be right for me?
            - Yes, we will think more thoroughly about this and send the info in the next email going out Wed/Thu
                - Thank you!
- When you have pre-written "thank you for your active participation" it feels fake!
    - He, that is true, did not think about that. Thank you for the feedback. I would not have added it if Radovan and Richard would not have provided good feedback from circling the rooms.
- I found it vague and disorganised and difficult to follow. It felt like the instructors were presenting the content for the first time and didn't know what to do next.
- Lots of the content was relevant to code refinery but not the wider topic (e.g. it was how to design a code refinery lesson and not how to design a lesson as stated in the session title)
    - Thank you for your feedback. This is not the first time we have this training, but it is the first time in this constellation/way. The way of teaching may also sometimes make it seem new to instructors.
    - We thought of the content of showing one way of doing things, ie "what we have learned", if you have suggestions on how we could make that more clear on the event page, please let us know :)
- I really liked the active exchange with other participants, but I also would have enjoyed a bit more technical content in the presentations. In a sense it felt less like a 'training' and
    - "in a code refinery lesson" in the lesson/workshop description?

        - Please can you update in preparation for the next workshops so that I can decide whether or not they will be right for me?
            - Yes, we will think more thoroughly about this and send the info in the next email going out Wed/Thu
                - Thank you!
- Lots of the content was relevant to code refinery but not the wider topic (e.g. it was how to design a code refinery lesson and not how to design a lesson as stated in the session title)
    - We thought of the content of showing one way of doing things, ie "what we have learned", if you have suggestions on how we could make that more clear on the event page, please let us know :)
        - "in a code refinery lesson" in the lesson/workshop description?

- When you have pre-written "thank you for your active participation" it feels fake!
    - He, that is true, did not think about that. Thank you for the feedback. I would not have added it if Radovan and Richard would not have provided good feedback from circling the rooms.
- I found it vague and disorganised and difficult to follow. It felt like the instructors were presenting the content for the first time and didn't know what to do next.
    - Thank you for your feedback. This is not the first time we have this training, but it is the first time in this constellation/way. The way of teaching may also sometimes make it seem new to instructors.
- I really liked the active exchange with other participants, but I also would have enjoyed a bit more technical content in the presentations. In a sense it felt less like a 'training' and more like a get-together :+1:.
    - Thank you for your feedback. We thought the name "workshop" would combine the training and exchange nature of this event. But maybe it didn't do so enough? Do you have suggestions on how to clarify it on the event page?
        - Well, the 'train the trainer' title created the anticipation of being trained ;) don't get me wrong, I liked the exchange, but I felt that the training aspect fell a little short. imho the workshop would benefit from a little more (instructional) content that is presented in a more structured way by the instructors (as opposed to 'crowd-sourced knowledge' that is collated in a loose collection of questions, notes and links)
        - Other than that, you could add another section on the main page, sth like 'Workshop structure: we aim for a mix of modular presentations and work in small groups. The latter will be done in breakout rooms - be prepared to switch on your camera and have a fruitful exchange with fellow participants!'
            - Thank you for the suggestions, will add something like this :tools:
                - Added : https://github.com/coderefinery/train-the-trainer/commit/03308595436e636b3bb37ed9d1f0dee39eeccd0f
- My understanding (so far) of what (unlike the competition) CodeRefinery DOES NOT seek to exemplify - https://images.ctfassets.net/rxqefefl3t5b/2k45BFtw4sBMlBfjr1OZ90/5c6ecd3628a43b0cdc79815f665e224a/Screenshot_2023-05-02_at_09.35.23.png?fl=progressive&q=80
- My understanding of what (unlike the competition) today's session of CodeRefinery DID exemplify - Definition of 'Workshop' from the Oxford dictionary of the English language "a period of discussion and practical work on a particular subject, in which a group of people share their knowledge and experience"
- In my experience, even for the most seasoned Programmer coding, by its very nature, is nothing trivial, this being a simple straightforward fact just like there is no such thing as an egg which is a cube. I mean there is a reason why not just about everybody is a poet either. But, other minds have pointed this out without having to first explain why the universe is what it is in a way that could be understood without effort … https://i.pinimg.com/originals/7b/7e/c4/7b7ec4b91d2[illegible]4678[illegible].jpg
- The inclusive environment feels very welcoming. Keep up the good work!

Thank you all for your feedback! Highly appreciated!

Day 2 : Session 2 (20.08.24)- Tools and techniques adopted in CodeRefinery workshops

:calendar: Schedule

Time (CEST)	Title	EEST (UTC+3)	BST (UTC+1)
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 9.25	About CodeRefinery workshops	10.15 - 10.25	8.15 - 8.25
9.25 - 9.55	Collaborative notes and interaction	10.25 - 10.55	8.25 - 8.55
9.55 - 10.10	Break	10.55 - 11.10	8.55 - 9.10
10.10 - 10.45	Workshop overview, roles, onboarding	11.10 - 11.45	9.10 - 9.45
10.45 - 11.05	Sound	11.45 - 12.05	9.45 - 10.05
11.05 - 11.20	Break	12.05 - 12.20	10.05 - 10.20
11.20 - 11.50	How to prepare a quality screen-share	12.20 - 12.50	10.20 - 10.50

Time (CET)	Title	EEST (UTC+3)	BST (UTC+1)
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 9.25	About CodeRefinery workshops	10.15 - 10.25	8.15 - 8.25
9.25 - 9.55	Collaborative notes and interaction	10.25 - 10.55	8.25 - 8.55
9.55 - 10.10	Break	10.55 - 11.10	8.55 - 9.10
10.10 - 10.45	Workshop overview, roles, onboarding	11.10 - 11.45	9.10 - 9.45
10.45 - 11.05	Sound	11.45 - 12.05	9.45 - 10.05
11.05 - 11.20	Break	12.05 - 12.20	10.05 - 10.20
11.20 - 11.50	How to prepare a quality screen-share	12.20 - 12.50	10.20 - 10.50
11.50 - 12.00	Outro and feedback	12.50 - 13.00	10.50 - 11.00

:icecream: Icebreaker

Check-in

Which emoji best describes your current state of mind? Emoji cheat sheet

- :-)
- https://media.istockphoto.com/id/148421550/vector/bored-emoticon.jpg?s=612x612&w=0&k=20&c=uo5KBTqdJZbBQb_XIdhzuovx5t08fHfGR7dw7TCk90I=
- :saluting_face:
- :sleepy:
- :coffee:
- :umbrella:
- :cocktail: :coffee:
- :seedling:
- :tired_face:
- :sun_with_face:
- 🥴 (:woozy_face: not converted?)
- :coffee:
- :sunflower: :book:
- :satisfied:
- :coffee:
- :coffee:
- 🥱:tired_face:
- :bread: - I am baking bread today so I have been kneading dough while listening!

Introduction in breakoutrooms

Name / Affiliation / Location

Teaching

- Do you teach and organize teaching alone or with others? What would you prefer and why?
  - Most of my teaching is on my own, as a freelancer its more expensive to work with colleagues than to deliever a course on my own.
  - Teaching together, learns from each other, feedbacks to improve
  - Not formally taught yet, only given presentations actually .
  - Coming together is a beginning; keeping together is progress; working together is success. – Edward Everett Hale
  - collective teaching would benefit the teachers with collaborative curriculumn and experiences sharing.
  - These days always with others. Alone is easier to prepare but almost always is harder
- If applicable, have you seen any challenges when teaching together and how to overcome them?
  - during it.
  - Teaching with very diverse partners, it can be challenging to find a common language
  - It actually requires preparation,
  - with people very different than you.
  - Is it about teaching or about inspiring ? Discover the answer and get inspired ...
  - I teach alone, in the future there will be more collaboration with multiple team
  - It's like being in a band: you might be great at improvising and your band-partners members. Both approaches have their pros and cons.
  - might be great at improvising too, but a bit of rehearsing makes even the impro gig much better
  - Both has its own distinct advantages

Day 2 : Session 2 (20.08.24)- Tools and techniques adopted in CodeRefinery workshops

:calendar: Schedule

Thank you all for your feedback! Highly appreciated!

- The inclusive environment feels very welcoming. Keep up the good work!

Teaching

- Most of my teaching is on my own, as a freelancer its more expensive to work with colleagues than to deliever a course on my own.
- Do you teach and organize teaching alone or with others? What would you prefer and why?
  - Teaching together, learns from each other, feedbacks to improve
  - Not formally taught yet, only given presentations actually .
  - Coming together is a beginning; keeping together is progress; working together is success. – Edward Everett Hale
  - collective teaching would benefit the teachers with collaborative curriculumn and experiences sharing.
  - These days always with others. Alone is easier to prepare but almost always is harder during it.
- If applicable, have you seen any challenges when teaching together and how to overcome them?
  - Teaching with very diverse partners, it can be challenging to find a common language
  - It actually requires preparation,
  - with people very different than you.
  - Is it about teaching or about inspiring ? Discover the answer and get inspired ...
  - I teach alone, in the future there will be more collaboration with multiple team members. Both approaches have their pros and cons.
  - It's like being in a band: you might be great at improvising and your band-partners might be great at improvising too, but a bit of rehearsing makes even the impro gig much better
  - Both has its own distinct advantages
  - Need to plan before session, difficult if people don't 'plan' in the same way, e.g. with the same time frame.
  - Heterogeneity in learners make it impossible to get the same result from everyone.
    - In fact, the same applies to teachers :D
  - Teaching together requires to have similar opinions on how to teach. I share vews only with ~40% of my colleagues I think.
- No Teaching experience but it is good to give the outline of the lessons beforehand and learning goals.

:question: Questions

General / Practicalities

- What is this document for?
  - Asking questions, collecting discussion points, taking collaborative notes (Zoom chat deactivated, can only be used for communication with the hosts)
- Will the material be made available?
  - Materials are and will stay available at https://coderefinery.github.io/train-the-trainer/
    - Can I share them with a colleague
      - Yes please :)
    - Can I resuse them in my own teaching?
      - Yes please :)
- Will these sessions be recorded?
  - No, but the materials (see above) should include everything needed for self-learning.
- Can I get a certificate for this workshop?
  - You can get a certificate of attendance after the workshop by sending an e-mail to support@coderefinery.org, telling us a bit about what you have learned and which sessions you attended.
- How often will we be talking in breakout rooms?
  - We'll have about one breakoutroom session per episode
  - BR 1 : The Zoom chat does not work for the breakout room so we should take note of that. No webcam/mic = expect silence.
    - Thanks for notifying! This was not intended. Turned the chat back on.
  - One idea we have had is that paid courses could be delivered through the DRA (link below), who we know. Basically, DRA could find instructors and handle the money transfers (If someone wanted a paid course, we could help find someone to do it through DRA or independently)
  - :smile:
- Have you come across DRA, https://digital-research.academy/, Heidi Seibold, Joyce Kao who are doing similar work? I can connect you if you like.
  - Yes we have been discussing about possible future collaborations and are already sending possible clients their way (we, as a project cannot take money) :+1:
    - Excellent! :-D :smile:
  - We've discussed it but right now we don't have many people who could accept money to do this (and maybe shouldn't, because of their jobs). But, we would encourage others to use our materials as "independents" to deliver paid courses

---

Episode 1: CodeRefinery

Materials: https://coderefinery.github.io/train-the-trainer/coderefinery-intro/

- (questions continue here)
- Is there the plan to build capacity / be paid to deliever courses?
- .

Episode 2: Collaborative Notes

that. No webcam/mic = expect silence.
    - One idea we have had is that paid courses could be delivered through the DRA (link below), who we know. Basically, DRA could find instructors and handle the money transfers (If someone wanted a paid course, we could help find someone to do it through DRA or independently)
      - Thanks for notifying! This was not intended. Turned the chat back on.
      - :smile:
  - Have you come across DRA, https://digital-research.academy/, Heidi Seibold, Joyce Kao who are doing similar work? I can connect you if you like.
    - Yes we have been discussing about possible future collaborations and are already sending possible clients their way (we, as a project cannot take money) :+1:
      - Excellent! :-D :smile:
    - We've discussed it but right now we don't have many people who could accept money to do this (and maybe shouldn't, because of their jobs). But, we would encourage others to use our materials as "independents" to deliver paid courses

Episode 1: CodeRefinery

Materials: https://coderefinery.github.io/train-the-trainer/coderefinery-intro/

- (questions continue here)
- Is there the plan to build capacity / be paid to deliever courses?
- .

Episode 2: Collaborative Notes

Materials: https://coderefinery.github.io/train-the-trainer/collaborative-notes/

- So anyone can type anonymously?
  - Yes, as long as you do not log in, you are Guest XXX
  - And at least in our hedgedoc case you cannot log in so you will always be anonymous. You can do [name=xxx], to add your name to a comment
- Test of +1
  - +1 :+1: :+1:
  - :-1: a test
- You are hosting your own version of HedgeDoc, and Hackmd.io is a similar (free) option. What are the limitations of the free hackmd.io? Should each center install their own HedgeDoc instance?
  - It **seems** the free version of HackMD has gotten worse lately. It works still well enough with a few people editing at the same time.
- Have there every been problems with the anonymity and CoC?
  - We haven't had any issues so far.
  - With a different anonymous tool in a bachelor course, we had issues with anonymous participants (participants were posting slurs and unsafe links). The course organiser decided to stop allowing anonymous tools.
- Why not google documents?
  - Google documents are actually setting a hard limit for simultaneous writing, no more than 100 concucurrent editors. I am unsure if microsoft onedrive word documents would allow more users writing at the same time.
- What's the most people you have every had on HackMD / Hedgedoc?
  - I believe about 200 active participants?
- What are your thoughts on using the build-in Zoom Q&A?
  - We haven't tried this, in part because the way we do streaming now (you'll see in session 4), participants aren't in Zoom so don't have access to that. The doc-format works pretty well though, when we keep it in this limited "write at end" format.
- How do you see what has changed? e.g. if leave the session at the end, can I come back tomorrow and see X has been added? e.g. a diff for this maybe?
  - In the menu up right there is revision, it does show the document and changes at different time step :+1: Thanks!
  - And if the instructors never screenshare it to show what goes on it, then seems to be less used.
  - We recommend people only write at the bottom (and since we answer things live), so as a first level approximation you find where you were and read below that. Think of it more like chat than a full document. (Yes, we do revise questions at the end)
  - And if there aren't other people to read/answer it during the session (perhaps less likely in a small course), it seems to be less used.
- How about flinga boards or similar tools?
  - My take is that the simplicity of a simple text document wins over free-form in-person-workshop-like tools.
  - There's miro, can be a bit confusing though
- How does this work with small groups, e.g. 10-15 people?
  - Good question: it can be hard if people aren't rigorous about it, people start only using chat/voice. So there seems to be some lower bound in size but we haven't tested this much
  - Use of etherpad for collaborative discussion.
  - Use of slido for polling and quizzing.

Breakoutrooms until xx:55

Exercise/discussion: https://coderefinery.github.io/train-the-trainer/collaborative-notes/#exercise

- Breakout room 1:

different time step :+1: Thanks!
    - And if the instructors never screenshare it to show what goes on it, then seems to be less used.
    - We recommend people only write at the bottom (and since we answer things live), so as a first level approximation you find where you were and read below that. Think of it more like chat than a full document. (Yes, we do revise questions at the end)
    - And if there aren't other people to read/answer it during the session (perhaps less likely in a small course), it seems to be less used.
- How about flinga boards or similar tools?
    - My take is that the simplicity of a simple text document wins over free-form in-person-workshop-like tools.
    - There's miro, can be a bit confusing though
- How does this work with small groups, e.g. 10-15 people?
    - Good question: it can be hard if people aren't rigorous about it, people start only using chat/voice. So there seems to be some lower bound in size but we haven't tested this much

Breakoutrooms until xx:55

Exercise/discussion: https://coderefinery.github.io/train-the-trainer/collaborative-notes/#exercise

- Breakout room 1:
    - Use of etherpad for collaborative discussion.
    - Use of slido for polling and quizzing.
    - Presemo as a slido alternative (https://www.aalto.fi/en/services/presemo-a-fully-web-based-classroom-participation-system)
    - Use of text based approach for questions rather than live can be helpful if there is a language barrier issue.
    - A good teaching session which had many questions coming spontaneously to the teacher is a non-entity of sorts. If the overall flow of feedback / questions increases it could indicate that the session will not prove to be a helpful experience. A signal that generates noise is, well, noise, if calling a spade a spade signals anything. Note that the 'overall' bit and the 'to the teacher' aspect are important. Both the best and the worst teaching session have two things in common: a teacher, and the fact that NO Materials of any kind are used, ie speech is the only channel of communication via a common language. The only difference is that it is not the same teacher, at least not in the same session at the same time.

  Another less mentioned, but perhaps more relevant indicator of a great session is the simple fact that at the end of the session EVERYONE feels empowered, NOT tired. This is not an easy to quantify thing, but screams out the quality of the experience loud. More often then not, such sessions differ little from each other, apart from in terms of the theme / topic and the participant mix. It ends up being the 'keep your cake and eat it too,' of sorts, not quite the far less amusing post-bombardment devastation scenario in the wake of a take it or leave it go at anything.

    - profanity in collaborative documents where people can add content anonymously could be a problem (happens very rarely, fortunately and somewhat surprisingly)

- Breakout Room 2:
    - Good experiences with Zoom polls
    - Handling questions in hybrid setups: all participants (on-site and online) ask questions in written form
    - Collaborative notes can be challenging when conducting a course alone (without teaching assistants) :+1:
    - Providing a method to ask questions anonymously can make asking easier for shy participants
    - It can help somebody to start and create a structure which helps others to see and follow
    - Questions can give valuable feedback on how the course is going
    - Online collaborative whiteboarding platforms such as Miro
        - Large monitors are useful for this not to have to drag a "periscope" around
    - we discussed the Carpentries sticky notes system (red sticky note: signal tech issues and use for feedback, green sticky note: signal that exercise worked and for positive feedback)
        - Nice summary of sticky notes system: https://teachtogether.tech/en/#s:classroom-sticky-notes :+1:
        - reply to question is a way to handle multiple questions at the same time but it works well for smaller groups
- Room 4
    - good experience with whatsapp chat for questions. since most learners have the app already and one thing less to install and set up
    - What is the success rate of in-person workshops after COVID? Discussion: Teachers prefer in person and learners prefer remote or recorded to spool back and relearn.
- Room 5
    - experience with collaborative docs in in-person workshops: voice questions might win over the document
    - Good experience with Collaborative tools as HackMD and HedgeDoc while teaching with no bad feedback
    - "seeding" questions may help to not have an empty document
        - When used with smaller groups works best as one-way, like sharing links with

participants
    - It can help somebody to start and create a structure which helps others to see and follow
    - Questions can give valuable feedback on how the course is going
    - Online collaborative whiteboarding platforms such as Miro
    - we discussed the Carpentries sticky notes system (red sticky note: signal tech issues and use for feedback, green sticky note: signal that exercise worked and for positive feedback)
        - Large monitors are useful for this not to have to drag a “periscope” around
        - Nice summary of sticky notes system: https://teachtogether.tech/en/#s:classroom-sticky-notes :+1:
- Room 4
    - good experience with whatsapp chat for questions. since most learners have the app already and one thing less to install and set up
    - What is the success rate of in-person workshops after COVID? Discussion: Teachers prefer in person and learners prefer remote or recorded to spool back and relearn.
        - reply to question is a way to handle multiple questions at the same time but it works well for smaller groups
- Room 5
    - experience with collaborative docs in in-person workshops: voice questions might win over the document
    - Good experience with Collaborative tools as HackMD and HedgeDoc while teaching with no bad feedback
    - “seeding” questions may help to not have an empty document
        - When used with smaller groups works best as one-way, like sharing links with students and so on
        - Needs to be done as a team, quite difficult if teaching alone
        - Most useful when online/hybrid
    - Anonymisity is important for transparent feedback and no biasis in classrooms
    - Not very collaborative but useful to share history or jupyter notebooks live https://github.com/mwakok/software_carpentries?tab=readme-ov-file#set-up-gitautopush
    - Used Flinga before that seems like Figma and also onsite teaching using sticky notes
        - There’s miro.com but can be a bit confusing, more for brainstorming than for teaching
    - Using quizzes apps like Kahoot is beneficial while teaching to have more engaging students
    -
- Room 6
    - teaching teenagers (in presence) but using shared pad becomes quickly very chaotic, so only usable for short moments when answering very precise questions
        - Yeah, I can imagine. For CR we try to keep it in this very limited format (all questions bullet points, only add to the end) and I think that helps.
    - using shared notepad for small workshop (5-10 people) to explore common thematic (mainly technical) to keep a shared notes and condense aftwerward (and publish on wiki to archive)
    - Difference between teaching in person and online. Some people prefer in-person and sometimes you have no option but to organise it online.
    - There are sometimes cultural differences and learners feel shy to ask / question the knowledge of the Teacher, so shared notes are a good way to safely ask questions.

Questions continue:

- Have we demonstrated how we do polls in HedgeDoc? Add your vote by adding an `o`
    - yes: oooo *(in session 1)*
    - no:
    - :smile:
    - not sure: o

Episode 3: One workshop, many perspectives

Materials: https://coderefinery.github.io/train-the-trainer/overview/ Next CodeRefinery workshop in September: https://coderefinery.github.io/2024-09-10-workshop/

Exercise until xx:39

https://coderefinery.github.io/train-the-trainer/overview/#discussion

- Room 1
    - Send pre-workshop email with pre-requisite instructions on what should be set-up beforehand.
        - But also mention that if help is needed for set-up, we will be ready before the session officially starts.
    - Discussion on whether we push to work in a pre-ready environment (e.g. Binder, Colab) or in their own environment.
    - Lots of help needed both before and after.
    - Installations: as much as we can provide ready made environments for the learners, maybe installations will be something that the learners will face sooner or later. Maybe one should teach more about installations?
    - My only cherished role in a teaching-learning session is the that of one who is living the moment as if there were no tomorrow. What cuts all ways really is that “By failing to prepare, you are preparing to fail.” — Benjamin Franklin. A helpful motto: “When one

- Send pre-workshop email with pre-requisite instructions on what should be set-up beforehand.
- But also mention that if help is needed for set-up, we will be ready before the session officially starts.
- Discussion on whether we push to work in a pre-ready environment (e.g Binder, Colab) or in their own environment.
- Lots of help needed both before and after.
- Installations: as much as we can provide ready made environments for the learners, maybe installations will be something that the learners will face sooner or later. Maybe one should teach more about installations?
- My only cherished role in a teaching-learning session is the that of one who is living the moment as if there were no tomorrow. What cuts all ways really is that “By failing to prepare, you are preparing to fail.” — Benjamin Franklin. A helpful motto: “When one teaches, two learn.” — Robert Heinlein. The best experience cannot be given, it can only be allowed to flow its way, so get out of its way.

- Room 2
  - Frequently, a single person fills all the roles for a single or multi-day workshop
  - Collaboration with different centers/ universities can be challenging
  - Preparing participants: it’s essential for the instructor to go through the demos and be sure that they will work (no unpleasant surprises live). This can be extended to advance-prep for participants.
  - Best experience is to hear (live or in feedback notes) some positive things directed at you :)
  - Fully prepared environments (eg self-hosted JupyterHub) can make preparation for participants easy

- Room 3
  - Some students have problems with setuping the environment. Maybe having a step by step instructions could help. But some students don’t want to “start” the course before hand.
  - Preparing the environemnt before hand ensures everyone is onboard from day 1. The same thing happend in CSC training sessions and another session Peng attended.
  - Had experience in attending as teacher and student in previous workshops

- Room 4
  - knowing the background of the audience is helpful
    - Try and ask in advance of the background of the audience. Do a poll at the start of the workshop. In person one can do it interactive.
  - among co-instructors: discuss topics and flow
  - if more people are involved, it can help changing roles from time to time to see different perspectives
  - preparation is key
  - Circulate instructions beforehand
    - Do a poll before session: Have you installed (QGIS) successfully?
    - Push hard to do this in the welcome email
    - Having the material readily available online ans sharing it
  - communicate clearly to the participants
  - Making a pre-survey
  - What was the best workshop experience for you as learner, helper or instructor? What made it great?
    - learner
      - Engaging and practical workshop with more exercices and follow-up theoretical part explanations
    - helper
      - When the instructor is well prepared and as helper ther’s only typos to fix and interesting questions to answer
    - instructor
      - Assist the instructor, coordinate the collaborative notes, and make sure that the time is respected
  - How to prepare in advance?
    - Collect resources and prepare the workshop plan

- Room 5
  - We’ve all taken most roles
  - How to prepare in advance?
    - Meet and discuss once or twice
    - Make a pre-survey to check for interest and background
    - Going through the materials
    - Checking the toolkit (zoom)
    - Agree on the responsibilities and back-up plans (bit of scenarios planning)
  - How to prepare trainings in advance?

Episode 3: One workshop, many perspectives

Materials: https://coderefinery.github.io/train-the-trainer/overview/ Next CodeRefinery workshop in September: https://coderefinery.github.io/2024-09-10-workshop/

Exercise until xx:39

https://coderefinery.github.io/train-the-trainer/overview/#discussion

    - Having the material readily available online ans sharing it
    - communicate clearly to the participants
    - Making a pre-survey
  - What was the best workshop experience for you as learner, helper or instructor?
    - What made it great?
      - learner
        - We've all taken most roles
        - How to prepare in advance?
          - Meet and discuss once or twice
          - Make a pre-survey to check for interest and background
          - Going through the materials
          - Checking the toolkit (zoom)
          - Agree on the reponsibilities and back-up plans (bit of scenarios planning)
        - How to prepare trainings in advance?
          - Collect resources and prepare the workshop plan
      - Engaging and practical workshop with more exercices and follow-up theoretical part explanations
      - helper
        - When the instructor is well prepared and as helper ther's only typos to fix and interesting questions to answer
        - Assist the instructor, coordinate the collaborative notes, and make sure that the time is respected
      - instructor
        - I have a colleague whom I've ran the same workshop a few times already, goes quite smoothly now and we've made a few changes. Hard to reach that point with new colleagues but getting there
        - Students/participants were engaging with the material and many good feedback collected even some of them asked for the next workshop
- Room 6
  - What has made a great experience for you as a learner/helper/instructor?
  - **Common ground between learners & teachers**
  - Motivation
    - Teaching is much more enjoyable when the learners want to be there
    - I spend a lot of time at the start of a workshop making sure learners feel that the content is possible for them to understand and master

Poll: I have joined a CodeRefinery workshop before: (add an `o` for your answer)

- yes (within last 2 years): ooooooo
- yes (longer time ago): oo
- no: oooooooo
-

(questions would continue below:)

- What is the motivation / advantages of streaming workshop versus using zoom?
  - We'll cover this more in week 4, but roughly: zoom became unmanageable and essentially quiet once things got big. Streaming allowed "one to many" communication with no limit, and collab notes allowed better interaction anyway. By separating the rooms, it was more pleasant to teach. Videos could be released immediately since there was no risk of privacy of participants. There could still be in-person/online small groups who have "watching parties."
- What are the blue red and green font colors here, does it symbolify anything or just a convention HedgeDoc uses?
  - Visual separation of formatting, mostly. (It helps to quickly realize you've missed/ mistyped a control Markdown symbol, etc.)
  - Yeah, it's basically "code highlighting" for markdown. No special meaning but quite convenient for us.
- Can a person have multiple roles in a workshop as a helper and a teacher?
  - Yes, many teachers also contribute to the rest of the workshop by answering questions in collaborative notes. :+1:
  - Some teachers also host a local classroom in week one and do their teaching in week two of the workshop
  - Roles that do not go well together are broadcaster/director/host and instructor (goes well until there is a problem; co-teaching solves that partly)
- Is this next workshop only for team leads or also open for helpers and coordinators?
  - You mean the next session of this workshop or the upcoming "CodeRefinery workshop"(this one: https://coderefinery.github.io/2024-09-10-workshop/)?
    - If interested: Join the discussion in our chat: https://coderefinery.zulipchat.com/#narrow/stream/316508-coderefinery-tools-workshop/topic/Fall.202024.20CodeRefinery.20workshop
    - The upcoming workshop coordinators are alredy working, but can be joined if you are interested
    - As helper roles we currently only have "collaborative notes helper", which is all about helping to answer questions in the collaborative notes
    - We do have some co-teaching slots open :)

Episode 4: Sound

https://coderefinery.github.io/train-the-trainer/sound/

    - Visual separation of formatting, mostly (It helps to quickly realize you've missed/ mistyped a control Markdown symbol, etc.)
    - Yeah, it's basically "code highlighting" for markdown. No special meaning but quite convenient for us
- Can a person have multiple roles in a workshop as a helper and a teacher?
- Is this next workshop only for team leads or also open for helpers and coordinators?
    - Yes, many teachers also contribute to the rest of the workshop by answering questions in collaborative notes. :+1:
    - You mean the next session of this workshop or the upcoming "CodeRefinery workshop"(this one: https://coderefinery.github.io/2024-09-10-workshop/)?
    - Some teachers also host a local classroom in week one and do their teaching in week two of the workshop
        - The upcoming workshop coordinators are alredy working, but can be joined if you are interested
    - Roles that do not go well together are broadcaster/director/host and instructor (goes well until there is a problem, co-teaching solves that partly)
        - As helper roles we currently only have "collaborative notes helper", which is all about helping to answer questions in the collaborative notes
        - We do have some co-teaching slots open :)

Episode 4: Sound

https://coderefinery.github.io/train-the-trainer/sound/

- To me, Richard is a bit quieter than the other two :+1: :+1: :+1:
    - to elaborate: I think that the audio coming from him is missing some mids
    - Do you have any links about this kind of evaluation and adjustment?
- Do you suggest / recommend a specific headset(s) with mounted microphone?
    - in our experience headsets with a microphone will almost always be better than a separate microphone sitting on the desk or in the monitor
    -
- Speakers have different tones of voice and speaking styles, is there a recommendation or experience from CR workshops for how this is adjusted and catered to environemnts for listeners?
    - Yeah, people's voices are different and we haven't gone as far as voice training or improving.
- How to change the volume for speakers? Tried Zoom settings and system settings and didn't help.
    - From linux I can use different pulsemixers `pavucontrol`, `pulsemixer` to set the gain to above 100%. I'm not sure the equivalent on other OSs but

---

Exercise until xx:10 https://coderefinery.github.io/train-the-trainer/sound/#exercises

- room 1
    - Practically, no typical headset is designed or can capture the technical purity of human voice within its natural environment. But one needs to understand the polar pattern of the (specific) microphone or how much of the signal will be picked up by the microphone from different directions, a sine qua non when it comes to mitigating unwanted sound sources to bleed into the signal.

For best results, keep the right direction & distance to your (directional) mic. Usually, that's between 15-30 cm to get get a natural sound. A microphone records everything it can, also silence, so check the headphone output BEFORE adjusting the gain, in some specific cases aim for a dry sound, ie the influence of the room or how audible the room is on the recording ! Finally, adjust the mix between dry and room sound, or change the frequency response in tandem using a compressor to reduce the dynamic range of a signal for a more consistent listening experience = the quietest parts of your signal will appear louder by reducing the loudest parts.

It is all about one thing and one thing only - love your voice, let people fall in love with the acoustic quality of your charm. So remember: GIGO, it matters!

- room 4
    - headset is probably better listening experience but they can be uncomfortable to wear during long meeting days :+1:
        - Yeah. Luckily only instructors
        - Therefore breaks are non-negotiable
    - loud keyboard can be an issue
    - If one has a silent room that does not echo, it can be ok to teach without headset
    - on Linux I am using `pulsemixer` to adjust levels and `pavucontrol` to change outputs/sources
- Room 3
    - Had different headsets and we could see the difference (bluetooth low quality)
    - mic modes on mac
    - How to change the volume?
    - check default microphone for zoom.
        - It's surprisingly hard to get more control than the basic slider you see in the apps!
        - You need to look
- room 5
    - Most tricks help tuning yourself but hard to do it as a group. Perhaps doing a short

- room 4
  - headset is probably better listening experience but they can be uncomfortable to wear during long meeting days :+1:
    - Yeah. Luckily only instructors
      - Therefore breaks are non-negotiable
  - loud keyboard can be an issue
  - If one has a silent room that does not echo, it can be ok to teach without headset
  - on Linux I am using `pulsemixer` to adjust levels and `pavucontrol` to change outputs/sources
- Room 3
  - Had different headsets and we could see the difference (bluetooth low quality)
  - mic modes on mac
  - How to change the volume?
  - check default microphone for zoom.
    - It's surprisingly hard to get more control than the basic slider you see in the apps! You need to look.
- room 5
  - Most tricks help tuning yourself but hard to do it as a group. Perhaps doing a short recording.
    - Yeah. Some independent recording/playing aronud, but also asking for advice. I would do small group/one-on-one things.
  - Background noises cancellation tools or wise choice of quiet environments
  - Audio check with colleagues or through the app with the audience saying numbers for example
  - Zoom or Teams have sound checks but work different in the app than the browser. They also work different across Operating systems
  - Some sound issues might be related to network speed or stability, also to RAM available. Worthwhile to try and control those as well
  - Make it welcoming for the audience to give you feedback on the audio and other settings, give reminders

! Finally, adjust the mix between dry and room sound, or change the frequency response in tandem using a compressor to reduce the dynamic range of a signal for a more consistent listening experience = the quietest parts of your signal will appear louder by reducing the loudest parts.

It is all about one thing and one thing only - love your voice, let people fall in love with the acoustic quality of your charm. So remember: GIGO, it matters!

Episode 5: How to prepare a quality screen-share

Materials: https://coderefinery.github.io/train-the-trainer/screenshare/

- If your Zoom does not support "share portion of screen", sharing a tab and resizing it to portrait for yourself is a possible workaround (this at least works when joining the call from Chromium browser)
  - Another workaround is to use an editor/IDE that displays multiple panels in one window and can show a web page, code files and terminal session +10
- Is 'share a portion of your screen' where you draw a box in Zoom to share a section?
  - yes :+1:
- When using zsh, one can run something like this in a terminal

  ```
  tail -n $NLASTCOMMANDS -f ~/.zsh_history | sed 's/^............../LAST COMMAND:/g'
  ```

  (I place this in a window under the main terminal, I use i3)
- I am a bit confused on what platform you are putting this code?
  - Yep, that's a problem. It can be good to keep a buffer on the bottom of the screen.
  - Radovan is using Linux with (I think) the "i3 window manager". It can be quite involved and there is lots of personal customization here.
  - If you move your mouse outside the Zoom window, the controls should go away. But still a good note.
- When teaching with a custom prompt, is it worth starting with the default (that your learners will have) and explicitly mentioning "I change these things to make it clearer when teaching"
  - Maybe. I think one can also make the zoom bars not auto-hide (then it won't display under), but that is annoying also since it wastes space.
  - One should always comment on what may be different. Thing is "default" may look different for many people :+1:
  - Warning: going to teach you how to customize your prompt, but here's an online tutorial, be prepared to lose several days of your life…" :laughing:
- Does anybody still use shellshare? Does it work?
  - I'm not going to teach you how to customize your prompt
- Does the stacking of the screens work on windows?
- Now I can't see the last line in the terminal
  - Don't know unfortunately

    - Yep, that's a problem. It can be good to keep a buffer on the bottom of the screen.
    - Radovan is using Linux with (I think) the "i3 window manager". It can be quite involved and there is lots of personal customization here.
    - If you move your mouse outside the Zoom window, the controls should go away. But still a good note.
- When teaching via a command prompt, is it worth starting with the default (that your learners will see) and explicitly mentioning "I change these things to make it clearer when teaching"
    - Yes, I agree.
    - I think one can also make the zoom bars not auto-hide (then it won't display under), but that is annoying also since it wastes space.
    - One should always comment on what may be different. Thing is "default" may look different for many people :+1:
    - Warning: Don't teach yourself how to use vim or emacs on the command prompt, but here's an online tutorial, be prepared to lose several days of your life..." :laughing:
- Does anybody still use shellshare? Does it work?
    - I'm not going to teach you how to use vim or emacs :)
- Does the stacking of the screens work on windows?
- Now I can't see the last line in the terminal
    - Don't know unfortunately

Exercises (done together) https://coderefinery.github.io/train-the-trainer/screenshare/#exercises

Write your cool tips and ideas below. Stay after xx:00, to

- To be safe, I have created on my computer an account that I use only for teaching, so I do not need to do any manual set up
- share history or jupyter notebooks live (5-10 secs refresh rate) into a git repo. Share the link with the learners so they can catch up https://github.com/mwakok/software_carpentries?tab=readme-ov-file#set-up-gitautopush

---

Wrapup

- Next session Tuesday August 27 9 CEST: About teaching & cool things we all would like to share
- All workshop materials will stay available (and be put on Zenodo right after the workshop, currently other epsisodes are still in development)
- CodeRefinery community: https://coderefinery.zulipchat.com/
- Cool gems session: If you want to share something, please edit your registration and fill it in there (link to modify in registration confirmation); or e-mail to support@coderefinery.org
- If you want to co-teach with us in the upcoming (or any future) CodeRefinery workshop; or just want to learn more about this opportunity: Let us know in chat, send an e-mail to support@coderefinery.org, or join any of the upcoming Monday planning meetings at 14:00 (Europe/Stockholm time)

Feedback about todays session

What one thing was the most valuable/useful thing you learned today? Or generally one thing you liked about this session:

- Easy to follow along and filled with useful tips on sound, video etc.
- Sound and screen share optimization as well as relevant tools and good practices for online instruction. "While you may be unable to control your users' mindset, you can anticipate it, and make sure that your user experience caters to it." - Jonathan Cherki
- Nice opportunity to (re)think about my online teaching setup :+1: :+1: :+1:
- Other ways to display command history :+1:
- Did not know that bluetooth gear had so much latency! :+1:
- A good reminder about headsets with microphones
- Nice and friendly teaching. Thank you!
- Screen share with command history and terminal customisation
- Really useful tips on sound and screen sharing :+1:
- Nice experience share
- Very useful and practical tips!

What one thing would you improve about this session?

- Just digging to find something: I have read some of the things you have talked about here already in some blog posts, so perhaps these sessions are not as shokingly full of new content as I imagined (but actually sharing experiences is something I can't do by reading a blog post)
    - Perhaps: could you link some blog posts or other sources for what you have discussed so far? If it makes sense. Not sure.
- Talking about interface optimization approaches, which applies to sound as much as to visuals or any other aspect either for that matter, it might offer a perspective to bear in

- Easy to follow along and filled with useful tips on sound, video etc.
- Sound and screen share optimization as well as relevant tools and good practices for online instruction. “While you may be unable to control your users’ mindset, you can anticipate it, and make sure that your user experience caters to it.” - Jonathan Cherki
- Nice opportunity to (re)think about my online teaching setup :+1: :+1: :+1:
- Other ways to display command history :+1:
- Did not know that Bluetooth gear had so much latency! :+1:
- A good reminder about headsets with microphones
- Nice and friendly teaching. Thank you!
- Screen share with command history and terminal customisation
- Really useful tips on sound and screen sharing :+1:
- Nice experience share
- very useful and practical tips!

What one thing would you improve about this session?

- Just digging to find something: I have read some of the things you have talked about here already in some blog posts, so perhaps these sessions are not as shokingly full of new content as I imagined (but actually sharing experiences is something I can’t do by reading a blog post)
  - Perhaps: could you link some blog posts or other sources for what you have discussed so far? If it makes sense. Not sure.
- Talking about interface optimization approaches, which applies to sound as much as to visuals or any other aspect either for that matter, it might offer a perspective to bear in mind the concept of introducing asymptotic complexity increment. The latter would mean the goal is measuring how fast the effort of engaging, on one or more of the three parallel levels of interaction, must change to reach higher levels of partial certainty on the correctness of the implementation of the method in question, in this case the pedagogical paradigm. For the former, lower is better I guess, or in this case, in the intended sense, slower is faster. Let us consider an abstract example: People liked using the search engine Google (a Google session to put it in a fancy way) in the early days because it was as simple as it could get, a literally featureless entry point with no bells and whistles. That was the level of its visible complexity, or rather Google set a bar as low as it one could have, and then any higher point of VISIBLE complexity that was to be introduced, such a move would only be on the path of an asymptotic curve curving towards the set bar, never reaching it, as far as the entry point is concerned. We still see the same level of complexity when we fire up google and we never feel distracted to start with, except for the rather forgivable nuisance of the GDPR conformity pop-up notice which appears only AFTER the first move has been made, and thankfully it works plain and simple from there on (unless one starts feeling lucky, pun intended). Now, think about Youtube … the user experience complexity is a totally different matter, the same company today (unfortunately) yet one now has a steadily worsening engagement experience. Good learning sessions tend to be more of the Google search engine experience and a far cry from the current Youtube trend I would opine. Or, to sum it up and keep it sweet and simple, it helps to identify the limits of human-human and human-technology interaction, within the framework of any applicable technology-technology interaction and joy is the result of respecting those limits. Wishful thinking ?
- Bring the cat back!
  - I wish it came, but it’s been resting all morning! I don’t bother it or stage apperances…
- Maybe having a digest/ summary of previous iterations of this session might be interesting. I feel a lot of experiences have been exchanged and a lot of techniques/ tools have been mentioned. Spending some time in compiling this into a selected list and adding it as content to the course itself might be worthwhile.
  - Good idea! We will write a blog post about this workshop and will include a summary of all discussions there :)
- Possibly have some command line examples from Windows rather than (just) *nix? I will look at the screenshots after the session. :+1: :+1:
  - Also, a thought if you do do breakout sessions with this work, do one for Unix, one for Windows, one for Mac as an option.
- I could not follow the part Radovan showed codes (Radovan is using Linux with (I think) the “i3 window manager”.)
- ..

Any other comments?

- Looking forward to the topics of following sessions.
- Looking forward to the next sessions!
- Thank you!

Relax, nothing else matters, live a life and have fun ! If you obey all the rules you may end up missing all the fun.” The best way to predict your future is to create it. - Abraham Lincoln :+1:

Day 3 : session 3 (27.08.24) - “About teaching and cool things we all would like to share”

:calendar: Schedule

Time (CEST)	Title	EEST (UTC+3)	BST (UTC+1)
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 9.45	Computational thinking	10.15 - 10.45	8.15 - 8.45

- look at the screenshots after the session. :+1: :+1:
- Relax, nothing else matters, live a life and have fun ! If you obey all the rules you may end up missing all the fun. The best way to predict your future is to create it. - Abraham Lincoln :+1:
    - Also, a thought if you do do breakout sessions with this work, do one for Unix, one for Windows, one for Mac as an option.
- I could not follow the part Radovan showed codes (Radovan is using Linux with (I think) the "i3 window manager".)
- ..

Any other comments?

- Looking forward to the topics of following sessions.
- Looking forward to the next sessions!
- Thank you!

Day 3 : session 3 (27.08.24) - "About teaching and cool things we all would like to share"

:calendar: Schedule

Time (CEST)	Title	EEST (UTC+3)	BST (UTC+1)
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 9.45	Computational thinking	10.15 - 10.45	8.15 - 8.45
9.45 - 10.00	Break	10.25 - 11.00	8.25 - 9.00
10.00 - 10.30	Teaching philosophies	11.00 - 11.30	9.00 - 9.30
10.30 - 11.00	Co-teaching	11.30 - 12.00	9.30 - 10.00
11.00 - 11.15	Break	12.00 - 12.15	10.00 - 10.15
11.15 - 11.50	Sharing teaching gems (voluntary)	12.15 - 12.50	10.15 - 10.50
11.50 - 12.00	Outro and feedback	12.50 - 13.00	10.50 - 11.00

:icecream: Icebreaker

Check-in

Which emoji best describes your current state of mind? Emoji cheat sheet

- ...
- :sleeping:
- :coffee:+2
- :robot:
- Still little sleeppy :)
- 🥴 (:woozy_face: still not converted 🤷‍♂️)
- Our planet could be a cube. It has six sides to it. It must be a cube. A cube it is, this Blue Planet ... just wondering ...
- 🦾
- :sweat_smile:
- computer says no :-) some network trouble
- running late, and participating from a :train2:

Introduction in breakoutrooms

Name / Affiliation / Location

Best classroom experiences

As a learner or teacher, what was the best workshop/lesson/seminar you have attended? What made the experience great?

- Room 1:
    - First summer HPC school
    - CSC HPC summer school
    - storytelling teaching approach
- Room 2
    - CodeRefinery CRSE course
        - very enthusiastic instructor

        - intuitive and very accessible way of showing technical details and fostering deeper understanding
        - a lot of hands-on materials and small coding exercises
    - Anything as long as it is interactive and somehow enjoyable from both teacher and learner sides
- Room 4:
    - EPCC MPI Distributed training workshop (as a teacher). Somehow the first felt the best
        - Lots of hands-on coding/experience gained.
        - Very interactive instructor who also discussed ideas for more personalised cases of distributed computing.
    - NVIDIA DLI course on Deep Learning
        - relevance of topic to my current work.
        - Dynamic / friendly / positive teaching style.

Best classroom experiences

As a learner or teacher, what was the best workshop/lesson/seminar you have attended? What made the experience great?

- intuitive and very accessible way of showing technical details and fostering deeper understanding
- a lot of hands-on materials and small coding exercises
  - Anything as long as its interactive and somehow enjoyable from both teacher and learner sides

- Room 1:
  - EPCC MPI HPC training workshop
  - CSC HPC summer school
  - storytelling teaching approach
- Room 4:
  - First summer distributed computing workshop (as a teacher). Somehow the first felt the best
    - Lots of hands-on coding / experience gained.
    - Very interactive instructor who also discussed ideas for more personalised cases of distributed computing.
- Room 2:
  - NVIDIA DLI course on Deep Learning
  - CodeRefinery CSE course
    - relevance of topic to my current work.
    - very enthusiastic instructor
    - Dynamic / friendly / positive teaching style.
    - Modular Code Development; very instructive
  - Some CSC courses / LinkedIn learning courses
    - up to date materials and topics.
    - Good way to merge theory with practicalities without making learner feel bored (Example: Jonathan Reichental)
    - Challenging information to learn more and research about the topic
    - Trainers are the experts in the field

Episode on teaching gems

If you want to share or demo something about tools and techniques you have found helpful in your teaching, please add your name/initials and rough topic below for smooth transition.

> Note: It does not have to be your tool or your idea, this episode is meant to share experiences and discover new things. You also do not need any slides or materials, a story will do fine :) Length of the stories will be depending on interest, but be prepared to fit it into 3 min (shortest possible).

- RD: teaching clock
- RB: Containerized teaching setup
  - Isolated home dir is a good idea!
- AVM: Screenkey for showing keyboard shortcuts

:question: Questions

General / Practicalities

- What is this document for?
  - Asking questions, collecting discussion points, taking collaborative notes (Zoom chat deactivated, can only be used for communication with the hosts)
- Will the material be made available?
  - Materials are and will stay available at https://coderefinery.github.io/train-the-trainer/
    - Can I share them with a colleague
      - Yes please :)
    - Can I resuse them in my own teaching?
      - Yes please :)
- Will these sessions be recorded?
  - No, but the materials (see above) should include everything needed for self-learning.
- Can I get a certificate for this workshop?
  - You can get a certificate of attendance after the workshop by sending an e-mail to support@coderefinery.org, telling us a bit about what you have learned and which sessions you attended.
- How often will we be talking in breakout rooms?
  - We'll have about one breakoutroom session per episode

Episode 1: Computational thinking

Materials: https://coderefinery.github.io/train-the-trainer/computational-thinking/ (Link to slides in the middle of material)

Questions to audience

How might breaking down a complex problem into smaller parts change your approach to problem-solving in your current projects?

- Avoids cognitive overload.
- I used to give an example about this in workshops of cutting long tree into pieces from top to avoid that it falls and destroy a house bloc.
- E...

Episode 1 : Computational thinking

Materials: https://coderefinery.github.io/train-the-trainer/computational-thinking/ (Link to slides in the middle of material)

Questions to audience

How might breaking down a complex problem into smaller parts change your approach to problem-solving in your current projects?

- Avoids cognitive overload
- I used to give an example about this in workshops of cutting long tree into pieces from top to avoid that it falls and destroy a house bloc.
- Easier to start on something small
- Each individual part is familiar and can take existing solutions.
- Only focus on the new things
- One can lose oversight of where is supposed to be getting to due to problems with summation of biases introduced during the solving of the parts.
- .

Can you think of a research project where identifying patterns in your data led to new insights or breakthroughs?

- Multi-targeted drug design where you need to identify both chemical and biological pathways/patterns
- Distilling complex physics problems into simpler 1D statistics is very often done.
- I used to do bioinformatics, pretty much everything in biology is about patterns in strings of charachters (DNA, proteins). Finding those patterns and using them is the way to go
- Pattern recognition DOES NOT lead to new insight, it is new insight which allows for the recognition of a 'pattern', the latter being an allocation of possible bias.
-
- ..

What challenges do you face when trying to simplify complex concepts in your field, and how do you decide which details to focus on?

- Absence of a terminology / jargon to describe a new idea. :+1:
- Similar to above, when teaching often students know what they want to achive, but don't have the termonology to express it, so working through what they want is helpful, after teaching them some termonology.
- Whatever is most useful to the learner first?
- I work with researchers in different fields. I don't always fully understand their domain knowledge but I can take the basiscs or generalities and get code that works for them
- A very practical problem is trying to resist going for a coffee in the midst of taking a shot at a complex concept, in the hope that the true details are to be found in the coffee.
- not all information is easy to find or a lot of conflicting information or even sometimes imformation explosion

Questions from the audience

How do you determine the priority of tasks when designing algorithms for your academic projects, and what criteria do you use to ensure that the most critical tasks are addressed first?

- Working chronologically when going through a problem - start at the beginning.
- What gives the more insight with the least effort can be a nice start
- Talk to colleagues and Subject matter experts to understand various perspective to decide and prioritize.
- Logic will get you from A to B. Imagination will take you everywhere. - Albert Einstein
-

- About the 4 core concepts, is this based on the model of how human brain works and perceives vision / abstract ideas?
- I don't know if it's based on how the brain works, but it's related to creating a framework for solving programming programs that has been extrapolated to solve any other problem as well.
- Devil's advocate question: cooking (by recipe) actually employs all four principles outlined here and has done so since before the advent of computers. Why then is this concept presented as "computational"?
  - Random aside, this is an interesting connection to our "cooking metaphor to HPC"
- Devil's answer to the Devil's advocate: Great question, cooking, as far as the act happens in the Devil's kitchen, does not employ numerical algorithms. The Devil introduced the

- Can I share them with a colleague
  - Yes please :)
- Can I reuse them in my own teaching?
  - Yes please :)
- Will these sessions be recorded?
  - No, but the materials (see above) should include everything needed for self-learning.
- Can I get a certificate for this workshop?
  - You can get a certificate of attendance after the workshop by sending an e-mail to support@coderefinery.org, telling us a bit about what you have learned and which sessions you attended.
- How often will we be talking in breakout rooms?
  - We'll have about one breakoutroom session per episode

- Not all information is easy to find or a lot of conflicting information or even sometimes information explosion

Questions from the audience

- About the 4 core concepts, is this based on the model of how human brain works and perceives vision / abstract ideas?
  - I don't know if it's based on how the brain works, but it's related to creating a framework for solving programming programs that has been extrapolated to solve any other problem as well.

How do you determine the priority of tasks when designing algorithms for your academic projects, and what criteria do you use to ensure that the most critical tasks are addressed first?

- Working chronologically when going through a problem - start at the beginning.
- Devil's advocate question: cooking (by recipe) actually employs all four principles outlined here and has done so since before the advent of computers. Why then is this concept presented as "computational"?
  - Random aside, this is an interesting connection to our "cooking metaphor to HPC"
- What gives the more insight with the least effort can be a nice start
- Talk to colleagues and Subject matter experts to understand various perspective to decide and prioritize.
- Logic will get you from A to B. Imagination will take you everywhere. - Albert Einstein.
- Devil's answer to the Devil's advocate: Great question, cooking, as far as the act happens in the Devil's kitchen, does not employ numerical algorithms. The Devil introduced the computer precisely to make man lose apetite for that what is cooked in the Devil's kitchen to be served to all. Only then can the Devil eat the cake and keep it too …
  - Well, an algorithm without numbers is still an algorithm, i.e. a set of rules :)
- Are there other names for this technique? I may have missed something crucial, but it strikes me as 'logical thinking' in a way.
  - Yes, you could also just see it as problem solving. Most people know how to do it and if you've worked with programming, you've probably applied it. It can be the case that it's very obvious to some, but not to others.
- Any chance we can ge a link to the slides?
  - They are in the course material: https://coderefinery.github.io/train-the-trainer/computational-thinking/ :thumbs_up:

Exercise questions to discuss

1. Conceptually would the strategies provided by computational thinking help you in completing tasks more efficiently?

- Room 1:
  - divide and conquer technique
  - We discussed what happens if you *don't* use these, since they are so natural
- Room 2:
  - in some cases yes, if the task is not too small
  - not sure if more efficiently but it does help at least getting strted
  - Computational thinking is an extension of what others have called solutionism: the belief that any given problem can be solved by the application of computation. Whatever the practical or social problem we face, there is an app for it. - James Bridle
- Room 3:
  - Helps in starting the task.
  - For many it is the normal way to do problem solving. Also in Agile & Scrum methodlogies.
- Room 4:
  - Yes, it helped me during my scientific research to manage multiple projects and research publications simultaneously.
  - The degree to which something, that is a result of what one means by 'Computational Thinking,' is successful in producing a desired result, that is success, cannot be a function of personality.

2. Do you think that the effectiveness of computational thinking depends on the person's personality type?

- Room 3:
  - Yes. Different people have different ways to communicate.
  - Online tools vs pen and paper
- Room 1:
- Room 4:
  - yes, we agree it's quite different. But good vision and motivation this can be effective
  - We are all agree that it depends on the personality type but it could be learned and trained.
- Room 2:
  - yes, laying out all the steps necessary to achieve an overall goal can be very daunting; some people are not good at/ comfortable with following fine-grained instructions
  - once the problem is broken down, different personalities might choose different ways to tackle the parts

3. Would this type of thinking be an option for you to integrate into your current workflow?

- Room 1:

research publications simultaneously.
    - The degree to which something, that is a result of what one means by ‘Computational Thinking,’ is successful in producing a desired result, that is success, cannot be a function of personality.
2. Do you think that the effectiveness of computational thinking depends on the person’s personality type?
    - Yes. Different people have different ways to communicate.
    - Online tools vs pen and paper
- Room 1:
- Room 4:
    - yes, we agree it’s quite different. But good vision and motivation this can be effective
    - We are all agree that it depends on the personality type but it could be learned and trained.
- Room 2:
    - yes, laying out all the steps necessary to achieve an overall goal can be very daunting; some people are not good at/ comfortable with following fine-grained instructions
    - once the problem is broken down, different personalities might choose different ways to tackle the parts
3. Would this type of thinking be an option for you to integrate into your current workflow?
- Room 1:
    - Yes; we didn’t know the name explicitly; but we use it
    - It’s quite important for
- Room 2:
    - I’m already doing it implicitly :smile: :+1: :+1:
    - Writing a program is like writing a song, its all about the tune, lyrics are merely the necessary nuisance a coder needs to put up with. The latter can, if so written, result in a lot of noise.
- Room 3:
    - Easy if you are working alone. But different team may have their own ways of working (inertia). One way would be then to gradually show the rest of the team its effectiveness.
    - https://teamtopologies.com/key-concepts
- Room 4:
    - We are using it in our daily life so it’s already an option for us, only the terminologies are a bit new.

Episode 2 : Teaching philosophies

Materials: https://coderefinery.github.io/train-the-trainer/teaching-philosophies/

Questions to discuss:

- Share your approach to teaching and your teaching philosophy with your group.
- Please share your tricks and solutions in the live document for others.

Additional ice-breaker questions:

- What is your motivation for taking this training?
- How structured or informal are your own teaching needs?
- What difference do you notice between the teaching what we (also Carpentries) do and traditional academic teaching?
- What other skills need to be taught, but academic teaching isn’t the right setting?

Breakout Room exercise

- Room1
    - Using Jupyter notebooks - good for learning a technique, but not for learning the basics.
    - Possibly command line usage has a branding problem - “programming other programs”?
    - Motivation
        - For best practices in teaching, learning from each other, collaboration
    - the skills we learn in CR training are expected to stay , it is not exam oriented
    - Learning applied tech is like an apprecentiship: there is theory, but real learning happens by co-working with others.
- Room 2
    - State objectives clearly and repeat them a few times
    - Worse is better: teach what can be used more quickly and it can be improved later
    - Try to keep it informal, interactive, flexible
    -
    - Helps when having practical tasks in mind for the materials taught
    - I think there are two bits of teaching - teach the basics and terminology so people know what they want to know, and then teaching how to find out how to do things
        - How to deal with frustration though: try some personal approaches, more clues… like Richard said - how to find answers yourself.
    - Leave some reasoning to the learner, not always giving full answers
    - The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. — William Arthur Ward

Breakout Room exercise

- Room1
  - Possibly command line usage has a branding problem - “programming other programs”?
  - the skills we learn in CR training are expected to stay , it is not exam oriented
- Room 2
  - State objectives clearly and repeat them a few times
  - Try to keep it informal, interactive, flexible
  - Helps when having practical tasks in mind for the materials taught
  - Leave some reasoning to the learner, not always giving full answers
  - The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. — William Arthur Ward
  - Teaching kids to count is fine, but teaching them what counts is best. –Bob Talbert
- Room 4
  - Question 1: What is your motivation for taking this training?
    - Currently do instruction, wanting to improve as an instructor and helper when conducting software carpentries.
    - Planning to start some side hustles providing online courses and why not joining the CR in the near future
  - Question 2: How structured or informal are your own teaching needs?
    - More recently, follow software carpentries, but look into what aspects can be tailored to keep to different time slots.
    - I used to be more informal in the past, repetetive practice sessions. Would like to be more structured given the teaching tools.
    - I’ve done life-coaching and some teaching but I feel i’m both of them
  - Question 3: What difference do you notice between the teaching what we (also Carpentries) do and traditional academic teaching?
    - Smaller size, and more targetted.
    - Collaborative tools and targeted audience working with small groups
    - Academic probably focuses on an audience with a more focused/narrow background, while this style encompasses a wider audience.
    - Academic teaching keeps theory and practical lab sessions very separate, whereas carpentries combine both.
  - Question 4: What other skills need to be taught, but academic teaching isn’t the right setting?
    - Public speaking, collaborative meeting skills
    - Industrial cases and related work skills

Episode 3: Co-teaching

Materials: https://coderefinery.github.io/train-the-trainer/co-teaching/

- In general: Co-teaching is a beast with at least two personalities each of which
  - demand higher preparation requirements
  - face remarkable complexity in terms of coteacher coordination
  - in the virtual component face shortcomings related to the technical equipment and the failure of the human factor

Exercise/Discussion: https://coderefinery.github.io/train-the-trainer/co-teaching/#exercise

Questions:

- Have you already tried this or similar model in your teaching?
- Does it seem natural to apply this model in your subject area (tell what it is)? How could it be adapted to fit best?
- How do the downsides compare to a single instructor monologuing?
- Have you tried or seen a different model for two instructors to present? Please share with us how it works.

Breakout Room exercise

- Room1
  - Motivation
    - For best practices in teaching, learning from each other, collaboration
  - Learning applied tech is like an apprecentiship: there is theory, but real learning happens by co-working with others
  - Worse is better: teach what can be used more quickly and it can be improved later
  - I think there are two bits of teaching - teach the basics and terminology so people know what they want to know, and then teaching how to find out how to do things
    - How to deal with frustration though: try some personal approaches, more clues… like Richard said - how to find answers yourself.

- Using Jupyter notebooks - good for learning a technique, but not for learning the basics.

- Co-teaching requires the framework of co-operative learning. Team teaching does not by default translate into parallel learning, and vice versa.

- Re: CodeRefinery in-person beginnings – I agree with the “didn’t work so well in-person”, I’m not sure if it’s that we weren’t as ready for it, or that the different characteristics made it not work so much
- Have you tried or seen a different model for two instructors to present? Please share with us how it works.
  - That’s a great question! The main difference is that a single instructor can teach himself to be good (i.e. not just boring monologuing) **alone**, while for teams of 2+ to work well, **everyone** needs to be baseline-competent. So the effort to avoid

- room 1

    - In the virtual component face shortcomings related to the technical equipment and downsides gets reduced if (and only if) all team members share it.
    the failure of the human factor

Exercise/Discussion: https://coderefinery.github.io/train-the-trainer/co-teaching/#exercise

- Co-teaching requires the framework of co-operative learning. Team teaching does not by default translate into parallel learning, and vice versa.

Questions:

- Re: CodeRefinery in-person beginnings – I agree with the "didn't work so well in-person", I'm not sure if it's that we weren't as ready for it, or that the different characteristics made it not work so much
- Have you already tried this or similar model in your teaching?
- Does it seem natural to apply this model in your subject area (tell what it is)? How could it be adapted to fit best?
- How do the downsides compare to a single instructor monologuing?
- Have you tried or seen a different model for two instructors to present? Please share with us how it works.
    - That's a great question! The main difference is that a single instructor can teach himself to be good (i.e. not just boring monologuing) **alone**, while for teams of 2+ to work well, **everyone** needs to be baseline-competent. So the effort to avoid
- room 1
    - We have tendend to do presenter/interviewer more
    - challenges are co-teaching with different personalities
- room 2
    - Main challenge: finding enough personell :smile:
    - It's not always possible/feasible to have colleagues co-teaching
    - A combination of both, but presenter and interviewer could be difficult to plan an to follow
    - Guide and demo-giver with technical / non-technical roles could be nice
        - How to plan the guide and demo-giver
            - First guide, then demo
            - Demo, then guide describes
    - Alternative model: main presenter + 'technical expert'
        - main presenter introduces concepts from a birdseye view and passes on to the technical expert for additional details, experiences and answers to questions from the audience
    - We frequently do a classical division: on person in presenting, one or more are answering questions in the chat; there is rarely any interaction between them however

- As has been pointed out by Paolo Freire, in general, 'education is suffering from narration sickness' … 'The outstanding characteristic of this narrative education, then, is the sonority of words, not their transforming power'. Perhaps one approach could be to rephrase the question after having the answer: It is not apriori about the model implemented, it is about whether that what was implemented was judged aposteriori to have succeeded in becoming, or at least could pretend to have done so, the intended context matched model implementation it probably was computed to be.

- room 4
    - the co-teaching approach does not seem to be widely practiced
    - it can be interesting to see "mistakes" which are less likely to happen in solo-teaching?
    - co-teaching can be nicer for audience but maybe more stress/anxiety for instructors to prepare? more stress in terms of not knowing all the questions in advance. less stress worrying about forgetting something

Teaching gems

(scroll up to find the green box "Episode on teaching gems")

What's your favorite way of teaching? Any nice tools you use?

- I think the Code Refineries use something for their graphics that looks a bit 'cartoony' - but I cannot remember what it is called!
    - Drawn on remarkable and then coloured and tidied up in Inkscape. :+1:
- I am a big fan of remarkable. I think sharing option is fairly good. Unfortunately bit expensive tool.
    - true, for me it was worth it already for the pdf annotation for research papers, cannot read on computer screen

Has anyone found a nice online tool to draw?

- https://webwhiteboard.com/
- https://www.youtube.com/watch?v=4-l8MY5kYGc
- I love this tool suite https://excalideck.com/
- Figma and Kahoot are useful as well
    - For drawing https://excalidraw.com/
- I have seen Miro …

Screenkey: Screenkey for showing keyboard shortcuts

stress worrying about forgetting something.

Teaching gems

(scroll up to find the green box "Episode on teaching gems")

What's your favorite way of teaching? Any nice tools you use?

Has anyone found a nice online tool to draw?

Screenkey: Screenkey for showing keyboard shortcuts

- Is there a Windows version people can recommend?
  - I found this list of alternative for windows users like me: https://alternativeto.net/software/screenkey/ :+1:
- I sometimes use the on-screen keyboard already provided by the OS :+1:

Wrapup

- Next session Tuesday September 3rd 9 CEST: About teaching & cool things we all would like to share
- All workshop materials will stay available (and be put on Zenodo right after the workshop, currently other epsisodes are still in development)
- CodeRefinery community: https://coderefinery.zulipchat.com/
- If you want to co-teach with us in the upcoming (or any future) CodeRefinery workshop; or just want to learn more about this opportunity: Let us know in chat, send an e-mail to support@coderefinery.org, or join any of the upcoming Monday planning meetings at 14 CEST
- Preparations for next week will be sent out by e-mail

Feedback about todays session

What one thing was the most valuable/useful thing you learned today? Or generally one thing you liked about this session:

- The co-teaching lesson and experiences from others
- Discussions were quite good, had a nice group
- Breakout rooms and collaborative tools
- Computational thinking and co-teaching are interesting ideas! I have seen it before, but never knew or even recognized it was a method.
- Thanks for group discussions working with my chat only interaction! :smile:
- ..

What one thing would you improve about this session?

- Minor point - it would be useful to have the teaching gmes box at the bottom of the document rather than the top - tricky to scroll between! In the end I opened two windows which was easier.
  - Thanks, yes agree. Also had that problem, but did not want to move it in between.
- The meaning of teaching "philosophy" wasn't very clear, felt more like teaching "styles"
  - Thank you, we will take that into consideration
- Some prepared slides to present for an intro about the topics
  - You mean in addition to the materials? To use in the beginning of each session?
- The *instructor views* segment could be named as additional material, if it was not meant to be discussed within the course.
  - Thank you, we will take that into consideration

Any other comments?

- I think the Code Refineries use something for their graphics that looks a bit 'cartoony' - but I cannot remember what it is called!
  - Drawn on remarkable and then coloured and tidied up in Inkscape. :+1:
- I am a big fan of remarkable. I think sharing option is fairly good. Unfortunately bit expensive tool.
  - true, for me it was worth it already for the pdf annotation for research papers, cannot read on computer screen
- https://webwhiteboard.com/
- https://www.youtube.com/watch?v=4-l8MY5kYGc
- I love this tool suite https://excalideck.com/
- Figma and Kahoot are useful as well
  - For drawing https://excalidraw.com/
- I have seen Miro ...

Day4: Session 4 (3.09.24) - Streaming and video editing

:calendar: Schedule

Time (CEST)	Title	EEST (UTC+3)	BST (UTC+1)
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 10.00	Why we stream & Behind the scenes	10.15 - 11.00	8.15 - 9.00

- Minor point - it would be useful to have the teaching gmes pox at the bottom of the document rather than the top - tricky to scroll between! In the end I opened two windows which was easier.
  - Thanks, yes agree. Also had that problem, but did not want to move it in between.
- The meaning of teaching "philosophy" wasn't very clear, felt more like teaching "styles"
  - Thank you, we will take that into consideration
- Some prepared slides to present for an intro about the topics
  - You mean in addition to the materials? To use in the beginning of each session?
- The *instructor views* segment could be named as additional material, if it was not meant to be discussed within the course.
  - Thank you, we will take that into consideration

Any other comments?

Day4. Session 4 (3.09.24) - Streaming and video editing

:calendar: Schedule

Time (CEST)	Title	EEST (UTC+3)	BST (UTC+1)
8.45 - 9.00	Connecting time	9.45 - 10.00	7.45 - 8.00
9.00 - 9.15	Intro and Icebreaker	10.00 - 10.15	8.00 - 8.15
9.15 - 10.00	Why we stream & Behind the scenes	10.15 - 11.00	8.15 - 9.00
10.00 - 10.15	Break	11.00 - 11.15	9.00 - 9.15
10.15 - 10.35	Video Editing	11.15 - 11.35	9.15 - 9.35
10.35 - 10.55	Exercise: Video Editing	11.35 - 11.55	9.35 - 9.55
10.55 - 11.10	Break	11.55 - 12.10	9.55 - 10.10
11.10 - 11.30	Open Broadcaster Software(OBS) introduction	12.10 - 12.30	10.10 - 10.30
11:30 - 11:50	OBS setup & what next	12.30 - 12.50	10.30 - 10.50
11.50 - 12.00	Outro and feedback	12.50 - 13.00	10.50 - 11.00

:icecream: Icebreaker

Check-in

Which emoji best describes your current state of mind? Emoji cheat sheet

- :coffee: +
- :runner:
- :tired_face: Tired!
- :nerd_face:
- 
- :cloud: :tea:

Introduction in breakoutrooms

- Name / Affiliation / Location

What's the most number of people you have taught to?

- 5 or so
- 2 or 3 in total
- ~35
- 7
- ~80 in a lecture in a previous (lecturer) role, 26 in a training course setting
- 2 or 3
- ~30
- 3 (helpers, not really co-teaching)
- ~50 in training spaces and more than 100 for public audience talk
- CodeRefinery answer: our biggest workshops are something like 10-20 people helping out

What's the most number of people you have taught with?

- 35 (software carpentry)
- CodeRefinery answer: The biggest ones are maybe 200-300 people. "small" for a stream is ~100 people.
- 20 at a time, with 1 asleep, 2 fidgiting with some silly device and the rest were... well... staring at their good luck... straight ahead it was of course, right before them.
- 3, as in three, none of them smiling. I wish I could figure out why... it was such fun really...

- 35 (software carpentry)

Introduction in breakoutrooms

- CodeRefinery answer: The biggest ones are maybe 200-300 people. “small” for a stream is ~100 people.
- Name / Affiliation / Location

What’s the most number of people you have taught with?
What’s the most number of people you have taught to?

- 5 or so
- 2 or 3 in total
- ~35
- 7
- ~80 in a lecture in a previous (lecturer) role, 26 in a training course setting
- 2 or 3
- ≈30
- 3 (helpers, not really co-teaching)
- ~50 in training spaces and more than 100 for public audience talk
- CodeRefinery answer: our biggest workshops are something like 10-20 people helping out in many different roles.

How can you divide teaching into separate independent tasks?

- By content blocks, having roles like main and assistant…
- Having responsibility for different sections of the course.
- Wow, now thats what we call a question… bravo… indeed, how does one separate sleeping and snoring into separate independent tasks…
- Course, exercices, resources, tools and forms.
- Different people teach different topics
- CodeRefinery answer: the big logical blocks are instructors, in-person and breakout room helpers, and Notes-questions answers.

What’s the most interesting or useful thing you’ve learned from an online workshop, and how have you applied it/planning to apply in your life or work

- Having breaks every hour or so
- New approach to screensharing, using the ‘portrait’ approach
- Manage breathing: reduce stress and use silence to let the audience grasp what you’re saying
- Well, the most interesting or useful thing I’ve learned from online workshops is how not to conduct a workshop, or at least some elements of the same, and I have applied this understanding by engaging in the global effort to discover more about how not to conduct a workshop, or at least some elements of the same, with the motto: That’s one small step for a man, one giant leap for mankind..

:question: Questions

Why we stream

- If during streaming there is no interaction between the teacher and the audience, why don’t we just record the lectures and stream them? So one can do a better job, perhaps?
  - Streaming it making it a “thing” that’s a group experience. It feels good to be a part of it, so we hope that encourages people to actually attend. All the mass Q&A is fun, and the audience helps to make it easier to present. Still, this is a good question.
  - I’m besides chat and things that happen in chat, I’m not sure. The Notes-doc is definitely unique to us.
  - I really appriciate it being “a thing”. If it was just recorded, I would say I would watch it tomorrow, and then I never would.
    - But then you can just fake it, and still have all the breakout rooms and collaborative document and so on. I appreciate the value of having a proper “event”
    - I think I follow you, but if you pre-recorded the talking bits, then the presentors couldn’t change what they do based on the feedback in the shared document.
    - true, but I assumed that the teachers were too busy to actually do that. The more (less)division of labor between the teachers and the helpers (or collaborative document editors), the more (less) sense it makes to pre-record - IMHO
- What interaction options do ‘streamers’ use, e.g. on Twitch/Youtube (not Code Refinary)?
- Streaming is a kind of exercise for the movie star type. If you cannot be one, act like one, pretend, fool the listener into thinking you are one, take their brains and emotions under control. And if they don’t leave the session with tears of love for learning running down their chubby cheeks you know who to kick for doing a job just that bit worse than it should have been necessary.
- Have you kept track of streming stats? like how many people logged in and stayed troughout, how many interacted?
  - Yes please, it would be good to see the stats
  - Here is stats repo
  - Thanks, I get a 404 error, is it private?

- What interaction options do 'streamers' use, e.g. on Twitch/Youtube (not Code Refinary)?
  - I'm besides chat and things that happen in chat, I'm not sure. The Notes-doc is definitely unique to us.
  - I really appriciate it being "a thing". If it was just recorded, I would say I would watch it tomorrow, and then I never would.
- Streaming is a kind of exercise for the movie star type. If you cannot be one, act like one, pretend, fool the listener into thinking you are one, take their brains and emotions under control. And if they don't leave the session with tears of love for learning running down their chubby cheeks you know who to kick for doing a job just that bit worse than it should have been necessary.
- Have you kept track of streming stats? like how many people logged in and stayed troughout, how many interacted?
  - Yes please, it would be good to see the stats
  - Here is stats repo
  - Thanks, I get a 404 error, is it private?
  - No, I don't think so. https://github.com/coderefinery/workshop-stats
  - Still get 404. Does it work for anyone else?
    - Yeah I think it's private. We need to fix this...
- How big is small (10-15), medium (~30-40)?
  - I would say below 50 is small.
- Have you considered scaling to more events instead of to a larger audience? I mean, this seems the opposite of scaling, but this kind of content would be more useful the more often is it taught ("I need to learn GIT/Python/* in December!")
  - We would, but right now we simply don't have enough time and people to do more small events. But yes, "scheduling conflict" is a big problem - the best we can do right now is written material+videos to follow up on (and hope it's interesting enough)
  - Python for Scientific Computing 5-7/November/2024
    - This was just an example. I mean, if I am a PHD student that needs to start working on a topic on Month X which happens to be inconveniently placed in the year, I will not benefit as much from having a workshop at Month X+3
      - The materials are open and the videos are publically available on YouTube channel
        - What's the ratio of people watching videos afterwards vs people attending the event? That ratio can give an idea of the popularity of the two possible approaches.
  - The more you scale up the less interactive the course / workshop will need to be. It then becomes a question of priorities, and if teaching is a lower priority then learning than I guess there really should be no prizes for guessing which approach should reign the airwaves.
- I guess outside Europe / Africa, time zones could be a problem?
  - yes, that is true
  -
- Sorry, I might have missed this: have you been rehearsing and/or doing dry runs?
  - With instructors I like to try to do a dry run. I often don't do a live broadcast dry run these days since I'm confidente enough, but when you are just starting, it's a good idea.
- Do you always use Zoom for the 'presenter end' or have you tried other things? Teams? Jitsi? etc.?
  - we use mostly zoom
  - It was first pioneered using Jitsi. You can probably use others, too.
  - See the hardware notesrecommendation here
- A comment: for deRSE24 we have been using the GWDG streaming service and OBS, it worked quite well (with a 20s delay or so)
- Do you do a screenshare from Zoom -> OBS or OBS -> Zoom?
  - Instructors share to Zoom, OBS captures Zoom, OBS sends to the world.
  - When I am using OBS for a single-person presentation recording, I go the other way: OBS captures my desktop/camera, OBS generates the preview window, and then Zoom captures and broadcasts the preview window.

Behind the stream

Video editing

- Does OBS take a lot of RAM or CPU? what are the specs on your machine?
  - Richard have a relatively powerful computer with 8 AMD CPUs.
    - 4 CPUs are relatively enough for a good OBS setup, most important is a stable internet connection (ideally wired).
    - An extra monitor is recommended for the setup
    - Cool! Yes, there is plenty of livestreaming options. Twitch+our OBS config gives us 2-3s of latency (which is low enough for live Notes interaction). You might be able to tune this if needed.

Poll

-

Jitsi, etc.)
    - 4 CPUs are relatively enough for a good OBS setup, most important is a stable internet connection (ideally wired).
    - An extra monitor is recommended for the setup
    - See the hardware notesrecommendation here
- A comment: for deRSE24 we have been using the GWDG streaming service and OBS, it worked quite well (with a 20s delay or so)
- Do you do a screenshare from Zoom -> OBS or OBS -> Zoom?
    - Instructors share to Zoom, OBS captures Zoom, OBS sends to the world.
    - When I am using OBS sfor a single-person presentation recording, I go the other way: OBS captures my desktop/camera, OBS generates the preview window, and then Zoom captures and broadcasts the preview window.

Behind the stream

Video editing

- Does OBS take a lot of RAM or CPU? what are the specs on your machine?
    - Richard have a relatively powerful computer with 8 AMD CPUs.

Poll

- Have you tried video editing? Add an `o` as answer below
    - yes: ooooo
    - no :
    - Video editing makes all the difference. I have never known a way to avoid this sacred of steps, so to speak. In short, no recording can ever be production ready by default in the books of the wiser among the streamers. In fact, during the editing one can considerably refine and improve the learning experience in great ways. But then one would need to invest both time and effort.
- If `yes` Which are the tools you use?
    - Windows Movie Maker / ClipChamp
    - (raw stream from Zoom)
    - I think it was shotcut - I did only basic trimming
    - Kdenlive
    - Clipchamp, OpenShot, Sony Vegas
    - also ffmpeg from command line to cut beginning and end

Questions

- What is Whisper?
    - Open-source model from OpenAI that converts speech to text transcripts.
    - this one? https://openai.com/index/whisper/
    - curious if anyone has tried it with languages other than english
    - I heard that one of my spanish colleague try it with another software. I don't remember the tool. But with AI bloom there are many licensed tools that can do it with fifferent language
- What do you use to crop the videos?
    - I see now, ffmpeg
- how to merge multiple parts in one? I asked because you edited only the icebreaker part?
    - you can manage it in input part, see ffmpeg-editlist readme.
- Do you generate custom thumbnails for the video?
    - no, because of the time, but its good to have. If we had a volunteer to manage that, then we should!
- Does anyone recommend any other (GUI) tools that work well? ffmpeg is very new to me!
    - if you want a basic trimming and editing , you can use Quicktime player/imovies on mac
    - Clipchamp as the Microsoft video editor

Open Broadcaster Software (OBS) introduction & setup

- Are the profiles public?
    - Yes, you can see it here
    - Thanks :smile:
- Does it work in Linux+Wayland?
- How do subtitles get uploaded to YouTube?
    - I think everything we do with OBS would.
    - You upload the video and subtitles as part of upload.
    - :+1:
- How does codewhisperer compare against youtube's automatically generated subtitles?
    - by experience, whisper was better couple of years ago. We haven't compared it recently
    - I guess and advantage is that you can edit whisper subtitles in case something is odd or whatever, probs cannot do that in YouTube

Wrapup

- All workshop materials will stay available (and be put on Zenodo right after the workshop)
- Upcoming Workshops
    - CodeRefinery workshop September 10-12, 17-19, 2024
    - Build Systems Course and Hackathon
    - Python for Scientific Computing

- Does anyone recommend any other (GUI) tools that work well? ffmpeg is very new to me!
  - if you want a basic trimming and editing , you can use Quicktime player/imovies on mac
- Are the profiles public?
  - Clipchamp as the Microsoft video editor
  - Yes, you can see it here
  - Thanks :smile:
- Does it work in Linux+Wayland?
- How do subtitles get uploaded to YouTube?
  - I think everything we do with OBS would.
  - You upload the video and subtitles as part of upload.
  - :+1:
- How does codewhisperer compare against youtube's automatically generated subtitles?
  - by experience, whisper was better couple of years ago. We haven't compared it recently
  - I guess and advantage is that you can edit whisper subtitles in case something is odd or whatever, probs cannot do that in YouTube

Wrapup

- All workshop materials will stay available (and be put on Zenodo right after the workshop)
- Upcoming Workshops
  - CodeRefinery workshop September 10-12, 17-19, 2024
  - Build Systems Course and Hackathon
  - Python for Scientific Computing
- CodeRefinery community: https://coderefinery.zulipchat.com/
- If you want to co-teach with us in the upcoming (or any future) CodeRefinery workshop; or just want to learn more about this opportunity: Let us know in chat, send an e-mail to support@coderefinery.org
- A summary email will be sent out to all participants

Feedback about todays session

What one thing was the most valuable/useful thing you learned today? Or generally one thing you liked about this session:

- ffmpeg seems great, will give a try for fun!
- great to see everything in practice!
- Very insightful to see what happens behind the scenes.

What one thing would you improve about this session?

- Some more interactivity would've been nice but since most the group preferred demo instead of exercise perhaps it's only a personal opinion
- It would have been good to share the OBS profiles repo beforehand in the setup email - so I could have had it to hand. (https://github.com/coderefinery/obs-config)
  - +1 good idea, we should do this next time
- Perhaps a simpler starter OBS setup?
  - Good idea... I should have, but basically ran out of time to prepare.

Any other comments?

- Cool to see the switch from RSE to trainer to AV guy, impressive!

Instructor guide

Target audience

- Everyone teaching online workshops about computational topics or interested in teaching
- Previous and future instructors and helpers of CodeRefinery workshops.

Timing

Session 1

- 15 min: Intro
- 10 min: About the CodeRefinery project and CodeRefinery workshops in general
- 30 min: Collaborative notes and interaction
- 15 min: Break
- 45 min: Lesson design and development
- 15 min: Break
- 45 min: Lesson template
- 15 min: Break

Session 2

- 20 min: How we collect feedback and how we improve
- 10 min: Outro and feedback

- 30 min: How we collect feedback and how it is important
- 10 min: Outro and feedback
- Previous and future instructors and helpers of CodeRefinery workshops.

Session 2

Timing

- 15 min: Intro
- 10 min: About the CodeRefinery project and CodeRefinery workshops in general
- 30 min: Collaborative notes and interaction
- 15 min: Break
- 25 min: Workshop overview, roles, onboarding/installation, helpers
- 20 min: Sound
- 15 min: Break
- 30 min: Screenshare
- 10 min: Outro and feedback

Session 3

- 15 min: Intro
- 30 min: Computational thinking
- 15 min: Break
- 30 min: Teaching philosohies
- 30 min: Co-teaching
- 15 min: Break
- 35 min: Sharing teaching tips, tricks, tools etc
- 10 min: Outro and feedback

Session 4

- 15 min: Intro
- 10 min: Why we stream
- 20 min: Behind the stream
- 15 min: Video editing (part 1)
- 15 min: Break
- 25 min: Video editing (part 2, exercise)
- 20 min: OBS introduction
- 15 min: Break
- 30 min: OBS setup
- 15 min: What's next?

Talking points for each sessions intro

- Brief CodeRefinery intro
- Instructors intros
- Notes intro
  - Check-in
  - Icebreaker
  - Participant intros in breakoutrooms (Random assign first, then same for whole session)
- Daily schedule, learning objectives
- Code of conduct
- Chat (please use collaborative notes)

Participant preparations for each session

Session 2: In general you will only need your brain for the exercises and discussions. To get the most out of this workshop day we recommend that you join the session with the same computer, display and microphone setup as you would usually use for teaching/ presenting. It may also help to get familiar with the different options your computer and Zoom setup provides for screenshare and audio adjustments. But it is not a must.

Session 1: In general you will only need your brain for the exercises and discussions, but some will have the option to go deeper, which may require some accounts or tools to be set-up. We will inform about these things the week before each session.

For Session 1 we recommend to have a GitHub account (https://coderefinery.github.io/installation/github/#github). There will be an optional exercise where you can build a lesson locally. This requires a basic Python environment (https://github.com/coderefinery/sphinx-lesson-template/blob/main/requirements.txt), but do not worry if you cannot or do not want to set up these, there will be plenty of other exercises.

- Participant intros in breakoutrooms (Random assign first, then same for whole session)
- Daily schedule, learning objectives
- Code of conduct
- Chat (please use collaborative notes)

Copied from e-mail communication with participants before each session

Session 1: In general you will only need your brain for the exercises and discussions, but some will have the option to go deeper, which may require some accounts or tools to be set-up. We will inform about these things the week before each session.

For Session 1 we recommend to have a GitHub account (https://coderefinery.github.io/installation/github/#github). There will be an optional exercise where you can build a lesson locally. This requires a basic Python environment (https://github.com/coderefinery/sphinx-lesson-template/blob/main/requirements.txt), but do not worry if you cannot or do not want to set up these, there will be plenty of other exercises.

Participant preparations for each session

Session 2: In general you will only need your brain for the exercises and discussions. To get the most out of this workshop day we recommend that you join the session with the same computer, display and microphone setup as you would usually use for teaching/ presenting. It may also help to get familiar with the different options your computer and Zoom setup provides for screenshare and audio adjustments. But it is not a must.

Session 3: In general you will only need your brain for the exercises and discussions. If you want to share some “cool gem” in the last episode of this workshop day, please update your registration (link to edit can be found in the registration confirmation e-mail sent by support@coderefinery.org or noreply@neic.org). This is very low barrier, you do not have to be the developer of a tool or technique to share with the group what makes it useful for you; and it does not even have to be cool, sometimes the small “normal” things are the best. You can check out some topics that have been proposed so far in this GitHub issue (#90), you can also comment there if you want to add or change your topic. These will be lightning talk length, so something between 2-5 min depending on how many are interested to share something, but we can collect them all in the collaborative notes for further exploration/reading. But it is not a must.

Session 4: This upcoming session (session number 4) will cover the technical aspects of CodeRefinery-style online teaching. These topics have never before been covered in this much detail, so if it’s interesting to you, don’t miss it. (You might also want to pass this on to others who are interested in technical production of online events; registration is still open!).

First, we talk about why we stream (how we got here, advantages, disadvantages, how to be an instructor in a livestreamed workshop, and what it looks like from a broad level from the production side)

Then, we show how we are able to release videos so quickly. This is useful far beyond CodeRefinery, and there are also some hands-on exercises. Come prepared to set up a Python environment and the ffmpeg command line tool if you want to try these.

Finally, we see details about how to set up Zoom→Livestream environment for yourself. There isn’t time to do this as a proper exercise, but it’s the starting point and the instructor will offer help after the course for those interested. If you want to try this out during the session, install OBS Studio in advance.