Welcome to The Carpentries Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try etherpad.wikimedia.org).

Users are expected to follow our code of conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

All content is publicly available under the Creative Commons Attribution License: https://creativecommons.org/licenses/by/4.0/

Day-2: Line 730

Sign in: Name (Pronouns), Institution, Email & Twitter (optional), From where do you attend this workshop,
Please sign in so we can record your attendance.

- Anne Fouilloux (she/her/hers), University of Oslo - Norway, annefou@geo.uio.no & @AnneFouilloux, Oslo (Norway)
- D. Sarah Stamps (she/her/hers), Virginia Tech - USA, dstamps@vt.edu, @dsarahstamps, Blacksburg, Virginia (USA)
- Konrad Förstner (he/him/his), ZB MED - Information Centre for Life Science / TH Köln, foerstner@zbmed.de, @konradfoerstner, Cologne, Germany
- Julika Mimkes (she/her/hers), Staats- und Universitätsbibliothek)Göttingen , mimkes@sub.uni-goettingen.de, @JulikaMimkes, Göttingen (Germany)
- Johan Andresen (he/him/his), Danish Business Agency (starting March 8), johan.andresen@gmail.com, Copenhagen (Denmark)
- Matthias Lüthi (he/him/his), METAS (Eidgenössisches Institut für Metrologie), matthias.luethi@metas.ch, Bern, Switzerland
- Callum Rollo (he/him/his), University of East Anglia - UK, c.rollo@outlook.com, @callum_rollo
- Harry Fisher (he/him/his) Pharmatelligence, Cardiff. harryfisher21@gmail.com
- Sunniva Indrehus (she/her/hers), University of Oslo, Norway, sunniva.indrehus@geo.uio.no
- Maggie Hellström (she/her/hers), Lund University, Lund, Sweden, margareta.hellstrom@nateko.lu.se (no Twitter)
- Rebecca Stevick (she/her/hers), Institut Pasteur - Paris, rebecca.stevick@pasteur.fr & @rjstevick
- Jonas Hügel (he/him/his), University Medical Center Göttingen, Germany, jonas.huegel@med.uni-goettingen.de
- Will Blevins (he/him/his), Centro Nacional de Análisis Genómico-Centre for Genomic Regulation (CNAG-CRG), will.blevins@cnag.crg.eu, @willblev, Barcelona, Spain
- Kimberly Meechan (she/her/hers), EMBL Heidelberg, Germany, kimberly.meechan@embl.de
- Mabrouk Boutagouga, Professor of Anthropology, University of Batna 1, Algeria, @boutagouga, www.fa3iloon.org, www.aranthropos.com, boutagouga.mabrouk@gmail.com
- Chris Neuroth (they/them), freelance/with humans tech collective, Germany

- Kerstin Haase (she/her), ECRC of Charite and MDC Berlin, Germany, kerstin.haase@charite.de, @HaaseKerstin
- Thomas Gladytz(he/him/his), MRI scientist at MDC Berlin, Germany, thomas.gladytz@mdc-berlin.de
- Jiří Vyskočil (he/him/his), CASUS – Center for Advanced Systems Understanding, Germany, j.vyskocil@hzdr.de
- Charlie George (She/Her), Univeristy of Oxford - UK. charlotte.george@imm.ox.ac.uk, @A_CharlieGeorge
- Agata Bochynska (she/her), University of Oslo, Norway, agata.bochynska@ub.uio.no, @AgataBochynska
- Ryan Daniels (he/him), University of the Western Cape, South Africa, jryan.daniels@gmail.com,
- Heidi Karlsen (she/her), University of Oslo, heidi.karlsen@ub.uio.no

Please fill out the pre-training survey at
https://www.surveymonkey.com/r/instructor_training_pre_survey?workshop_id=instructor-training

You can keep track of the time in your current timezone at https://timeanddate.com/worldclock.

Ice breaker

What is your favourite emoji?

- If you need inspiration: https://emojipedia.org/
- sparkles:
- usually: , last year: ♀
- my favourite doesn't seem to display here, so I selected 😊 instead !
- until they come up with a capybara emoji
- until Python makes it big
- always
- : Whale farting combo
-
- and
- it's supposed to be a hug, but I use it as *jazz hands*
- "eh?"

I. Welcome

Code of Conduct:
https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

Assessing Trainee Motivation and Prior Knowledge

Background (3 min)

Have you ever participated in a Software, Data or Library Carpentry Workshop?

- Yes, I have taken a workshop. +1+1+1+1+1+1
- Yes, I have been a workshop helper. +1+1+1+
- Yes, I organized a workshop. +1+1+
- No, but I am familiar with what is taught at a workshop. +1+1+1++11+1+1+1+1+1
- No, and I am not familiar with what is taught at a workshop.+1+1+1

Which of these most accurately describes your teaching experience?

- I have been a graduate or undergraduate teaching assistant for a university/college course. +1+1+1+1+1+1+1+1+1+1+1+1+1
- I have not had any teaching experience in the past.
- I have taught a seminar, workshop, or other short or informal course. +1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1
- I have been the instructor-of-record for my own university/college course. +1+1+1+1+1+1+1+1+1+1+1+1+1+1
- I have taught at the primary education level.+1+1+1+1
- I have taught informally through outreach programs, hackathons, laboratory demonstrations, and similar activities.+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1

Formative Assessments Come in Many Forms

Identify the Misconceptions (10 min)
Choose one of the wrong answers to the question below and write in the Etherpad what the misconception is associated with that wrong answer.
Q: what is 27 + 15 ?
a) 42
b) 32
c) 312
d) 33

c) 312 = 7 +5 = 12, and then the values of the 10s are wronly summed. 3 is not added with the additional 1.
32 = 27+5 (and I forget adding 10)
33 = 27+5 (and I add the "1" to the wrong digit)
42 if a student is too lazy to compute and chooses 42 as an answer to anything. ;-)
312 = 2+1 =3 and 7+5=12, combining 3 and 12 as 312 and not adding it up
b) 32 , because the learner forgets to distinguish between adding 1's and 10's... Hello world!
312: *7+5=12*, **2+1=3** => **2***7*+**1***5* = **3***12*, QED
33 = 27 + 15 = 27 + 10 + 5 = 27 + 1 + 5
32 = they splitted it up in 27 +5 +10 and skipped th last part
33 = 27 + 1 + 5 (learner is not recognizing 15 as a distinct number, rather thinking it's a 1 and a 5 placed side-by-side)
33 = 7+5 is 13, not 12. Then forgot to add the 10
They forget that important questions always have one answer: 42. ;) <- yes!+1+1+1+1
32 - forget to carry over the 10 - broken model?

32 - they thought they should add 1 and 5 to 27 but got that wrong as well
33=the student know it is necessery to carry on 1 but he is carrying back to the same column

Modeling Novice Mental Models (10-15 min)

Take 10 minutes to create a multiple choice question related to a topic you intend to teach. Type it into the Etherpad and explain the diagnostic power of each its distractors, i.e., what misconception is each distractor meant to identify?

Your funder wants yout to publish Open Access. What possibilites do you have?
a) I put my preprint to arXiv, only.
b) I pay high APCs for publishing in a hybrid OA journal.
c) I put my draft in a git repository, only.
d) I did what my supervisor always did: I ignore this and publish in a jigh JIF-journal, only.
c) I ask my librarian :-)

There are different ways to publish OA. Journals want you to believe, that paying for OA in a hybrid journal is the best way, since this is how they can make the most money. Either publish in a full Open Access journal or publish in the traditional way but put your preprint/postprint in a repository.

Shell basics. Which of the following commands will change directory using a **relative path** from /home/myname/Documents to /home/myfriend/Downloads?
a) cd ../../myfriend/Downloads # Correct answer
b) cd /../../myfriend/Downloads # No leading / in relative path
c) cd(../../myfriend/Downloads) # The shell uses spaces, not parentheses () to pass arguments. Aimed at Python users
d) cd /home/myfriend/Downloads # misconception: that's an absolute path. Reinforce concept of relative vs absolute path

What is the path to the folder $HOME/Documents (unix shell related question)?
A) /home/USERNAME/documents
B) /home/USERNAME/Documents
C) /usr/local/Documents
D) /home/USERNAME/Document

Q: Which statement best describes the relationship between Git and GitHub?
a) Git and GitHub is are two alternative version controlling systems (belives GitHub is itself a version controlling system)
b) Git is a command line tool, and GitHub is the graphical interface of Git (understands that Git has a CLI and GitHub has GUI, but doesn't fully understand the relationship)
c) GitHub is the company that makes the open-source version controlling software called Git (understands that Git is open-source, but mistakenly believes that it was created by GitHub)
d) Git is a version control system, GitHub is an online hosting platform which can interface with Git (correct)

Q: What are/is the correct command(s) to save your changes of the file "plants.txt" to the git repository?
a) git add plants.txt; git commit -m "added tree" <- correct
b) git add plants.txt; git commit -m "changes made" <- too general commit message
c) git commit -am "added tree" <- might include other changed files
d) git commit -m "added tree" <- file was not staged, so commit will not have any effect

How often is the code in the body of the given loop executed:
word = "test"
for char in word:

- print(char)

a) 4 times -> correct
b) 6 times -> the " were interpreted as part of the string
c) 1 time -> did not understand the concept of the loop
d) never -> char is not a part of test

Which command is used to load the package "ggplot2" in R?
a) library("ggplot2") <- correct
b) ggplot() <- command within the package used to plot, after you've loaded the package. identifies confusion with commands vs packages containing commands
c) package("ggplot2") <- command doesn't exist. identifies confusion between package/library
d) install.packages("ggplot2") <- command to install the package, only needs to be done once before you load the package during each session. identifies confusion between installing package/loading package

Whats the best sequence of commands to do use to upload your file 'file.txt' from your local repository to github

a) git add 'file.txt' && git commit -m 'created new file' && git push -> good but don't understand the value of checking current git status first
b) git status && git add 'file.txt' && git commit -m 'created new file' && git push
c) git status && git commit -m ' created new file' && git push -> don't understand the concept of git add
d) git add 'file.txt' && git commit -m 'created new file' -> don't understand the concept of pushing

You have a data frame named cats, that has 5 rows and 4 columns. You would like to remove the complete 4th row. Which function will achieve this?
a) cats[-4] -> wrong, learner is assuming vector, not data frame with rows and columns
b) cats[,-4] -> wrong, learner is switching row and column indices
c) cats[-4,] -> correct
d) cats - 4 -> wrong, treating data frame as a numeric variable

In Open Data and Open Science, the FAIR principles (Findable, Accessible, Interoperable, Reusable) are considered central. Which of the following statements is true?
A. FAIR is a recognized standard in data science (wrong)

B. To be FAIR, data must be openly accessible and free of charge (wrong)
C. FAIR is a set of guiding principles (correct)
D. FAIRness is not concerned with humans using data, only computer processes (wrong)

u
What is the correct filter statement to filter all rows containing "group_1" in column "group"?
A. data %>% filter(group = "group_1") (one = is not a logical operator)
B. data %>% filter(group == group_1) (no quotes around the character string)
C. data %>% filter(group == "group_1") (correct!)

How do you share a previously non-existing file with your co-workers using the same git repository
a) git stage new-file && git commit && git push (the command for staging is add, not stage)
b) git commit -a -m 'go' && git push (works, but might share more than intended)
c) git add -p new-file && git commit -m "what my commit achieves" && git push

How do you get R to display sentences before text entries in a column of a data frame?
a) paste(My cat is, cats[coat])
b) paste("My cat is", cats[coat])
c) paste("My cat is", cats$coat)
d) paste(My cat is, cats$coat)

What is returned from the following?
x = [1,2,3,4,5]
print(x[2:4])
a) [2,3,4] > don't recall that indexing starts from 0, and that top index is exclusive
b) [3,4,5] > don't recall that top index is exclusive
b) [3,4] > this one's right :)

How do you count the number of lines in a file?
A. wc -l file.txt → correct answer
B. count file.txt → learner does not remember the wc -l command
C. ls file* | wc -l → learner is mixing use of the * option to count the number of files that start with file rather than counting the number of lines in the file
D. cat file → learner doesn’t remember the correct command wc -l

How do you save output of the *dmesg* command to a file:

- a) dmesg | log.txt - trying to use pipe to write to a file
- b) cat dmesg > log.txt - using cat as if dmesg were a file
- c) dmesg > log.txt - this one's correct (right? 😅)
- d) echo dmesg > log.txt - using echo to print something; works, but gives wrong answer

How ca we copy file.doc from folder A to folder B (both folder are in the root)
1- copy /A/file.doc to /B/file.doc.
2- copy A/file.doc to B/file.doc Not understanding the path architecture
3- copy file.doc to /B/file.doc Not understanding (source:target) notion

Navigate from /home/YOURNAME/Project/Data to your home directory:

1. cd ~/ : understanding the shortcut notation "~" for home directory. CORRECT ANSWER.
2. cd /home/YOURNAME/Project : not understanding where is the home directory
3. cd /Home : not understanding that bash is case sensitive
4. cd ./home : not understanding the difference between ~ and .

You want to add a list of URN's to your Jupyter Notebook. What do you do?
A) You copy the path in the example Notebook
B) You open the file with URN's, copy the text and paste it into the cell.
C) You check where your have the file stored on your compute and specify the path

What does 'sensitive data' term refer to?
a) a special type of research data that are collected by fragile software (refers to the understanding of 'sensitive' as 'delicate')
b) special type of personal data that include information about people's race or sexual orientation (correct answer)
c) special type of personal data that is about people's intuitive attitudes (refers to the understanding of 'sensitive' as 'intuitive')
d) a special type of research data that is easy to loose (refers to the understanding of 'sensitive' as 'delicate')

What can you learn from this error message: ...

III. Expertise and Instruction

https://carpentries.github.io/instructor-training/03-expertise/index.html

What Makes an Expert?

What Is an Expert? (5 min)

Name someone that you think is an expert (doesn't matter what they're an expert in). As an expert, what makes them special or different from other people? What is something that you're an expert in? How does your experience [+differ] when you're acting as an expert ~~differ~~ from when you're not an expert?

- They have automatized some skills
- Feynman - Was able to break down complex problems so non-experts were able to understand them.
- Someone with the label "professional" (has done some kind of organized and certified training)
- Christian Drosten: he is researching, publishing and talking about Corona-viruses.
- 1. Morris Fiorina. 2. expertise is usually associated with having in-depth knowldge and understanding of a specified subject. usually the way to become an expert is to know what is known and, in science, to contribute to knowledge. 3. in political psychology, and more specifically individual differences and political participation. 4. feeling efficacious in moving freely between related questions
- My mom (expert in me)
- A definition I read many years ago: "An expert is someone who puts the word 'expert' on their

business card" - signalling that the term is maybe not so valuable in itself . I think there are very few true experts, but there are people around with a solid knowledge of their discipline, with many years experience of both the practices & underlying theory, but who still are constantly re-evaluating their thinking in order to connect to the world around them!

- Robert Haase - They've created open-source software that is widely used for image analysis. They've taught many courses on the topic.
- Felipe Fernandes, expert in open source science. Excelent core understanding of the ecosystem and know where to look for relevant info. Understand how to help people transition from consumer to contributor
- Experience - Someone who has already made a lot of mistakes and learned from that. Ideally he has also learned from mistakes that others made before him.
- Hadley Wickham (author and maintainer of R tidyverse ecosystem); he spends a lot of time writing software to help researchers simplify/beautify their code and visual representations of the data, as well as creating resources to teach others how to use R.
- Someone who understands enough to teach/communicate their knowledge clearly
- Jeff Bezos (making money) - they have a public profile and have demonstrated they know how to do it in some form, you trust that they have done something others would find it hard to do. I am an expert in glucocorticoids, I am far more confident making generalisations and outlining broad concepts in a field i am expert in, whereas in a field i'm not an expert in I get hung up on specific facts and get worried about over/understating things
- Donald Knuth (computer science) - authored multiple books on algorithms, developed TeX
- A professor (expert in writing papers and knowing the right people)
- Dr. Gladys West, expert mathematician in applied geodesy. Published several papers, degrees in mathematics, award-winning, respected in her field.
- Dr Guy Midgley: Expert in Climate change and environmental consequences. Guy has university training in conservation and environmental sciences so he has a solid background in the fundamentals of considering climate and environment. He has published extensively in peer reviewed journals on the topic, so his opinions and work has survivied the critique of other experts. He works closely with the South African players in policy, agriculture and conservation, so he is familiar with the ramifications of environmnetal change on a broader scale (more than just conservation and biodiversity).
- An expert to me is someone who does not only have the knowledge to correct people when they demonstrate misconceptions in this person’s field, but to see the potential in questions from less experienced people as well - and, perhaps most importantly, to be have the courage and ability to demonstrate in practice, live, his or her own doubts and thinking process, not only digested knowledge.

Fluid Representations (5 min)

Give at least one example of a fluid representation that you use in your own work. If you can, also give an example of a fluid representation that might occur in a Carpentry lesson.

- Writing a function in a Python script vs importing that function from elsewhere on system/from a package
- Deriving a result in math or physics, (usually a lot of different ways to the same anwser)
- Programming in a jupyter notebook vs some other IDE
- Switching between CO2 /"carbon dioxide" or N2O/"nitrous oxide" also when talking to people with no or little experience in greenhouse gases (or chemistry in general)
- Different unit bases (e.g energy can be expressed in J, kWh, eV etc)

- Add 1 cup of flour (120 grams) (150 ml) <- my granny taught me that a cup is 150 ml, i.e. a volume rather than a weight equivalent, so I would run a risk of misunderstanding your recipe ;-) <<- my mom taught me that flour is a tricky one to measure, since weight can vary with humidity and volume can vary depending on how you 'scoop' it! Yup, noticed that yesterday when baking a loaf!
- Switching between the synonyms 'novel' and 'unannotated' when referring to genes which you detect that are not in the reference annotations
- directory and folder
- Mathematical treatment of waves on a pond and planetary waves over ocean basins
- a politician as a set of scores on traits & as something that is not so neatly categorized as we tend to believe by normal language...
- Copying something on a computer (use the window manager or the terminal)
- Using 'migration' and 'gene flow' interchangably (genetics terminology which doesn't mean the same thing in other fields)
- remote access - using technologies like VPN oder HAN for accessing licenced online resources
- switching between abbreviations and full terms (e.g. WGS, WGD, CNAs, SNPs and sequencing, whole genome doubling/tetraploidy, copy number aberrations/events)
- Visualizing data as they are or using transformations like singular value decomposition
- Using Google Apps vs Using open sources apps
- Giving a brief history of Anthropology in US
- switching between the English and the German term for something, eg. DNA/DNS <- this happens a lot when you forget the word in your native language and/or learned a new subject in a different language!ADN in French too!
- Discussing rheological flow laws in terms of plastic flow, ductile flow, power-law creep, or dislocation creep.
- Copying a file into the home directory leveraging the tilda command (~) or by putting in the full path.
- Clicking on 'Run' button vs using a key shortcut to run a chunk of code
- Forgetting that my own IDE installation has more features that the one the students are using -> referring to functionality that they cannot see or don't have access to
- measuring distance in seconds (e.g. particle physics, implcitly multiplied with speed of light)
- doing the same thing in git on the command line or using a GUI tool such as SourceTree
- When an instructor can reply to someone’s question (that perhaps is not totally relevant) in a way that is respectful to the student but also subtly keep us back on track and makes us advance, in other terms: flexibility and multiple entries to what is being taught, that can be spontaniously activated.

What other words or phrases can have the effect of demotivating learners? What alternatives can we use to express this meaning in a positive and motivational way?
In the Etherpad, make a list of demotivating words/phrases and alternatives.
This discussion should take about 5 minutes.

- RTFM(read the f#$*ing manual) / LMGTFY(let me google that for you)-> Perhaps we can open the "help" page to see if there are any clues as to how we should proceed; I think section XYZ has the answer
- this is not so hard, but you do .....
- .question 2: can you try that for me? is it okay if I move on? can I help in any way? does it get tricky/difficult - where? I hope it is okay if I disagree with you a little bit here, is it? ==> ask questions rather than making assertions

- I'll just simply do **120 word per minute typing in 5 different windows** and now we... (full disclosure: this was me)+1
- simply+1+1
- Obviously+1+1
- All you have to do is ... +1+1000+1
- I assume you all understand the fast fourier transform, now the implementation
- "Clearly"
- If you think logically, you'll realise ...
- I had a teacher tell us in an introductory class (in organic chemistry) that if we couldn't follow along her lecture, then we should consider an alternate career, "like dentistry" (I've always wondered what she had against dentists.)
- In textbooks: "to the intelligent reader it follows naturally that..." 😳+1
- simple
- Easy+1
- Of course
- quickly+1 +1+1
- just ("we just need to load this package", "we just need to add this here", etc) +1
- As you all know+1
- This is an easy excercise +1
- Only do/go/check ....
- this is not extensive
- as you can see
- You are doing well, it is so easy+1
- The next time you will do better
- it is quite hard and I do not expect you to finish +1
- None of you will surpise me because I have seen all types of pupils before (as my teacher said our first day in high school)

Break

until XX:35

IV. Memory and Cognitive Load

https://carpentries.github.io/instructor-training/05-memory/index.html

Test Your Working Memory (5 min)
This website https://miku.github.io/activememory/ implements a short test of working memory. In this test, you will see about twenty words, each for a short amount of time. Try to memorize as many as you can.

What was your score? Write your answer below.

- 4 (I have ADHD, not sure if this helps or hurts 😄) -> 5
- 8->7
- 5 -> 6

- 7->8 words, that' s all what i remebered
- 3 (another ADHD person haha maybe it's us ;)) -> 5 -> 3
- 4 -> 4 (tried word categories, like verbs)
- 4 5
- 6 --> 10
- 4 --> 5 --> 5
- 6 --> 9
- 5 --> 9! WOW! Big difference!
- 6 --> 10!! o_O
- 6 --> 7 I think I already tried sort-of chunking the first time
- 3 --> 6
- 4 --> 5
- 5 ->6
- 6 > 6 > 7
- 5 --> 6
- max 6 > 4
- 4 ---> 19 (The strategy was to record the sequence with my phone... ;)) -50 points to Gryffindor (HP is universal, isn't it)

Concept Maps as Instructional Planning Tools

Example concept maps:
https://carpentries.github.io/instructor-training/fig/array-math.png
https://carpentries.github.io/instructor-training/fig/conditionals.png
https://carpentries.github.io/instructor-training/fig/create-destroy.png
https://carpentries.github.io/instructor-training/fig/dict-set.png
https://carpentries.github.io/instructor-training/fig/io.png
https://carpentries.github.io/instructor-training/fig/git_concept_map.png
https://carpentries.github.io/instructor-training/fig/lists-loops.png

Concept Mapping (10 min)

Create a hand-drawn concept map for a part of a Carpentries lesson you would teach in five minutes (ie. the amount of material you would teach before doing a formative assessment). You can use the same subject about which you created a multiple choice question, or a different subject. Trade with a partner, and critique each other's maps. Are there any concepts missing in your partner's map that you would include? Are there more than a handful of concepts in your map? If so, how would you re-divide those concepts to avoid overwhelming your learners' working memory?
Take 10 minutes to draw the concept maps and share with your neighbor. Write "*done*" in the Etherpad chat once you have finished.

https://docs.google.com/forms/d/e/1FAIpQLSfa4S01QQLPkH8D2bwg9MDjP5M7IY7qnesM-4D80yW-PWOMWw/viewform?usp=sf_link

Lunch / Break

--> XX:30 (13:30 GMT) Check in your timezone: https://www.timeanddate.com/worldclock/fixedtime.html?msg=Carpemtries&iso=20210225T1330

VI. Motivation and Demotivation

https://carpentries.github.io/instructor-training/08-motivation/index.html

Link to the figure: https://carpentries.github.io/instructor-training/fig/what-to-teach.png

Authentic Tasks: Think, Pair, Share
Think about some task you did this week that uses one or more of the skills we teach, (e.g. wrote a function, bulk downloaded data, built a plot in R, forked a repo) and explain how you would use it (or a simplified version of it) as an exercise or example in class. **Pair** up with your neighbor and decide where this exercise fits on a graph of "short/long time to master" and "low/high usefulness". In the class Etherpad, **share** the task and where it fits on the graph. As a group, we will discuss how these relate back to our "teach most immediately useful first" approach.

This exercise and discussion should take about 10 minutes.

- Installing packages in R (Cran, bioconductor, devtools) and using sessionInfo()
    - Could use it to install a set of packages we will use later in the class, install one package from each source to show differnt syntax
    - Load libraries into session
    - Use sessionInfo() at the end to show loaded packages.
    - Takes quite a short time to master, but is very useful as you need it to do anything outsode of baseR
- Teaching a new student how to use a complex finite element code that is installed on a cluster, which requires using the terminal. Break down the steps with some critical skill-building like navigating in the terminal (cd, ls commands), searching in files for keywords (grep, * commands), and counting files (wc -l, * commands). All of these would plot on high impact and low time to learn (teach this first).
- Setting up my dotfiles with version control (very useful for people changing environments often jumping from clusteres to personal cmputer etc.)
    - I would teach this with an example repositroy and encourage learners to put in their own versions for their personal set-up.
    - An intermediate learner in git could use this as a repetision task
- Wrote a shell script to 'tidy up' files after a big batch of analyses are completed, and back up important files to cold storage
    - introduce concept of hot storage (e.g. */scratch*), warm storage (e.g. */home*), and cold storage (e.g. */backup*)
    - how to write *if/else* statements

    - how to iterate through subdirectories in a *for* loop
    - how to get user input (to confirm permanently deleting files)
- Made a branch for experimental refactoring of a codebase.
    - I would teach this toward end fo a git course. Say you want to change one smal thing in your codebase. This has a knock on effect, you change several files.
    - After an hour you realise this is a lot more work and you don't have time right now. Do you go back through all file pressing ctl-Z?
    - With git, if you made a branch, you can save your progress, go back to your main branch and return to this later. No loss of data or wasted effort
- importing different files as data frames in R
    - show what tweaks allow students to important different formats (long/short, csv, or from statistical software packages like SAS)
- use the console to delete files and direcorty on a server and add an alias (rm -I) for rm .
    - does not take much time to master, but can have an high impact, because you need to confirm that you want to delete files and can not delete them by canche
- Make a 2d graph from 3d data: Show how to use functions to make it possible to display slices in different axes (adapting scales, colorbars etc). --> I would put this probably somewhere in the middle of the graph, do not the highest impact
- Delete files in a folder
- Downloaded files from the NCBI SRA database
    - Depends on the application... always very useful, mastery depends on the application. Start with a simple example downloading one file, then do a bulk download or something more difficult to master
- Set up a new git repo
- Locate a file on a file system
- Visualize usage data
- Cloned a repository, and ran code from it. I would show cloning a repository from github, and then opening a file from that in e.g. jupyter notebooks. There could then be an exercise where learners choose a github repository that's relevant to their work and clone it to their own computer.
- Import URN's to be analyzed in Jupyter Notebook
- Used my IDEs refactoring tools to rename a variable
    - give an example of the same code, once with all variables named a,b,c,d,... and once with meaningful names
    - walk through manually changing a variables name, making a mistake on the way, e.g. renaming the definition and 2 out of 3 usages, but forgetting the last
    - re-doing the same using the automated tool which is faster and safer
- estimate the noise level in a multivariate dataset using its singular values
- Imported data into R and created aligned figures to compare different experimental setups
- Import .csv data into R and create a simple graph with ggplot2 (data import: short time to master, very high usefulness; building graphs: longer time to master, not as high usefulness - depends on the tasks performed)
- Teaching participants how to use population genomics work flows. This would allow them to make use of publicly available high density SNP chip data. In this example, I'm considering up skilling forensic scientists (no deep experience with unix or population genomics)
    - The workflows require becoming familiar with unix and several other tools.
    - The time to master the tools is actually quick long as it requires getting comfortable with the analyses behind them before one can be creative with the output.
    - This is definitely a lesson on the 'avoid' side of the scale or a specialist workshop.

- Creating a pptx document using R markdown. How to change the output format in R markdown, the concept of r chunks and inline r code and using a template document to help with formatting / styling
- download several large simulation snapshot files from a remote cluster, prioritizing those with potentially more interesting data, do this for multiple simulation directories
    - set variables for remote directory
    - set variables with patterns of the priority files
    - create local directories
    - loop over remote directories, use rsync to get the priority files
    - loop over remote directories, use rsync to get the remaining files
    - --> easy to master, useful if you ever deal with automated transfer of many files
- teaching how to use dataverse to upload a new dataset after cleaning it, and writing metadata

- Extracted metadata about supported variable types from a repository catalogue, using a sparql endpoint, and then analyzed the results. Easy: using the sparql endpoint GUI, applying pre-prepared queries. Easy: making small adjustments to query. Easy: exporting the output as CSV. Easy: transfer CSV to GoogleDrive. easy: Import CSV into GoogleSheets. Easy: sort data for analysis of commonalities. Difficult: learning sparql syntax. Difficult: understanding how to design sparql query needed. Difficult: grasping details of underlying triple store.

How Learning Works by Susan Ambrose, et al., contains this list of evidence-based methods to motivate learners.

In groups of two or three, pick three of these points and briefly describe in the Etherpad how we can apply these strategies in our workshops.

- Strategies to Establish Value

    1. Connect the material to learners' interests.
        1. Useful if you know the background of your participants. Give data examples connected to their field of studies
        2. Use the participants' language and be aware of their subject culture.

    - identify the learner's background and motivation for attending, even ask them to provide examples beforehand of cases from their field (if relevant/possible) that one can use when creating assignments and giving examples.

        1. Ask participants some simple questions about the topic
        2. Poll the learners at the beginning (or during registration) to find out their interests and connect the skills that will be taught with what they are interested in.
        3. Construct a storyline of the lesson that fit the learners background
        4. ask a question motivating participants to think about how they, themselves, can relate their interests to the material.
    2. Provide authentic, real-world tasks.
        1. we can provide real-world data examples when teaching simple coding tasks or data wrangling tasks

3. Show relevance to learners' current academic lives.
    1. Not just academic lives, but working situation! Ask participants to define their own real-world "use case", and try to map learning topics onto these+1 important to remember all learners are not academics! +1(and even if they are today they may move to industry in a very near future!)
    2. Ask the learners to bring their own data/problems to the workshop and they can work on their data when learning to use a new tool.+1+1
        1. This could be done with a survey beforehand. if possible.
4. Demonstrate the relevance of higher-level skills to learners' future professional lives.
    1. Quick example/links to use of laguage/material in relevant industry
5. Identify and reward what you value.
    1. Affirmatively thank learners for asking questions/assisting classmates
6. Show your own passion and enthusiasm for the discipline.
    1. Use some examples that are from your work/subjects you find interesting.
    2. Tell, how and why did you to start this discipline?
    3. Smile a lot and get enough sleep the night before so you can be energetic and have unending patience
    4. Tell a story about your own experience (or you friends or colleagues)

- Strategies to Build Positive Expectations

1. Ensure alignment of objectives, assessments, and instructional strategies.
    1. Have a rough timetable / list of objectives given to learners before the course
2. Identify an appropriate level of challenge.
    1. Know your audience and the level at which is most suited to them. Maybe provide more advanced examples for people who grasp the concepts quicker?
    2. Perhaps have a pre-course survey to figure out peoples knowledge level
3. Create assignments that provide an appropriate level of challenge.
    1. start with simple tasks and adjust the level in later assignments
    2. Build off of existing lessons or leverage exisiting lessons that have been successful in the past
    3. give options within an assignment for simpler or more complex solutions (like in yoga - there is always a base pose and variations that can make it more challenging ;))
4. Provide early success opportunities.
    1. Quick and simple first exercise at the beginning of the episode+1
    2. Gradually increase the difficulity in the exercise (like Anne showed us with the loop example)
5. Articulate your expectations.
    1. select your words and sentences carefully - given particpants less to focus on
    2. Provide a list of skills that learners should have when finishing the course/lecture
    3. Make sure that the learners have realistic/correct expectations about the content of the workshop
6. Provide rubrics.
    1. Start a new lesson after an example / task
    2. The lessons & all episodes should be presented in a easy-to-view listing, to illustrate context
7. Provide targeted feedback
    1. Reive <-- Is some text missing here?
    2. Repeat the question first and mention the name of the person who asked it.

3. Talk to participants individually
4. Respond to formative assessments

8. Be fair.
    1. make clear that mistakes are possiblities to learn +1
    2. Be aware if you do not "like" someone and ask yourself why +1
    3. Work to meet learnders where they are, all effort articulating issues can't be left to them
9. Educate learners about the ways we explain success and failure.
10. Describe effective study strategies.
    1. Mention the tricks/tips that you have learnt in working on your subject (e.g. annotating code)

- Strategies for Self-Efficacy

1. Provide learners with options and the ability to make choices.
    1. Give a few options for exercises or allow them to bring their own data/problems
    2. Choose how they would like to work, in small groups/pairs or as a whole class
2. Give learners an opportunity to reflect.
    1. Self-reflection is very important - with regard to both "what did I learn today" and "how does the course topics fit with my needs"
    2. State open questions for reflection and discuss the answers in the big zoom room (like we do now)
    3. Anonymous surveys for learners to reflect/respond in a safe space
    4. Breaks
        1. Provide short and long breaks, be clear when and how long they are.
    5. If you ask questions, give a specific amount of time or wait until multiple people want to give an answer instead of waiting until the first person has an answer
    6. have participants explain their solutions to neighbours +1

This exercise and discussion should take about 5 minutes.

- *Think* back to a computational (or other) course you took in the past, and identify one thing the instructor did that motivated you. *Pair* up with your neighbor and discuss what motivated you. *Share* the motivational story in the Etherpad.
- This exercise should take about 5 minutes.
- The instructor showed a lot of passion for the subject, he could clearly state the consequences of gaining the concepts he taught. This was a big motivation to listen to the teacher and try to gain the same insights.
- My python teacher loves teaching python, he looked so happy doing it, that I loved to listen to him.
- Another python teacher showed in a 5 min. lesson that it is easy to find out, when Trump sleeps - by analyzing his tweets (this episode is a few years old)
- Our practical excercises were easy, but they were always related to current research, so it felt interesting and important
- Related leanring python to harry potter - i can't remember how he did it but i do remember it made it seem fun and less formal and intimidating
- Showing their own mistakes, and working through them live e.g. coding errors
- The instructor showed a lot of excitement for what they were teaching.
- Bring your own data exercises led by my R instructor helped me get excited about using the same

code in multiple places

- I still remember how my high school physics teacher (ca 1982?), brought his own Sinclair ZX80 computer + a few really simple sensors (that again he had bought for hois own money) into class, and let those of us who were interested write simple code to control measurements of e.g. gravity constant (letting a marble fall between two photo cells). That was so cool, and really proved to me that computers were useful - if you knew how to program them! I then brought this knowledge & enthusism to my university studies...
- My lecturer in computational physics (taught in old fortran77!!!!) starting to use modern examples (modern fortran which is for instance object oriented)
- One of my stats professors in uni showed us how to do some visualizations and statistical tests with R, then told us to find an online database, download the data, and come up with some findings. Seeing the relationship in an international dataset a country's GDP, % investment in education vs military, and overall quality of live, was pretty amazing
- In one of my programming courses we had to do a group work where we needed to develop a programm to solve/help with current issues at the institute
- I recall learning introductory statistics where the teacher provided a code that participants had to use to learn central limited theorem and related concepts. It enabled us to simulate (Monte Carlo) the results of differences in the number of observations and sample sizes. This way of teaching was very engaging and turned the concept into a very practical matter.
- I find learning much easier when I can see the real-world relevance and ideas seem structured. One of my university lecturers (botany) would always immediate reference an example of a scenario where the biochemistry/physiology we were learning about could be seen in our backyards or on field trips. He would contrast different plants from the same habitat/setting but different physiology.
- Creating practical excercises in a way that the very excercise is simple but the result is very cool (or surprising) like a fun graph or a nicely formatted output - made it a lot of fun to play around with the code and was almost 'addictive' (in a good way!)+1
- My co-supervisor (so it’s not a course really) used a very clever and funny metaphor in order for me to better understand the difference between two topic modeling algorithms. It inspired me in multiple ways
- https://media.ccc.de/v/36c3-10652-bahnmining_-_punktlichkeit_ist_eine_zier +100

How Not to Demotivate Your Learners

Brainstorming Demotivational Experiences (5 min)
Think back to a time when you were demotivated as a student (or when you demotivated a student). *Pair* up with your neighbor and discuss what could have been done differently in the situation to make it not demotivating. *Share* your story in the Etherpad.

- When the instructor said stuff like "yhea, this is really basic....." before answering a question
- Clearly didn't want to be there (many academics): read from 10 year old slides in monotone and finish in half time
- i had a lecture where we had outdated slides, where already outdated technologies were marked as futere technologies. It just felt like they were not intrested in teaching the course and would not invest time to update it.
- A lecturer was explaining photosynthesis in undergrad but saying things that were completely contradictory to what was in the textbook. I'm guessing he got confused with the different

pigments but instead said the textbook must have an error.
- a teacher picked only the few female physics students for difficult tasks
- a physics' professor started his course with very difficult calculations (he hoped to give us a recent example on a comet's movement), in his next lesson he explained sine and cosine.
- Instructor took large parts of the course to talk only about their own niche research. Made it much harder to see the relevance to your own work!
- Moving on from excercise after only the 10% fastest poeple have completed it sooo common!
- Tell you that "no one ever understands this, it's okay" instead of explaining the concept in a different way
- A professor started the class by saying "none of this is really relevant anymore (php), but it's in the curriculum so we have to teach it"
- A course about spectroscopy was transfered from the 4th year of the studies to the 1st year without any changes. We just read E*(hay fork) = h*(hay fork) ??? We did not understand anything.
- I repeat my earlier example: I had a teacher tell us in an introductory class (in organic chemistry) that if we couldn't follow along her lecture, then we should consider an alternate career, "like dentistry" (I've always wondered what she had against dentists.) Instead, she should (of course) have tried to figure out how to rephrase her own message, seeing that there were many of us who had problems with the same concepts, and then approached us with kindness rather than sarcasm... We students tried desperately to cheer ourselves up and motivate each other to study harder.
- A professor saying "Sometimes your best isn't good enough."<- big ouch...
- My philosophy professor always saying "I don't believe in philosphy but i need to feed my family, so iteach it", this was the most demotivating teacher
- So far almost all maths lectures (given by math profs) I attended at an university level were for the purpose of proofing maths and not to help the students understand these concepts.
- When they answer to your quesiton in a way that confuses you more then you were previously.
- When the instructor was going through the material really quickly
- Our chemistry lecture used a book as script that had been 10 years out of print and wasn't available anywhere
- I had a lecturer who used to have about 100 slides for an hour long lecture and would just read from every slide and not even look at the class

https://github.com/UKHomeOffice/posters/blob/master/accessibility/dos-donts/posters_en-UK/accessibility-posters-set.pdf Hmmm - doesn't seem to explicitly include color blindness (except for pointing out that color alone shouldn't be used to define "meaning")? >

Since we are so used to being praised for our performance, it can be challenging to change the way we praise our learners. Which of these are examples of performance-based, effort-based, or improvement-based praise?

1. I like the way you tried a couple of different strategies to solve that problem.
2. You are getting really good at that. Keep up the hard work!
3. You are really good at that.
4. That was a hard problem. You did not get the right answer, but look at how much you learned trying to solve it!
5. You are a natural!

Break

XX: 41

We will start by observing some examples of teaching and providing some feedback.
Watch this example teaching video as a group and then give feedback on it. Put your feedback in the Etherpad. Organize your feedback along two axes: positive vs. negative and content (what was said) vs. presentation (how it was said). This exercise should take about 10 minutes.
https://www.youtube.com/watch?v=-ApVt04rB4U

- Group 1
    - Good:
        - "now we are gonna start"
        - asked for questions
    - Bad:
        - to small font size
        - "very easy"
        - to complex language
        - not focussed on one task/topics, but multiple
        - fix this, do that -> be more specific
        - sidekicks to excel
        - way to fast
- Group2
    - Good
        - presentation
            - eye contact
            - Gesturing/animated behavior?
        - content
            - after the break he started the lecture with "what we did before the break"
            - asked if there were any questions at the end of the section
    - Bad
        - content
            - The presenter seemed ill-prepared, as if he was relying on his ability to handle python tasks as an indicator that he can teach python. Not effort made to make the work accessible to novices.
            - Dropping randomly complex terms like 'lexical binding'
            - didn't explain mistakes
            - useless variable names
        - presentation
            - Small letters on the screen
            - Distracted by his phone
            - no pointer
- Group 4 awesome
    - Good:
        - audible

        - Attempting live coding
        - asked for questions after 2 mins 30
    - Bad:
        - Faced the board to talk
        - Lots of ums, ahs, "just"
        - Rapid changes of teminology for same thing
        - Introducing adanced unnecessary topics like polymorphism without explaining
        - Sarcastic to students during intro
        - Ignored his errors instead of teaching them
        - "This is really basic..."
        - Fonts used on the screen are too small
        - Using phone
        - speaking very speed

- Group 3 awesome
    - Good:
        - seems to arrive on time
        - looks up at class at least once
        - ask once if there are any questions
        - appologized for misspeaking
    - Bad:
        - usage of highly technical terms without considering audience comprehension
        - slandering use of GUI-based software
        - lots of use of demotivating language e.g. "just", "simple", "right"

- Group 5
    - Good
        - he spoke loud enough
        - hardware was working
    - Bad
        - content:
            - objective was unclear: teaching coding / coding principals?
        - presention
            - font size too small
            - he talked too quickly
            - no clear start
            - unclear audiance (beginners or experts)
            - jumping between topic
            - unprepared
            - cell phone rang
        - phrasing:
            - "simple"
            - "really simple"
            - "trust me"
            - "easy"

Feedback on Yourself (25 min)

The goal of this exercise is to practice giving and receiving feedback, so do not be overly concerned about creating a perfectly polished presentation. We do not give you a lot of time to prep and the teaching time is short, so embrace the challenge (and improvisation) of the exercise and see what you can learn by watching other people teach, thinking about how to give them feedback, and how it feels to get feedback on your own teaching.

- Split into groups of three.
- Individually, spend 5 minutes preparing to teach a 90-second segment of the lesson episode you chose before the start of the training course. You will not be live coding; you can use a whiteboard or other visual aids if available (but this is not required!). We recommend using this 90 second teaching moment to introduce the topic of your lesson.
- Each participants gives a 90 second presentation directly followed by feedback of the others. Also with in this setup one person should be responsible for the timekeeping.
- After everyone has given their talk, return to the main group and put everyone's feedback in the Etherpad.

- - calm voice
- - linked to previous session - put it in context
- - nice friendly welcome

- -Clear introduction
- -Good explanation of the topic
- -

Checked in on participants to make sure we are on the same step.
Made comparisons between the new tools and other tools that people may be more familiar with.

+ good pace, good use of highlighting the currently used aspects
- if you use a continuous script: make sure to scroll slowly
- explicitely define technical terms

https://carpentries.github.io/instructor-training/12-homework/index.html

X. Welcome Back

https://carpentries.github.io/instructor-training/13-second-welcome/index.html

Reflect from day-1 (5mn)

What have you learned? What are your thoughts
No need to write down anything in the etherpad. This 5mn are for you to reflect on day-1

Questions (5 min)
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.

Will someone invite me to the Carpentries Slack Workspace? https://swc-slack-invite.herokuapp.com/
Thanks! +1+1+1 can someone write here what it is?It's the main peer to peer comms for carpentries instructors and similar. great cheers!
https://docs.carpentries.org/topic_folders/communications/tools/slack-and-email.html
How can one join the TopicBox channels? Also, several of those seem to be inactive since many years - time to prune a bit?
Can we observe a carpentries workshop given by a more experienced instructor? Especially online...+1+1
Can we volunteer to teach geographically remote sessions during covid work-from-home?
If we want to organise a carpentries workshop at our institution, can we invite other carpentries trained people to assist (how do we go about setting this up)? Is there an procedure in place already?
Can it be a good idea to occasionally organize a workshop where the learners are from the same field and contextualize and use examples/material from this field in addition to the Carpentries material?
Is it allowed to use small bits of the Carpentries teaching material in courses, that are not labled as Carpentry workshops?+1
may I self-organize a workshop at my workplace, regardless of (a) the workplace (or what workplace it is), (b) the number of participants, and (c) the affiliation of the participants?+1+1+1+1+1
Is carpentries giving certificates for yes, here is a link for how to get a learner certificates:
https://docs.carpentries.org/topic_folders/hosts_instructors/certificates.html?highlight=certificate

XI. Live Coding is a Skill
https://carpentries.github.io/instructor-training/14-live/index.html

Why Participatory Live Coding?

Compare and Contrast (10 min)
Watch the two live coding videos 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.

live coding: https://www.youtube.com/watch?v=bXxBeNkKmJE&feature=youtu.be
Content +

- Explained how the prompt changes from $ to >
- Useful thing to learn
- good using a different variable name to comfirm concept +1+1+1

Content -

- There was no explanation of the structure of loop before creating it. +1+1
- many commands in one go+1
- Doesn't explain the mistake that he made+1+1+1+1

Presentation +

- speaks calmly and in simple words+1+1+1+1+1

- repeats the same tasks to explain concept (using different name for variable)
- friendly tone+1

Presentation -

- Very busy shell prompt
- No eye contact+1+1
- font size quite small+1+1
- Terminal not full screen - other stuff in background
- Uses entire screen for command window -> front row student heads may be blocking view for people in back
- Not saying out loud the commands they are typing+1+1
- notifications not muted on desktop+1+1+1+1+1
- one of the students has a red postit the whole time - should stop and address questions+1
- only explains what's happening afterwards
- Demotivating language "you can see that" without explaining was is "seen" +1
- Does not point to the part of the code he is talking about
- custom terminal/shell settings

live coding: https://www.youtube.com/watch?v=SkPmwe_WjeY&feature=youtu.be
Content +

- introduces each command one by one+1+1+1+1
- Great use of explaining up arrow for history + single line formatting
- (re)introduces arrow up, and explains it - giving the students another valuable tool without disturbing the lesson flow
- it is clear why learners want to follow the teaching. such as learning how to get displayed things in files.
- Good explanations of the purpose of each command

Content -

- not repeating what they did before the break
- Unusual file names can be hard to relate to or introduce confusion about what the files actually are+1 basilisk is hard to spell
- not asking if anyone has any questions
- No explanation of the structure of the loop before typing the loop
- Some inconsistencies in vocab, especially unix vs shell vs bash

Presentation +

- Checks for questions before starting
- Big font+1+1+1+1
- Standing and making eye contact+1+1 Much better general interaction with the learners
- Reads out exactly what is typed+1+1+1+1+1+1+1
- slow pace+1+1
- explains what went wrong and how to interpet error when making a mistake+1+1+1+1+1+1
- Uses only top two thirds of screen for command 's -> front row student heads aren't blocking view
- Uses the same terminal as the learners+1
- points to the section of code he's talking about so its clear what he's referencing +1+1

Presentation -

- Occasionaly speaks to board

- looks directly at the camera
- shell prompt too plain

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 will not 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. Do not do this, since there is a fair chance you will 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 is 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

alternative tools for online workshops: https://carpentries.org/online-workshop-recommendations/#other-tools

Practice Teaching (20 min)
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. 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 the live coding sessions. Give each other feedback using the 2x2 rubric we discussed previously and enter the feedback you received in the below.

https://ndownloader.figshare.com/files/2292172

Group 1

Content +

- Very clear content
- good explanation
- step by step coding
- Good recoery from technical issue

Content -

- Specialist language not explained (commit, hash etc)

Presentation +

- Get all students on the same page at beginning
- Highlighting the lines you're referring to

Presentation -

- moving mouse around without need
- Would be good to split into more chunks
- More print statements would help people follow along

Room 5:
Talk 1: Sarah
Content +
The objective is clear
you explain and define what the different elements are

Content -
I missed the meaning of -F

Presentation +
the voice is calm
You create a safe environment

Presentation -

try to avoid "ähm" (but there are only a few)

Talk 2: Julika

Content +
Different colors used for different parts of commands and definitions are clear.
Nice, clear illustrations

Content -
Could provide objectives of lesson at the beginning
If using slides, I would like to see screenshots of what the learner would be seeing.

Presentation +
"Happy that you are here today" makes me feel welcome
Appreciate the humor about cat authorship - Yes, I add to that! :-)
Good that you asked if we had questions

Presentation -
Not clear by what should be typed into the terminal.
I think it was a bit too quick

Talk 3: Heidi

Content +
Provided objectives at beginning.
Recognized others may have different operating systems and the helpers could help the learners with their operating system.
Once Spyder was opened, there was a definition of Spyder

Content -
Minimize other programs on the screen to minize distractions.

Presentation +
Started from scratch to show how to open terminal using Launchpad
Good slow start

Presentation -
Opened with notes on the screen instead of lesson content.
Use the whole screen for your presentation
Explain the three windows of Spyder

XII. Preparing to Teach
https://carpentries.github.io/instructor-training/15-lesson-study/index.html
https://software-carpentry.org/audience/

https://carpentries.github.io/instructor-training/fig/Blooms.png <- Bloom's taxonomy
Useful to analyze episode objectives with respect to the taxonomy levels

Break

until xx: 25

XIII. More Practice Live Coding
https://carpentries.github.io/instructor-training/17-performance/index.html

https://carpentries.github.io/instructor-training/demos_rubric/

Code of conduct violations:
https://docs.google.com/forms/d/e/1FAIpQLSdi0wbplgdydl_6rkVtBIVWbb9YNOHQP_XaANDClmVNu0zs-w/viewform

code of conduct:
https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html#code-of-conduct-detailed-view

Lunch/Break

XX:43

Feedback: https://forms.gle/MiLY2Fx6dkJVoGK8A

https://amy.carpentries.org/forms/request_training/

repos: https://carpentries.github.io/instructor-training/checkout/index.html#eligible-repositories

Help wanted:
https://carpentries.org/help-wanted-issues/

Slack and Mailinglists
https://docs.carpentries.org/topic_folders/communications/tools/slack-and-email.html

Workshop Template
https://github.com/carpentries/workshop-template

What's in an Introduction? (10 min)
Get into small groups (3-4 people) and discuss the questions below for 5 minutes. Take notes on your answers.

- What do you hope to accomplish in a workshop introduction?
    - discuss why we are here and what we are going to do
    - Remind everyone of the level at which the workshop is being pitched.+1
    - Brin all participants to the same starting ground
    - Set the 'tone' of the workshop (welcoming, helpful etc)+1+1
    - Where are the fire exits and when is the coffee
    - introduce the instructors and helpers

        - Make everyone feel welcome+1+1
        - When and how long are pauses? When do we conclude the workshop?
        - Break the ice
        - discover the background of the participants
        - Test the system that's in-place to run the workshop (i.e. make sure you can see all the chat messages, people can be heard, find out if some people have technical issue, or any other important technical information)+1+1
        - Introduce the carpentries community/philosophy+1
    - What information do you need to include in an introduction to accomplish these goals?
        - Reminder of the prerequisites needed to follow this workshop
        - Remind everyone that this is their personal learning experience, not an exam.
        - the scope and goals of the session
        - provide some personal information
        - Timings of the day
        - Code of conduct +1

After 5 minutes, come together, and combine ideas as a large group.
Compare your ideas with the list of topics below.

- Did you miss anything?
- Did you come up with something that’s not listed below?

Glossario
https://glosario.carpentries.org/

Translate Carpentries lessons to other language. Best to post your message at
https://carpentries.topicbox.com/

(by the way, there are several persons willing to work on translating python lesson in arabic; I see a few messages on this when searchign right now).

Ice Breakers: https://carpentries.github.io/instructor-training/icebreakers/

Don't forget to complete the post-training survey:
https://www.surveymonkey.com/r/instructor_training_post_survey?workshop_id=instructor-training