
Welcome to The Carpentries Instructor Training!

Day 2

Trainers:

- Mike Henry (He/Him), mike.henry@choderalab.org, @mikehenry42
- Jeff Oliver (He/Him), jcoliver@arizona.edu, @jcoliverAZ
- Tim Dennis (He/Him), tdennis@library.ucla.edu, @jt14den

Please sign in so we can record your attendance.

- Elizabeth Oladapo(she/her), University of Sheffield, UK. e.o.oladapo@sheffield.ac.uk
- Brenda Kiage (she/her), Pwani University, kiagebrenda@gmail.com
- Hanna L Ford (she/her), University of Arkansas, Fayetteville, US; Center for Advanced Spatial Technologies, hlford@uark.edu
- Lisa Chinn (she/ her), lchinn@unmc.edu, University of Nebraska Medical Center, @existentialhum
- Erika Gavenus (she/her), gavenus@student.ubc.ca, University of British Columbia
- Yashar Monfared (he/him), ACENET, Canada, yashar.monfared@ace-net.ca
- Dennis Milechin, (he/him), Boston University, milechin@bu.edu
- Abhishek Chakraborty (he/him), Lawrence University, abhishek.chakraborty@lawrence.edu
- Givanna Putri (she/her), putri.g@wehi.edu.au, Walter Eliza Hall Institute of Medical Research
- Rongbo Jin (he/him), Universiti of Arizona, rongbojin@arizona.edu
- Paul Czechowski (he / him), Helmholtz Institute for Metabolic, Obesity and Vascular Research, paul.czechowski@gmail.com
- Wilson(he/him), retired math teacher, tsangphwilson@gmail.com
- Kudzayi Bamhare (he/him), freelance developer, kudzayibamhare@gmail.com
- Jean Chang (she/her), Broad Institute, jlchang@broadinstitute.org
- Jeffrey Weekley (he/him), University of California, Santa Cruz, jweekley@ucsc.edu
- Federico Aurora (he/him), University of Oslo, federico.aurora@ub.uio.no
- Margherita Francescatto, AstraZeneca, margherita.francescatto@astrazeneca.com

- Felipe Ruiz Bruzzone (he/him), Universidad de Chile. felipe.ruiz@ug.uchile.cl. @felipe_ruizb.
- Ashley Juavinett (she/they), UC San Diego, ajuavine@ucsd.edu, @analog_ashley
- Airined Montes Mercado, University of Puerto Rico, airined.montes@upr.edu
- Doschmund Kwiatkowski, Programme Director, AIMEE PROJECT LTD, doschmund@aimeeproject.com

----

Motivation and Demotivation

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

- Why is motivation important?
- How can we create a motivating environment for learners?

Objectives:

- Identify authentic tasks and explain why teaching using them is important.
- Develop strategies to avoid demotivating learners.
- Distinguish praise based on the type of mindset it promotes.

Motivation Matters

- Any technique can fall flat when learners are not motivated
- In a short workshop, motivation to continue learning independently is a critical outcome

How Can Content Influence Motivation?

- Believing that something will be too hard to learn often becomes a self-fulfilling prophecy.

-- -- *Exercise: Authentic Tasks: Think, Pair, Share* -- --
1) 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.
2) Pair up with your neighbor and decide where this exercise fits on a graph of “short/long time to master” and “low/high usefulness”. https://preview.carpentries.org/instructor-training/fig/what-to-teach.png
3) 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.

https://cuckoo.team/ttt0321
https://preview.carpentries.org/instructor-training/fig/what-to-teach.png

This exercise should take about 10 minutes.

Room 1:
Dennis: After updating an R Script, I commited the changes to my local git instance of the project and then pushed the changes to the git repo project. This is a skill that is short time to master and very usefullness.
Lisa : used Figshare (data repository) to download data files in csv, examined data manually- highly useful and short time to master
Abhishek: learn how to use 'learnr' package to create interactive exercises, couple of days to master, hopefully very useful as an instructor
Erika: summary statistics in R using dplyr (longer time to master due to other tasks needed --importing data, etc.; mid/high-level in usefullness especially for field)
Room 2:
Brenda: Subset dataset using R for data cleaning and python functions for subsampling
(highly useful and short-time to learn).
Room 3:
Wilson: I learned how to pivot columns of data in Tableau before creating a stacked bar chart for the data.
Jean: install and use a python library (relatively short time to master for libraries without complex dependencies and worth it to use others' code rather than write it yourself)
Federico: create a MySQL table. I would place in the middle of the square.
Room 4:
 Paul C: Using an AI model to create an outline of a ceratin topic, it takes almost no effort to teach, as almost no skill is involved, and I do not know if it will be useful for anything as applications of AI language apper to be preety new.
Airined: 1) Creating Co-expression Networks using Rna-seq Gene expression data. This task could take a couple of months to master, given that it requires an understanding of Linux, interpretation of upstream processes before building a network, and finally, the construction of the network could be easier after acquiring this knowledge.
Room 5:

- * Examples:
    - 1. reformatting data (long time to master but worth it)
    - 2. subsetting data (relatively short time to master and worth it)
    - 3. drawing plots (e.g., simple scatter or hist; long time to master but worth it -- maybe middle of the graph). however, different plotting programs may be unnecessarily hard to learn.
    - 4. Using R parallel to speed up bootstrap(short time to master and worth it (massive speed gain)
    - 5. installing new programs (e.g., R, or specific packages)

How Can You Affect Motivation?

-- -- *Exercise: Brainstorming Motivational Impacts* -- --

Think back to courses you have taken in the past and consider things that an instructor has said or done that you found either motivating or demotivating. Try to think of one example in each case, and share your example under the appropriate heading in the Etherpad.
This exercise should take about 5 minutes.
https://cuckoo.team/ttt0321
Motivating experiences

- Jeffrey: When learning a new programming language, it's highly motivating when an instructor gives rich examples and walks the class through them without assuming we know anything. Demotivating is when the instructor makes a lot of assumptions about what we may know. For example, in a course I took last year, the first several hours of instruction were devoted to reading musical notation (something I have no skills in). I found this to be highly demotivating, as I had zero context and no prior knowledge. He didn't give any examples and I was left wondering if the course was for me. (all the music majors seemed fine with this) In the second lecture, he put up some Python code examples, and presumed that the class knew nothing, as we walked through the code, line-by-line.
- Wilson: I was extremely motivated when I took Harvard's CS50 for the second time last year. The main difference between the first time and the second time was that the provider of the course had switched to a different platform for submitting and grading assignments. It made a huge difference for my experience as a learner.
- Kudzayi: I felt very motivated when my coding instructor complemented my coding style and creativity. This is because I had been doing a lot of extra work outside of class so all my hard work felt seen and acknowledged
- Dennis: When instructor provides practical applications of tools or workflows motivates me to try them, or incorporate them into my work.
- Rongbo: when I know something that can help publish a paper, that was very motivated for me
- Abhishek: when theoretical exercises are connected to real-life applications
- Doschmund : Motivated in a class learning Mathematica for the first time, where my instructor was really engaging and created an environment to learn foundations
- Yashar: Instructor mentioned his difficulties during learning a concept in statistics and how he overcame the challenges over time (motivating)
- Elizabeth: Instructor said "That's a very thoughtful question". I find it motivating
- Hanna: An instructor added in multiple reflection moments where we could "code along" with him - these were very re-affirming
- Lisa: encouraging or repeating what I had just said and say "As Lisa just stated...."
- Erika: course moved from theoretical to applied, and instructor gave us a heads up that the initial weeks were going to be dense, was able to motivate us to make it through the theoretical as a chance to explore new ways of thinking and leaving "breadcrumbs" to the applied section
- Brenda: Motivated when I started seeing lab results from a long period of blank gels
- Margherita: I felt extremely motivated attending an online course in which assignment complexity was gauged perfectly to be within reach with the right amount of effort
- Jean: being given actionable suggestions that can be used to improve the results of work I'm doing
- Felipe: A really motivating instructor let me to use the learnings for my own interests; eg. I can apply data from my work to some of the evaluations.
- Airined: Once, an instructor motivated me by using analogies and adding a story-like approach when teaching, which I think was very useful in terms of motivating students.
- Federico: instructor explaining through questions he would ask and then, after hearing our answer, answering them himself
- Paul: I was motivated when I found the course materials of today's course well organised and understandable.

- Ashley: instructor seems approachable; they have a mental model of learning that jives with mine
- Givanna: "that's a good question".

Demotivating experiences

- Givanna: "you are way behind the class. I suggest you drop out of the course and enrol in the non advanced course". Rather than giving suggestions on how to improve myself.
- Ashley: If you don't know X you can't do Y (and I didn't know X); very little time to ask questions or get feedback
- Wilson: I lost my motivation early on in my first attempt to do Harvard's CS50 online for the first time because I couldn't figure out how to submit assignments for auto-grading. They had switched the submission and grading platform before I took it for the second time. My experience was much better the second time around.
- Rongbo: something that is very difficult to leaarn but not very useful or not commonly used
- Hanna: Signing up for a "hands-on" workshop for high performance computing (HPC), but on day 1 there is 4 hours of lecture on the history of HPC. :
- Federico: instructor going to fast, not step by step, having to much material
- Paul: Too much remote course work e.g. using Zoom is demotivating as it is not faciliating establishment of collegual friendships.
- Kudzayi: I have felt demotivated when I feel like my instructor is going through topics too quickly and is not putting enough care and thought when answering questions
- Doschmund : Demotivated where the lecturer in my modelling class didn't look at the students, didn't ask questions, didn't really engage
- Erika: I find it hard to stay motivated when I do not have the chance to apply or reflect on how I might be able to apply new concepts to my own work, I have had biostats courses that just stay in the abstract
- Brenda: When I fail to accomplish daily tasks for days on end
- Dennis: Attended a tutorial were setting up the environment consumed half of the workshop time because there were various issues and problems encountered.
- Yashar: Instructor told us that if you don't get it in 1 minute, it means you have to go back to a previous lesson.
- Abhishek: when somebody codes too fast or runs through a topic just to get it done
- Jean: demotivated by lecturers who read their slides and don't seem to make an effort to connect the content to other material we've been learning
- Felipe. When instructor says that you need to know something that you obviously don't know to understand the content (somethin the teacher thinks is very basic).
- Margherita: I felt very demotivated when I happened to be in classes ways to advanced for my level
- Elizabeth: "Google it"
- Lisa: No effort to meet me at my/ the students level. Too advanced right away.
- Demotivating is when the instructor makes a lot of assumptions about what we may know. For example, in a course I took last year, the first several hours of instruction were devoted to reading musical notation (something I have no skills in).
- Airined: A very demotivating phrase used by an instructor was "Even though you will not understand this, try to read it" This was very unhelpful because right before giving the assignment to students, assumed that they wouldn't understand. Ouch!

Invite Participation

- Establishing norms for interaction
- Encouraging learners to learn from each other
- Acknowledging when learners are confused

Encourage a Growth Mindset

- Positive error framing

-- -- *Exercise: Helping Learners Learn From Mistakes* -- --
A learner at your workshop asks for your help with an exercise and shows you their attempt at solving it. You see they've made an error that shows they misunderstand something fundamental about the lesson (for example, in the shell lesson, they forgot to put a space between ls and the name of the directory they are looking at). What would you say to the learner?
In the Etherpad, describe the error your learner has made and how you would respond.
This exercise should take about 5 minutes. https://cuckoo.team/ttt0321

- Lisa: Ope! Looks like you forgot a space between ls and the directory we're looking at. Can you explain why you might need the space?
- Doschmund : I would ask the learner to walk me through the steps taken and work with them to validate the response as a brainstorming session and help them to figure out where the issue is. Instead of calling out the issue straightaway, I would ensure the approach helps them to learn and figure out the root cause themselves.
- Elizabeth: use of uppercase when lower case is required: commands are case senstive.
- Dennis: File not found error. I would point out the important portion of the error message that indicates what the issue is. Walk them through strageties I use to debug this issue, such as have them show me the code that is loading the file. Have them then compare the path specified in the code to the actual location of the file on the file system. Have them check the filename and make sure its the same. And so on...until the solution is found.
- Kudzayi: A learner forgets to add quotation marks when creating a string variable and recieves an error when they try to print it to the console (python). I would look at the error together with the learner (NameError: name {variable} is not defined) and ask them what they think this error means before explaining it to them. We would then go to the line which Python has picked up for the error and inspect it together, I would ask them if they notice anything. I would then go over the fundamentals of strings.
- Paul: A learner had once overseen to distinguish `foo = c(4,.04)` from `foo = 4,.04`. I pointed to the docucmentation which asked for a "vector of length 2", and the syntactical way of defineing a "vector of length 2".
- Jean: (ls example) I see you got the error message "command not found" can you walk me through what you were trying to do? What is the command you are trying to run? The bash shell is very literal and it uses spaces to separate pieces of information, the same way humans use pauses to separate words in speech. the command you want is `ls` and then it is followed by parameters or instructions on the specific way you want to run the command. remember to separate the two with a space so the bash shell can identify the command you want to use (ls) and then it can find/parse the specific way you want the result (-l -a). You can also separate those with spaces but ls is smart enough to know that the dash means "here are some parameters" so you can put both parameters behind a single dash (-la). Not all commands allow you to put multiple parameters together. You

may need to consult the "manual" (ie. man ls) to know what the command expects.
- Brenda: Learner uses a function without loading the package. I would explain to the learner the aspect of functions and loading of packages that contain that function. I would then ask the learner to define which package the function he/she is trying use is from then instruct them to load and call the function.
- Ashley: "Why don't these variables exist -- I ran the function!" (Learner defines the function, but doesn't call the function) -- I would stress that using functions requires two steps, first, defining it, and then running it. After we define the function, we can show that the function exists in the namespace, but the variables do not. We need to use the name of the function in a call in order to return the variables.
- Rongbo: learners forgot to remove NAs from their dataset before they make plots. I would tell them this is very common and don't feel bad, I aslo made the same mistakes when I was learning
- Felipe. Learner can not use a function in R because the package was not loaded. I would try to use the error to frame it in a positive way, remarking that is very common at the beginning of the learning, and that framing it correctly it can give us a very useful learning. "Hey, this is a common error in our daily coding activities. We have to think about what the error is telling us. In this case, the program cannot find a function; this could be a spelling or package management mistake. We have to check both. In any case, following this logic will help us in the future.
- Hanna: Using the ls example above: "Oh hey, there, that's pretty close, but if you add the space between ls and the name of the directory, that should work for you. Would you like to try that? I make mistakes like that all the time, it's no big deal."
- Airined: If a learner has made an error of using an incorrect flag in the Unix shell, for example using -l instead of -h, I would respond that it's ok sometimes we can get confused, and to review the manual or use the "help" command to look up for the flags and what they correspond to.
- Jeffrey: First I would conduct a quick assessment by asking the learner what task they intended to perfrom. Confirming they wanted to list the contents of a directory, I would review the format of a basic Unix command, and remind the learner that a command is broken down into three parts: the command, options and arguments. Then, I would reiterate the difference between these three parts. Finally, I would ask them to reformulate their command.
- Wilson: The student forgot to indent a block of Python code in a for-loop. I would wait for the student to run the program and then help him/her decipher the error message. I'd also take the opportunity to emphasize that debugging code is a common task of a programmer. It is part of the learning process.
- Abhishek: I will ask the learner to recheck their code/error message and hear what they think might have caused the issue. Then I can talk about how a computer needs specific instructions such as ls and then a space, followed by the directory name
- Erika: learner tries to use package in R before installing: "Let's look at what the error is telling us... what might that mean? and now let's look back at your code and what library() does as a function." That might lead them to seeing that library() requires the package already be installed and we can go from there.

other qu

- Federico: a learner has a syntax error in a MyQL-query: I would say: "let's look at this togetehr: tell me what the query is telling the computer to do and then I would guide them step by step to the right solution
- Margherita: In an R workshop the first exercise requires you to calculate the mean of the vector X, previosuly defined. Learner comes saying "It doesn't work". I would ask to try doing it together so we can go through the error message. The errore reads "Error in mean(x) : object 'x' not found". I'd explain that it means the function mean() is not finding x. Then I would suggest to go backwards

in the excercise to see where we defined X. I would probably ask if you can see a difference between the defined etc

- Givanna: When encountering an error, the first thing to do is to not panic. Secondly, look at the error message. In this case, we get a "command not found" message, which means shell is not able to understand the command/code you just typed in. Let's then go back to the code. Is there something unusual in it? Were u trying to run the 'ls' command? - I'll basically keep giving them hints as to what go wrong.
- Yashar: in example above, I would say: "that's great, you almost get it right but let's look at the error and see what we can do about it, oh you just need to add space before the name of the directory to get rid of the error" Then I'll wait until the particiapnt run the new code and I would say: Excellent! Let me know if you have any other question.

- Presenting the Instructor as a Learner
    - "The typos are the pedagogy"
- Praising effort or improvement, not performance or ability

-- -- *Exercise: Choosing our Praises* -- --
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 examples of praise do you think are based on performance, effort, or improvement?

- 1. That’s exactly how you do it – you haven’t gotten it right yet, but you’ve tried two different strategies to solve that problem. Keep it up!
- 2. You’re getting to be really good at that. See how it pays to keep at it?
- 3. Wow, you did that perfectly without any help. Have you thought about taking more computing classes?
- 4. That was a hard problem. You didn’t get the right answer, but look at what you learned trying to solve it!
- 5. Look at that - you’re a natural!

This exercise should take about 5 minutes.

Givanna: Effort: 1, Improvement: 2,4 Performance: 3,5
Yashar: 1 is effort, 2 improvement, 3 performance, 4 improvement, 5 performance
Ashley: 1-effort, 2-improvement, 3-performance, 4-effort, 5-performance
Brenda: 1,4, - effort; 2 - improvement, 3,5 - perfomance
Doschmund - 1 - effort, 2 - improvement, 3 -performance, 4 - effort/improvement, 5 - performance
Dennis: performance - 3, 5 ; improvement - 2 ; effort- 1, 4
Paul: (1): improvment (2): improvment (3): performance (4): effort (5): performance
Jeffrey: Performance: 3,5 Effort: 1,4 Improvement: 2
Abhishek: performance - 3, 5; effort - 1, 4; improvement - 1, 2
Airined: 1, 3, 4- effort; 2- improvement; 5- performance
Kudzayi: 1 - effort, 2 -improvement, 3 - performance, 4 - effort, 5 - performance
Elizabeth: 1) effort 2) improvement 3)performance 4)effort 5)performance
Hanna: 1) effort; 2) performance; 3) performance; 4) improvement; 5) performance
Felipe. 1 = effort; 2 = improvement; 3 = performance; 4 = effort; 5 = perfomance.

Rongbo: 1 -performance, 2-efforts, 3-performance, 4-efforts, 5-performance
Jean: 1 improvement 2. performance&improvement 3. performance 4. performance&improvement 5. ability/performance
Margherita: 1. improvement effort 2. effort 3. performance 4. effort? 5. performance
Erika: 1--effort, 2--improvement, 3--performance, 4--improvement, 5--performance
Federico: 1 effort 2 3 performance 4 improvement 5 performance
Wilson: 1,2,4 - effort, 3,5 - performance
Lisa: 1 effort, 2 performance/ improvement, 3 performance, 4 improvement, 5 performance

First, Do No Harm!

Things not to do in a workshop:

- Talk contemptuously or with scorn about any tool or practice, or the people who use them.
- Dive into complex or detailed technical discussion with the one or two people in the audience who clearly don't actually need to be there.
- Pretend to know more than you do.
- Use the J word ("just") or other demotivating words we talked about in a previous lesson.
- Take over the learner's keyboard.
- Express surprise at unawareness.

Not Just Learners

- Why does your motivation matter?

Keypoints:

- A positive learning environment helps people concentrate on learning.
- People learn best when they see the utility in what they're learning and believe it can be accomplished with reasonable effort.
- Encouraging participation and embracing errors helps learners to stay motivated.

Back at :49 past the hour!

Equity, Inclusion, and Accessibility

https://carpentries.github.io/instructor-training/09-eia/index.html
Questions:

- Why are equity, inclusion, and accessibility important?
- What can I do enhance equity, inclusion, and accessibility in my workshop?

Objectives:

- Identify instructional strategies that are consistent with universal design.
- Recognize systemic factors that can distract and demotivate learners.
- Understand the role of The Carpentries Code of Conduct in maintaining an explicitly inclusive environment.

A Positive Environment for All

https://www.diversity.pitt.edu/education/diversity-equity-and-inclusion-glossary
Definitions

- Equity: The proportional distribution of desirable outcomes across groups. Sometimes confused with equality, equity refers to outcomes while equality connotes equal treatment.
- Inclusion: Actively engaging traditionally excluded individuals and/or groups in processes, activities and decisions in a way that shares power. Inclusion promotes broad engagement, shared participation, and advances authentic sense of belonging through safe, positive, and nurturing environments.
- Accessibility: Refers to the intentional design or redesign of technology, policies, products, and services (to name a few) that increase one's ability to use, access, and obtain the respective item. Each person is afforded the opportunity to acquire the same information, engage in the same interactions, and enjoy the same services in an equally effective and equally integrated manner, with substantially equivalent ease of use.

The Carpentries Core Values

-- -- *Exercise: Discuss The Carpentries Core Values* -- --

- Take a moment to read through the Core Values on this page: https://carpentries.org/values/
- Choose one core value that resonates with you. What is a decision you might make in a workshop that could look different if you were actively considering the core value you chose?

This exercise should take about 5 minutes.

- Elizabeth (she/her): Act Openly: knowledge, skills and questions should be shared openly so others can learn. I would not encourage direct/private messaging (may be specifi to online teaching).
- Brenda: Value all contributions - Designing various approaches to offer feedback such as anonymous responses to allow everyone to contribute freely, surveys and actively putting into action any recommendations
- Hanna (she/her): Inclusive of All - Reaching out to broader audiences, using inclusive language in workshop materials, making sure that no one feels excluded from the opportunity. Being welcoming and respectful - and meeting people where they are whether that is education, operating system, identity, expression, preferred programming language, domain, or other
- Lisa (she/ her): Access for All- instead of using jargon-y language, one way to make things more accessible is to use plain language. Or, if jargon, like "metadata" is necessary, explian what it is to the best of my abilities.
- Erika (she/her) Strength through diversity: (just one example) including in the workshop advertisements the langauge that the workshop will be conducted in and offering resources at the start of the workshop for people who might be more comfortable working in another language.

- Yashar (he/him)Access for All - Designing the course material in a way that can be accessible to everyone regardless of their background and previous experiences with the tool/software. Try to start from 0 and go to 5 instead of start from 10 and try to finish at 100.
- Dennis (he/him) Always learning - allow participants to share their experiences or a little about their work with the room. As an instructor acknowledge when you learn something new from them sharing. A demonstration that even instructors are continuously learning new things and don't know everything.
- Abhishek (he/him) Always learning - use encouraging language and words that value persistence, effort, and learning more than the final result
- Givanna (she/her) Community collabotration. I would have consulted the community more on how to improve the teaching materials and their delivery, rather than designing them just based on how I normally do things.
- Rongbo (he/him): Community Collaboration. I'll spend more time to allow learners to talk to each other, know about each other, and have a sense of community. Then in the future, they could help each other.
- Paul (he / him): *"Always Learning"* Encourage the belief that "no effort is wasted", and that everything one learns will be good for something, even if one may not know why yet,
- Wilson(he/him): We are always learning. I'd ensure that workshop participants will be given the opportunity to provide feedback about the workshop and that the participants will receive a response to their feedback.
- Kudzayi (he/him): "Always learning" - I really believe in and live by the principle of having a growth mindset. I believe making learning a focal point is very essential in a workshop and not just putting out information. Learning is not always pretty so I would strive to create an environment where people are comfortable to ask questions, get things wrong and make mistakes. There's no learning without trying so learners need to be reassured that they are in a nurturing environment.
- Jean (she/her): Empower One Another - find a way for learners in the session to connect with each other, if they choose, and provide some starting structure so they can continue learning and/or practicing what they've learned with someone who has participated in the same training session.
- Jeffrey (He/Him): Access for All. As was previously mentioned, planning a workshop in a BYOD environment poses some challenges with system configurations and setting environments correctly, but there are more fundamental issues with a BYOD approach to a workshop, especially when catering to underserved communities. That is, not everyone has a device they can bring. So, rather than requiring people to bring their own devices as a pre-requisite, I would plan to have a lending library of devices that learners can check out for the duration of the workshop (and perhaps longer). Additionally, I would plan on conducting all the exercises in a virtual environment, like Jupyter or Google Colab, so that there was less time spent on making sure the learners had the right setup and more time learning.
- Federico (he/him) We are always learning: make more space for feedback and collaborative learning during the workshop
- Felipe (he/him). *Community Collaboration*. I would like to find more space in my lessons to ask for feedback from learners (during this workshop I have learnt ways to do it), as well as receiving and implementing learners suggestions to improve my workshops and courses.
- Ashley (she/they): Act openly. Invite learners to disclose when they've made mistakes and to monitor their own progress in a transparent way (e.g., here are the ways in which I've excelled, here are the ways in which I've faltered). I'd like to create a space where all learners, regardless of who they are, can trust the learning community and express doubts when they feel they cannot.
- Airined: "Always Learning": As an instructor, I would emphasize that all of us are constantly

learning and as science and technology are evolving, we are receptive to learning something new everyday. I believe this would motivate students and encourage them to keep learning, even though they might think it is hard to learn a task.

- Doschmund - People first - I would open up the participation to communities outside of research/academia to other community settings to benefit from teaching and learning data carpentry due to the speed of the technology change we are in, so wider communities is taken with direction of travel
- Margherita: act openly. I think it is easy sometimes to forget about completely transparent behaviour. Being in particular open in terms of difficulties encountered in the content of the workshop

Accessibility

Image: Cartoon showing strollers, suitcases, bicycles, carts, and wheelchairs using curb cuts
https://carpentries.github.io/instructor-training/fig/sketchplanations-the-curb-cut-effect.png

Universal Design in Learning (UDL)
The key to UDL is creating redundancies such that learners have multiple options in how they:
1) receive
2) engage, and
3) share information.

Systemic Exclusion

Stereotypes

- may be explicit (conscious and deliberate) or implicit (unconscious and automatic)
- guide what we notice about people
- guide how we interpret people’s behaviors
- can facilitate quick judgements in appropriate situations (e.g. stopping a child from driving a car)
- can lead to systematically negative attitudes and behaviors towards members of certain groups

When Instructors have stereotypes about learners
When learners experience stereotypes about themselves

What can we do about our own stereotypes?

Better Together: Learning with Friends

Equity versus Equality

Inclusive Practices in a Carpentries Workshop

Setting Expectations with the Code of Conduct

Listening with Assessment and Feedback

Examining your Actions

Looking for More? Want to Contribute?

The Carpentries is actively working on improving our content and practices with respect to equity, inclusion, and accessibility. If you are interested in being involved in the development of this content, please let us know! Contributions to this page may be made on GitHub (click the "improve this page" link at the top), though our #accessibility channel on The Carpentries Slack, or by emailing team@carpentries.org.

Keypoints:

- Inclusivity is a key attribute of a positive learning environment.
- Universal design benefits everyone.

BREAK (15 min)

Back at :55

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.
https://www.youtube.com/watch?v=-ApVt04rB4U Put your feedback in the Etherpad. Organize your feedback along two axes: positive vs. opportunities for growth (sometimes called "negative") and content (what was said) vs. presentation (how it was said).

- Abhishek: presentation: instructor is probably rushing through topics and not fully focussed, growth - use better and more encouraging words, content - the overall topic was clear to me
- Airined:
    - Positive vs opportunities for growth- As feedback, I would advice the instructor to explain in a more paused way.
    - Content: In terms of content, the instructor defines concepts as he understands but not from a novice perspective.
- Ashley: <u>Negative</u>: Content: A lot of dismissive phrases or unexplained jargon (e.g., "Even excel users can understand this," "Which of course is polymorphic," "Trust me"). Delivery: Instructor seems distracted, not particularly excited about being there. The code is impossible to see on the screen. <u>Positive</u>: Content: Some links between content (e.g., "a function is another data object"...). Delivery: Makes a mistake! But then the instructor doesn't explain how they fixed it or what

happened.

- Brenda: Content: Has context, topic is clear, defining functions.Delivery, Negative - intimidating phrases 'Could you sit down', use of words like of course, simple stuff,trust me, using phone: Presented in a very complex manner, with use of jargon.
- Dennis: Content: difficult to read the screen. Presentation: presentation felt intimidating and maybe content for someone who is already familiar with this topic, rather than a novice.
- Doschmund: Content - Small letters
    - Delivery - Intimidating start, No eye contact and not checking learners for engagement, demotivating words, distraction
    - Positve - Live coding, moving around
- Elizabeth: Positive: Starting out with teaching, negative: bad introduction and assumming that people know how simple a task is. A bit of agitation. Tiny fonts.
- Erika: Delivery/opportunity for improvement: patience with learners, reducing own distractions, language of "just" and "really simple"; contet/opportunity for improvement: use of technical examples and descriptions, skipping over error troubleshooting; delivery/positive: tries to simplify learning, live coding; content/positive: knowledgeable
- Federico: The presentation seems to have only opportunities for growth: he was going to fast, not repeating, not asking questions, non looking at the audience. Content: I would not know.
- Felipe. **Content**. *Positive*: Of course the teacher excels at the knowledge that is trying to deliver. Opportunities to improve: The content could be more close to learners experience . //// **Presentation**. *Positive*: Opportunities to improve: little intimidating, simplifies the learning.
- Givanna:
    - Content:
        - Negative - not enough planning (he seems to be making things up as he goes), small letters.
        - Positive - he is an expert in programming.
    - Delivery:
        - Negative - using demotivating word ("This is simple stuff"), using jargons (yes it is flexible binding, polymorphism) which not everyone understand - he has to say for those not familiar with functions?
- Hanna:
    - positive -- opportunity for growth
        - Positive: did invite questions
        - Opprtunities for Growth: "which of course is polymorphic" was a bit demotivating and not inclusive for those who maybe don't know that term
    - content -- presentation
        - speaking quite quickly
        - screen is quite small - there is an abundance of white space so zooming in looks possible
- Jean:
    - opportunities for growth - poor tone/not respectful "Could you sit down? Now please" "really simple stuff" "even an Excel user can understand this"
    - content: in presentation of function, didn't explain anything about the content of the function (critically, what does `type` do?); used words like "instantiate" and "invoking" without explanation
    - presentation: text on screen waaaay to small
- Jeffrey: Presentation: this information is presented way too quickly. Learners would benefit greatly by a much slower presentation. Also, multiple uses of the word, "just". Also, disempowering the learners, "For those of you who haven't seen functions before, trust me...this is

what you expect."

- Kudzayi: he uses a lot of complicated language to explain his concepts and often seems to be answering his own questions in his head without elaborating to his learners. He is not engaging his students or sharing any examples and has an air of arrogance to him. He does seem to know what he is talking about
- Lisa: Positive: knows what he's talking about
- opportunities for growth: language- "just" "simply" " of course", etc., went too fast with jargon
- Content:
- Presentation: the very first phrase was agressive,
- Margherita: positives: // opportunities: aggressive and diminutive // contents: the impression is a lot of content squeezed in a very short space // presentation: it would have probably been better to use larger fonts in the screen
- Paul: **On content and delivery** knows his stuff **On well and unwell** sometimes doesn't have emaotions in check, uses jargon, doesn't face the audience
- Rongbo: the deliveray is too fast and it seems that he's not very familiar with what he's teaching. He used a lot of de-motivating languages; he only talks to himself but not the audience. The font of his slides is too small and I'm not sure people can see it.
- Wilson: Content: "def" fits well in an introductory class for Python. However polymorphism is way too advanced for such a class. Delivery: The instructor admits mistakes, which is good. The instructor uses lots of demotivating languages, e.g., simple stuff, you don't need to know this.
- Yashar: Content was presented in a non-interactive manner and rushed (super fast). Language was not inclusive. No eye contact with audience. Assumed participants have prior knowledge. Never asked for feedback until the very end.

1. Split into groups of three.
2. Individually, spend 5 minutes preparing a 90-second introduction to the topic of the lesson episode you chose before the start of the training course. You will not be live coding.
3. Get together with your group and have one person teach their segment to the group. Keep a strict time limit of 90 seconds per person (one person should be responsible for the timekeeping).
4. After the first person has finished teaching, share feedback. The person who performed should start by offering feedback on themselves. The timekeeper should help to keep feedback to about 5 minutes per person to ensure everyone has time to perform and discuss.
5. Rotate roles and repeat steps 3 & 4
6. Return to the main group and briefly summarize the feedback you received in the Etherpad.

- Abhishek: explained outline clearly, but went into too much detail too soon, mention the utility/importance of tools that we will learn
- Airined: The presentation could improve in terms of adding elements such as analogies, and explaining the topic as if I were explaining to a family member. In terms of positive feedback, it was good that I provided the objectives at the begining.
- Ashley: Bit too much personal introduction for an introduction of this length; talking a bit fast. Good: provided larger context.
- Brenda: The introduction was long, I did not manage the time well and did not finish the outline. Good: Appreciation of presence of participants from different time zones, it was well paced and easy to follow.
- Dennis: A good outline of what will happen. Nice that personal expereince was included.
- Doschmund:

    - Introduction to the content was good, and slow down. Better planning would have helped the delivery
- .Elizabeth: Introduction should be a bit slower . Overall, good as I was able to state the outlines of the course, expectations and prerequisites.
- Erika: Positive: clear outlining of what will be covered and how steps connect; I found the introduction of 90 seconds to be chlallening --more than the very basic but not enough to get into the details or a lot of motivations
- Federico: I tried to repeat and break down concepts in order to reinforce the understanding and learning process. This was appreciated. I was a bit hesitatant and improvising.
- Felipe: **Positive**: presentation and introduction structured, overall view of the lesson was useful. **Negative**: too plain entonation and intention; an extra difficulty is to have that emphasis in a foreign language (my native language is spanish). I used too much technical terms to describe the lesson (too much details) and that can be overwhelming for a novie learner. A good idea is to simplify the language and/or provide more details about the utility of the different functions.
- Givanna:
    - Negative:
        - Not good in formulating sentence. A bit all over the shop.
        - Not introducing myself in the beginning
        - Speed up too much towards the end.
        - Specific examples that are not relevant to the course (like changing data type).
    - Positive:
        - Good explanation and lay out the pre-requisite.
        - Clearly explained the objectives of the lesson.
- Hanna:
    - Notes on Ashley
    - - really great introduction, enjoyed having some personal tidbits to latch onto and a bit of discussion on where they are with their personal journey in Python
    - - well delivered, clear, and concise
    - Notes on Givanna
    - - nice explanation of what we are going to cover/objectives
    - - well delivered, clear, and concise
    - - covered pre-requisites that were expected to be familiar with
    - - talking a bit quickly; maybe a few too many examples of data types
    -
    - Notes on Hanna (myself)
    - - should have done more with introduction (perhaps introduce myself)
    - - provide details on pre-requisites
    - - for the objectives we need to link them to a larger context
- Jean: Postitive: pace was about right; content was clear. Improvements: give more overview of the course content and set expectations for what the learning goals should be.
- Jeffrey: Positive feedback was good use of analogy. Clear learning objectives. Room for improvement: went over the 90 seconds by about 30%.
- Kudzayi: I like that you brought up the utility of variables, i.e., they can be reused in various parts of the program. Nice pace and rythm
- Lisa: feedback received: communication about what will happen throughout the course was goo. I also said that I talked too fast, which i think i was trying to get through it all in 90 minutes
- Margherita: feedback received: well structured, good to put people at ease to stop the speaker and ask questions. Probably too many concepts squeezed into a very small time (too complex). Better to introduce yourself with two names. Assuming too much that people already know some of the

technicalities I mentioned
- Paul: I didn't seem to get any negative feedback, my presentation was believe was perceived fine, caught attention, and was inclusive.
- Rongbo: well-structured but too many concepts, terms, and a bit overcomplex, sometimes too specific
- Wilson: Feedback for Wilson's introduction to "Repeating Actions with Loops":
    - Too rushed (45 seconds total time).
    - Could take time to repeat some key points.
    - Could take time to explain application of loops
- Yashar: Provide more background information before starting to talk about the topics we want to cover. Privode more detailed overview of the materials.

Prompt: Identify at least one specific change you will make to your teaching based on this feedback. Describe your change in the Etherpad.

- Abhishek: talk more slowly and calmly, space out details, keep mentioning the "why we are learning this?" often
- Airined: To improve, I will work on practicing the presentation, and deliver as in presentations fro conferences.
- Ashley: Talk more slowly, carefully choose content for a short intro.
- Brenda: Make the introduction short and precise, better management of time.
- Dennis: Be careful in using words likes "complex" or "more complicated" as this may disuade participation or engagement if presented in certain contexts.
- Doschmund: Think about sequencing sentences better and speak slowly (I never realised I speak fast :) and I always thought I talk too slowly)
- Elizabeth: : slow pacing, a bit more detailed description of the course
- Erika: Work on the flow between concepts or tasks, it felt a bit like I was listing discrete tasks which might be overwhelming. Trying more of a storytelling approach
- Federico: Have the way in which I will break down concepts better prepared
- Felipe: Simplify the language, in order to not provide too technical jargon. Give more emphasis to the language, so not to speak "plain".
- Givanna: Prepare better so I can be more concious of what I deliver and can slow down.
- Hanna: 1) definitely start by introducing myself next time; 2) expand the discussion of objectives to relate them to examples that learners can identify with more easily than just spouting the objectives (address the why this is helpful and/or how this can be used).
- Jean: Make sure I share the learning goals for the workshop at the outset so people know what to expect. Overview of expectation: what prior knowledge is expected (if any) and what is not expected - and state that learners should feel free to ask if they hear a term that isn't familiar to them.
- Jeffrey: Reduce the learning objectives and syllabus to something that's coverable in the time alloted.
- Kudzayi: More preparedness so that I am concise with my thoughts, style and teaching objectives
- Lisa: talk. slower. And also introduce myself.
- Margherita: start the intro with a "in the previous episodes" sketch, to create a moment in which attendands can check into possible bits missed or forgotten from the previous session
- Paul: I'll highlight the concepts in more detail then I did. I'll reheaerse as well, and have tight language.
- Rongbo: simplify the recap section, and make it easier to follow, less technical terms, and more accessible for studnets

- Wilson: I'd time a rehearsed lesson to get a better feel for how long it may take.
- Yashar: Start a bit slower and focus on the overalll learning objectives more than anything else in the beginning.

1. Read about [centrally-organized and self-organized workshops] (https://carpentries.org/workshops/#workshop-organising) and our handbook content on [Teaching and Hosting Workshops](https://docs.carpentries.org/topic_folders/hosts_instructors/index.html) -- be sure to click through to some of the associated checklists. These summarize commonly asked questions about organizing and running workshops.
  + When you arrive for the next part, we will ask you to add one question about our operations to a list.
  + We will then do our best to answer all of those questions during the day.
2. Prepare for the [live coding exercises](17-live/).
  If you have not already done so per the pre-workshop instructions, pick an episode from an existing Software Carpentry,
  Data Carpentry, or Library Carpentry lesson and
  read through it carefully.
  In the next two parts, you will use this to practice live coding/participatory instruction for 3 minutes.
  Remember, imperfect presentations can generate useful feedback!
  If you have not yet selected an episode to focus on and would like a recommendation, consider one of the following:
  - Data Carpentry
    - [Faceting and Clustering in OpenRefine](https://datacarpentry.org/OpenRefine-ecology-lesson/02-exploring-data/index.html)
    - [Basic Queries in SQL](https://datacarpentry.org/sql-ecology-lesson/01-sql-basic-queries/)
    - [Starting with Data in R](https://datacarpentry.org/R-ecology-lesson/02-starting-with-data.html)
    - [Starting with Data in Python](https://datacarpentry.org/python-ecology-lesson/02-starting-with-data/)
  - Library Carpentry
    - [Working with Files and Directories in the Unix Shell](https://librarycarpentry.github.io/lc-shell/03-working-with-files-and-folders/index.html)
    - [Faceting and filtering in Open Refine](https://librarycarpentry.github.io/lc-open-refine/04-faceting-and-filtering/index.html)
    - [For loops in Python](https://librarycarpentry.github.io/lc-python-intro/12-for-loops/index.html)
  - Software Carpentry
    - [Working with Files and Directories in the Unix Shell](https://swcarpentry.github.io/shell-novice/03-create/)
    - [Tracking Changes in Git](https://swcarpentry.github.io/git-novice/04-changes/)
    - [Selecting Data in SQL](https://swcarpentry.github.io/sql-novice-survey/01-select/)
    - [Repeating Actions with Loops in Python](https://swcarpentry.github.io/python-novice-inflammation/05-loop/)
    - [Exploring Data Frames in R](https://swcarpentry.github.io/r-novice-gapminder/05-data-structures-part2/)

Give Us Feedback (1 minute)

Write one thing you learned this morning that you found useful on one of your sticky notes, and one question you have about the material on the other.

https://forms.gle/3jCfqn8CWJHMRs6z6