BRIAN DAY 2 MORNING

Welcome Back

Overview

Teaching: 5 min
Exercises: 5 min
Questions

- What have we learned so far?
- What will we focus on today?

Objectives

- Review main points we discussed yesterday.
- Introduce topics we will discuss today.

Yesterday we focused on general aspects of educational psychology and pedagogy. Today we will focus on the specifics of teaching Software Carpentry, Data Carpentry, and Library Carpentry workshops. We will look at the lessons that have been written by the Carpentries and discuss how to teach effective workshops. We will review specific teaching practices we follow at our workshops and practice using some of them. One of the most important practices is participatory live coding. You rarely find this approach in a current university setting so we'll spend some time practicing this skill. We will conclude our training with a discussion about workshop logistics that will help you in preparing to teach your first workshop.

Questions

> Yesterday we asked you to read some resources about the logistics of teaching and running Carpentries workshops. Please add your questions about logistics and preparation to the Etherpad. We will answer these questions in the Etherpad during your work time and will return to this list later today.
>
> This activity should take about 5 minutes.

Key Points

- Instructors guide learners to construct the proper big picture (accurate mental model) of the topic rather than focus on details.

- Instructors rely on frequent feedback from learners to monitor their own presentation of the material.
- Instructors introduce a few concepts at a time to avoid cognitive overload.
- The best way to motivate learners? Show them how to do something they can immediately put to use and be enthusiastic about it.
- Teaching is a learned skill.

Live Coding is a Skill

Overview

Teaching: 20 min
Exercises: 50 min
Questions

- Why do we teach programming using participatory live coding?

Objectives

- Explain the advantages and limitations of participatory live coding.
- Summarize the key dos and don’ts of participatory live coding.
- Demonstrate participatory live coding.

One of the cornerstones of the Carpentries teaching is live coding: *instructors don’t use slides*, but work through the lesson material, typing in the code or instructions, with the workshop participants following along. This section explains how it works, why we use it, and gives general tips for an effective participatory live coding presentation.

Why Participatory Live Coding?

We do not use slides in our lessons. Instead, instructors plug their laptop into the projector and work through the lesson, typing in the code, reformatting data, and talking as we go. This is called “live coding”. However, the instructor is not live coding in a vacuum. Importantly, learners are strongly encouraged to “code-along” with the instructor. We refer to the practice of having the instructor live code and the learners code along as “participatory live coding” or, less formally, ‘code-along sessions’.

Up and Down

> List some advantages and challenges of participatory live coding from both a learner’s and an instructor’s point of view in the Etherpad.

This discussion should take about 10 minutes.

Solution

Some advantages are:

- Watching a program being written is more compelling than watching someone page through slides that present bits and pieces of the same code.
- It enables instructors to be more responsive to "what if?" questions. Where a slide deck is like a railway track, participatory live coding allows instructors to go off-road and follow their learners' interests.
- Lateral knowledge transfer: participatory live coding facilitates the transfer of tacit knowledge – people learn more than we realized we were teaching by watching *how* instructors do things.
- It slows the instructor down: if she has to type in the program as she goes along, she can only go twice as fast as her learners, rather than ten-fold faster as she could with slides.
- Learners get to see instructors' mistakes *and how to diagnose and correct them*. Novices are going to spend most of their time doing this, but it's left out of most textbooks.

Some challenges are:

- It requires instructors to be able to improvise when things go wrong or when learners have questions not directly addressed in the text of the lesson.
- It can be hard for learners to listen and type at the same time, due to the *split-attention effect* we discussed earlier. This is why it's very important that instructors first explain what they're going to do, then say what they are typing as they type it, and then explain what they did again afterwards.
- It may take a bit of practice for instructors to get used to thinking aloud while coding in front of an audience.

Live coding fits well into the practice-feedback model we've been discussing - by providing learners with continuous opportunities for practice (every time they type in a line of code) and continuous feedback (their code either works or fails with an error message). It's important to keep in mind, however, that feedback isn't helpful if you can't understand it. Many error messages are obscure and not written with novices in mind. Continue to use the strategies for error framing that we learned earlier to make sure this feedback is useful to learners.

Compare and Contrast

Watch this first participatory live coding demo video and this second demo video as a group and then summarize your feedback on both in the Etherpad. Use the 2x2 rubric for feedback we discussed earlier.

In the videos, the bash shell `for` loop is taught, and it is assumed learners are familiar with how to use a variable, the `head` command and the content of the `basilisk.dat unicorn.dat` files.

Note: Sometime sounds in the room can be poor. Turning on closed captioning by pressing the cc button will improve the accessibility of these videos.

This exercise and discussion should take about 15 minutes.

Solution

The instructor will lead a discussion about the videos and your feedback on them, making sure that the points of the Top Ten Tips below have been made.

Top Ten Tips for Participatory Live Coding in a Workshop

1. **Stand up and move around the room if possible.** This makes the experience more interactive and less monotonous. Use a microphone if one is available to make it easier for people with hearing difficulties to hear you.
2. **Go slowly.** For every command you type, every word of code you write, every menu item or website button you click, say out loud what you are doing while you do it. Then point to the command and its output on the screen and go through it a second time. This slows you down and allows learners to copy what you do, or to catch up. Do not copy-paste code.
3. **Mirror your learner's environment.** Try to create an environment that is as similar as possible to what your learners have to reduce cognitive load. Avoid using keyboard shortcuts.
4. **Use your screen wisely.** Use a big font, and maximize the window. A black font on a white background works better than a light font on a dark background. When the bottom of the projector screen is at the same height, or below, the heads of the learners, people in the back won't be able to see the lower parts. Draw up the bottom of your window(s) to compensate. Pay attention to the lighting (not too dark, no lights directly on/above the presenter's screen) and if

needed, re-position the tables so all learners can see the screen, and helpers can easily reach all learners.

5. **Use illustrations** to help learners understand and organize the material. You can also generate the illustrations on the board as you progress through the material. This allows you to build up diagrams, making them increasingly complex in parallel with the material you are teaching. It helps learners understand the material, makes for a more lively workshop and gathers the learners' attention to you as well.
6. **Turn off notifications** on your laptop and phone.
7. **Stick to the lesson material.** The core Carpentries lessons are developed collaboratively by many instructors and tried and tested at many workshops. This means they are very streamlined - which is great when you start teaching them for the first time. It may be tempting to deviate from the material because you would like to show a neat trick, or demonstrate some alternative way of doing something. Don't do this, since there is a fair chance you'll run into something unexpected that you then have to explain. If you really want to use something outside of the material, try it out thoroughly before the workshop: run through the lesson as you would during the actual teaching and test the effect of your modification. Some instructors use printouts of the lesson material during teaching. Others use a second device (tablet or laptop) when teaching, on which they can view their notes and the Etherpad session. This seems to be more reliable than displaying one virtual desktop while flipping back and forth to another.
8. **Leave no learner behind.** Use sticky notes, see below, to gauge learners' progress and understanding.
9. **Embrace mistakes.** No matter how well prepared you are, you will make mistakes. This is OK! Use these opportunities to do error framing and to help your learners learn the art of troubleshooting.
10. **Have fun!** It's OK to use humor and improvisation to liven up the workshop. This becomes easier when you are more familiar with the material, and more relaxed. Start small, even just saying 'that was fun' after something worked well is a good start.

Sticky Notes

Give each learner two sticky notes of different colours, e.g., yellow and blue. If someone has completed an exercise, they put the blue sticky note on their laptop; if they run into a problem and need help, the put up the yellow one. This is better than having people raise their hands because:

- it's more discreet (which means they're more likely to actually do it),

- they can keep working while their flag is raised, and
- the instructor can quickly see from the front of the room what state the class is in.

Sometimes a yellow sticky involves a technical problem that takes a bit more time to solve. To prevent this issue slowing down the whole class too much, you could use the occasion to take the small break you had planned to take a bit later, giving the helper(s) time to fix the problem.
Remind learners frequently about using their sticky notes, or they (and you) will forget.

Practice Teaching

Teach 3 minutes of your chosen lesson episode using live coding to one or two fellow trainees, then swap and watch while the other person(s) live codes for you. (For this exercise, your peers will not "code-along", but will instead observe and give feedback.)

Explain in advance to your fellow trainee(s) what you will be teaching and what the learners you teach it to are expected to be familiar with.

Don't record this exercise. Give each other feedback using the 2x2 rubric we discussed previously and enter the feedback you received in the Etherpad.

This exercise should take about 25 minutes.

Key Points

- Live coding forces the instructor to slow down.
- Coding-along gives learners continuous practice and feedback.
- Mistakes made during participatory live coding are valuable learning opportunities.

Preparing to Teach

Overview

Teaching: 20 min
Exercises: 30 min
Questions

- How should I prepare to teach?

Objectives

- Use a learner profile to think about someone likely to attend a workshop you will teach.
- Classify the level of a learning objective in terms of Bloom's taxonomy.
- Critically analyze a Carpentries lesson's objectives.
- Describe reverse instructional design and explain why this is useful when preparing to teach.
- Identify checkpoints in a lesson for formative assessment.
- Recognize instructor notes as a resource for preparation.

Yesterday we started a discussion of the importance of lesson study. We started out by focusing on the lessons we can learn as instructors from watching others teach, having others observe our teaching, and giving and receiving feedback based on these observations. In this section, we'll switch our focus to another important part of lesson study: detailed discussion of curricular components (including lesson objectives, contents, and exercises) with an eye toward actively preparing to teach a workshop.

Learner Profiles

To teach effectively, you have to know *who* you are teaching. Your audience can be identified in many ways. Frequently people who are hosting a workshop have a specific audience in mind, based on their own experience.
One "creative" way to think deeply about the audience for a workshop is to take a few moments to write *learner profiles*. Learner profiles have three parts: the person's general background, the problem they face, and how the course will help them. One example of a learner profile for a Software Carpentry workshop might be:

> João is an agricultural engineer doing his masters in soil physics. His programming experience is a first year programming course using C. He was never able to use this low-level programming in his activities, and never programmed after the first year.
>
> His work consists of evaluating physical properties of soil samples from different conditions. Some of the soil properties are measured by an automated device that sends logs in a text format to his machine. João has to open each file in Excel, crop the first and last quarters of data values, and calculate an average.
>
> Software Carpentry will show João how to write shell scripts to count the lines and crop the right range for each file, and how to use R to read these files and calculate the required statistics. It will also show him how to put his programs

and files under version control so that he can re-run analyses and figure out which results may have been affected by changes.

Learner Profiles

Read Software Carpentry's learner profiles. Note that these example profiles contain more information than you will ever know about a learner; this is a creative exercise in imagining (and empathizing with) the whole people behind the faces. Now, sketch out a profile of someone you might expect to attend your first workshop. Who are they, what problems do they face, and how might this training help them? Be as specific as possible.
Enter your learner profile into the Etherpad.

This exercise should take about 10 minutes.

Reverse Instructional Design (and Preparation!)

When sitting down to plan a course or workshop, it might be tempting to dive into reviewing the content, questioning your understanding, and anticipating questions that learners might have for you. While it is good to prepare your content, this approach can take you down extended rabbit-holes in which you anticipate and research questions that only an expert would think to ask, and never get around to thinking about how to get your learners from one point to the next, and how to know when they've gotten there.
This is a problem with curriculum design as well as preparation. When writing curriculum, it is easy to allow *content* objectives to distract from *learning* objectives. One way to prevent this is to take a "reverse" approach to instruction, as advanced in Wiggins and McTighe's *Understanding by Design*, that keeps the focus firmly on learning outcomes. The order of preparation in this case becomes

1. Determine your learning objectives
2. Decide what constitutes evidence that objectives have been met, and design assessments to target that evidence
3. Design instruction: Sort assessments in order of increasing complexity, and write content that connects everything together

In the context of preparing for a Carpentries-style workshop, the lesson design has already occurred, and many lessons include pre-written exercises to use for assessment. In this context, *reverse instructional design* principles might be applied as follows: 1) review the lesson's learning objectives carefully, thinking about how they will work for your audience, 2) scan the lesson to identify promising points to check in with your learners, using formative assessment to verify that objectives have been met, and then 3) review the connecting content in detail to be sure everything works and you have anticipated likely problems and questions.

Working With Learning Objectives

Once you have an idea of your intended audience, the next step is to think through the goals for your workshop. These goals are usually communicated through *learning objectives*.

The “learning objectives” section is an easy thing to pass over when you’re preparing to teach. It may seem obvious or unnecessary. However, good learning objectives are quite specific about the intended effect of a lesson on its learners. We aim to create learning objectives that are specific, accurate, and informative for both learners and instructors.

Bloom’s Taxonomy is a framework for thinking about learning that breaks progress down into discrete, hierarchical steps. While many ideas have come and gone in education, Bloom’s has remained a useful tool for educators, in particular because the hierarchy seems to be reasonably valid: outcomes at the top of the hierarchy cannot be achieved without mastery of outcomes at the bottom. In the long term, everybody wants to be at the top. However, in aiming to meet learners where they are, we also need to be mindful about helping them to “grow a level,” helping them to recognize when they have achieved that growth, and guiding them to look ahead to where we might not be able to take them.

Bloom's Taxonomy

Image credit: Vanderbilt University Center for Teaching

Evaluate Learning Objectives

Your instructor has posted links to a handful of current Carpentries lessons in the Etherpad. Select one learning objective from one of those lessons, then complete the following steps to evaluate it.

1. Identify the learning objective verb. How specifically does this verb describe the desired learner outcome?
2. Where does this verb fit on Bloom’s taxonomy? Do you think this is an appropriate level for your learners?
3. In your opinion, does the lesson do an effective job of meeting the stated objective?
4. What would the next level on Bloom’s taxonomy look like for your learners? How might you be able to help them think ahead to the next level without attempting to get them there during your workshop?

This exercise should take about 10 minutes.

Using Formative Assessments

When assessments are created in a reverse-design setting, their primary purpose is to inform the instructor about whether objectives have been met. But, as noted in previous lessons, these assessments are good for everyone! For learners, becoming more aware of their progress is motivating and helps to transfer learned content to long-term memory. Awareness of the learning process, also known as "metacognition," will also help them to identify appropriate next steps after a workshop has completed.

Where are your checkpoints?

Have a look at your lesson again. Choose a learning objective, and identify *where* in the lesson that objective should reasonably be achieved. How will you know that that objective has been met for all learners? Will this be clear to them?

Make a plan for *where* in your lesson you will use different types of formative assessment to help everyone in the room monitor their progress. Keep in mind that formative assessment can take many forms, including multiple choice questions, faded examples, spontaneous questions and calls for sticky notes. Write some notes or thoughts about this process in the Etherpad for discussion.

This exercise and discussion should take about 10 minutes.

You Can't Just Ask

Self-assessments of skill level are usually inaccurate because of the Dunning-Kruger effect: the less people know about a subject, the less accurate their estimate of their knowledge is. This is one reason why assessments should be specific, as opposed to asking if everyone understands.

Instructor Notes

Many of the Carpentries lessons have instructor's notes, with information from instructors who have already taught the material. This can be a valuable resource when preparing lessons, especially when teaching a lesson for the first time.
The instructor notes are linked on each lesson page under the "Extras" pull down menu. In addition, configuration problems and other technical hurdles common across multiple lessons are detailed here along with suggested solutions. This link is on the workshop page as well for easy access by learners and during workshops. We'll see more about workshop pages this afternoon.

Feedback On Your Challenges (Optional)

With these goals in mind, pair up with a partner to discuss the MCQ and faded example problems that you wrote yesterday. Give each other specific, actionable feedback that follows our 2x2 framework. Use that feedback to make at least one modification to your exercise(s). Discuss in the Etherpad the change you made and how it will help you get more useful information about your learners.

This exercise and discussion should take about 15 minutes.

Key Points

- To teach effectively, you have to know *who* you are teaching.
- Good learning objectives communicate the intended effect of a lesson on its learners.
- A good exercise provides useful guidance to instructors about next steps needed in teaching.

Morning Break

Overview

Break: 15 min