Ten Simple Rules for Creating an Effective Lesson

Greg Wilson[1,*]
1 RStudio, Inc. / greg.wilson@rstudio.com

* Corresponding author.

Abstract

FIXME: abstract

Author Summary

FIXME: author summary

Introduction

There are many kinds of lessons, both formal and informal, from seconds long to lifelong. Most people have sat (or suffered) through hundreds of these, but have never been shown how to design ones that are effective. These ten simple rules for creating lessons are:

- based on current educational research [1–3],
- filtered by what can be done by non-specialists with limited time and resources [4–6], and
- prioritized by experience teaching and training people to teach together [7–9].

1 Use Learner Personas to Define Your Audience

The first step in creating a good lesson is figuring out who the audience is. One way to do this is to make up biographies of two or three target learners. This technique is borrowed from user interface designers, who create short profiles of typical users to help them think about their audience, and the profiles themselves are called *personas*.

Learner personas have five parts:

1. the person's general background,
2. what they already know,
3. what *they* think they want to do (as opposed to what someone who already understands the subject thinks),
4. how the lesson will help them, and
5. any special needs they might have.

A learner persona for a weekend introduction to programming aimed at college students might be:

1. Jorge has just moved from Costa Rica to Canada to study agricultural engineering. He has joined the college soccer team, and is looking forward to learning how to play ice hockey.
2. Other than using Excel, Word, and the Internet, Jorge's most significant previous experience with computers is helping his sister build a WordPress site for the family business back home in Costa Rica.
3. Jorge needs to measure properties of soil from nearby farms using a handheld device that sends logs in a text format to his computer. Right now, Jorge has to open each file in Excel, crop the first and last points, and calculate an average.
4. This workshop will show Jorge how to write a little Python program to read the data, select the right values from each file, and calculate the required statistics.
5. Jorge can read English well, but still struggles sometimes to keep up with spoken conversation (especially if it involves a lot of new jargon).

Rather than writing new personas for every lesson or course, instructors often create and share a handful that cover everyone they hope to teach, then pick a few from that set to describe who particular material is intended for. Used this way, personas become a convenient shorthand for design issues: when speaking with each other, teachers can say, "Would Jorge understand why we're doing this?" or, "What installation problems would Jorge face?"

Personas help you remember one of the most important rules of teaching: *you are not your learners*. The people you teach will almost always have different backgrounds, different capabilities, and different ambitions than you; personas help you keep your lessons focused on what they need rather than on what your younger self might have wanted.

2 Encourage Effective Learning Strategies

Some learning strategies are provably more effective than others [10–12], so lessons should be designed to encourage their use. As summarized in [13, 14], the six most important are:

Spaced Practice: Ten hours of study spread out over five days is more effective than two five-hour days, and far better than one ten-hour day. You should therefore create lessons and exercises that include some older material in each new lesson. According to [15], "The lectures that predominate in face-to-face courses are relatively ineffective ways to teach, but they probably contribute to spacing material over time, because they unfold in a set schedule over time. In contrast, depending on how the courses are set up, online students can sometimes avoid exposure to material altogether until an assignment is nigh."

Retrieval Practice: Researchers now believe that the limiting factor for long-term memory is not retention (what is stored), but recall (what can be accessed). Recall of specific information improves with practice, so outcomes in real situations can be improved by taking practice tests or summarizing the details of a topic from memory and then checking what was and wasn't remembered. For example, [16] found that repeated testing improved recall of word lists from 35% to 80%.

Interleaving: One way to space retrieval practice is to interleave study of different topics: instead of mastering one subject, then the next, then a third, shuffle the order. Even better, switch up the order: A-B-C-B-A-C is better than

A-B-C-A-B-C, which in turn is better than A-A-B-B-C-C [10]. This is effective because interleaving fosters creation of more links between different topics, which in turn increases retention and recall.

Elaboration: Having learners explain things to themselves as they go along improves understanding and recall. One way to do this is to follow up each answer on a practice quiz with an explanation of why that answer is correct, or conversely with an explanation of why some other plausible answer isn't. Another is to have learners explain how a new idea is similar to or different from one they have seen previously.

Concrete Examples: One specific form of elaboration that is useful enough to deserve its own heading is the use of concrete examples. As discussed in the rule on concreteness fading, every statement of a general principle should be accompanied by one or more examples of its use, or conversely take each particular problem and list the general principles it embodies. [17] found that interleaving examples and definitions made it more likely that learners would remember the latter correctly.

Another approach is to teach by contrast, i.e., to show learners what a solution is *not*, or what kind of problem a technique *won't* solve. For example, when showing children how to simplify fractions, it's important to give them a few like 5/7 that can't be simplified so that they don't become frustrated looking for answers that don't exist.

Dual Coding: Different subsystems in our brains handle and store linguistic and visual information, and if complementary information is presented through both channels, then they can reinforce one another. However, learning is more effective when the same information is *not* presented simultaneously in two different channels [18, 19], because then the brain has to expend effort to check the channels against each other.

3 Write Summative Assessments to Set Concrete Goals

Summative assessment is something done at the end of a lesson to tell whether the desired learning has taken place: a driving test, performance of a piece of music, a written examination, or something else of that kind. Summative assessments are usually used as gates (e.g., "Is it now safe for this person to drive on their own?"), but they are also a good way to clarify the learning objectives for a lesson. "Understand linear regression" is hopelessly vague; a much better way to set the goal for a lesson is:

> Write a short R script that reads the tabular data in `housing.csv` and uses the `lm` function to calculate a regression coefficient relating house price to purchaser age.

This is better because it gives the lesson author a concrete goal to work toward: nothing goes in the lesson except what is needed to complete the summative assesments. This helps reduce content bloat, and also tells the author when the lesson is done.

Writing summative assessments early in the lesson design process also helps ensure that outcomes are actually checkable. Since telepathy is not yet widely available, it is impossible for instructors to know what learners do and don't understand. Instead, we must ask them to demonstrate that they're able to do something that they couldn't do without the desired understanding.

Finally, creating summative assessments early can help authors stay connected to their learners' goals. Each summative assessment should embody an *authentic task*, i.e., something that an actual learner actually wants to do. Early on, authentic tasks should be learners' own goals; as they advance and are able to make sense of generalizations, these tasks may be extensions or generalizations of earlier solutions.

Continuing with the statistical example above, calculating a regression coefficient may be an authentic task for someone who already knows enough statistics to understand what such coefficients are good for. If the intended learners are not yet that experienced (which you will know from your personas), this exercise could be extended to have them make some sort of judgment based on the coefficient in order to answer the deeper question, "Why bother?"

4 Write Formative Assessments for Pacing, Design, Preparation, and Reinforcement

The counterpoint to summative assessment is *formative assessment*, which is checks used while learning is taking place to form (or shape) the teaching. Asking learners for questions is a common, but relatively ineffective, kind of formative assessment. What works better is to give them a short problem—one that can be done in 1–2 minutes so as not to derail the flow of the lesson, and which will help them uncover and confront their misconceptions about the topic being taught.

Checking in with learners this way every 10–15 minutes accomplishes several things:

Pacing: Asking, "Does everyone understand?" almost always produces false positives. In contrast, if any substantial fraction of your learners cannot do a formative assessment correctly, you know right then and there that you need to re-explain the most recent material. When you start doing this, you will feel like you're going more slowly, but that's because you will now be teaching at the speed at which your audience can learn rather than the speed at which you can talk.

Design: Creating formative assessments that build toward a lesson's summative assessment gives you a structure for your lesson. Returning to the regression example, the summative assessment tells you that you should have exercises along the way in which learners load CSV data, use the `lm` function with appropriate parameters, and interpret the result. Writing a few minutes of material for each of these subjects is less intimidating than trying to explain the whole topic at once.

Preparation: Formative assessments give learners practice with the concepts, methods, and tools they will use when doing the lesson's summative assessment, and tells them where to focus their revision. Switching from statistics to music, if a violinist is able to do the bowing and fingering exercises for a piece, but is struggling with the rhythmic patterns, that tells her where she should spend her study time.

Reinforcement: Learners remember things better if they use material right away, and having formative assessments during the lesson does this.

Scope: Breaking a summative assessment down into parts and creating formative assessments for each usually shows you that you are trying to cram too much into one lesson. Writing assessments is therefore iterative, as early drafts of summative assessments are re-scoped to only require as much material as can plausibly be covered.

5 Design for Peer Instruction

No matter how good a teacher is, she can only say one thing at a time. How then can she clear up many different misconceptions in a reasonable time? The best solution developed so far is *peer instruction.* Originally created by Eric Mazur at Harvard [20], it has been studied extensively in a wide variety of contexts (e.g., [21, 22]).

Peer instruction is essentially a scalable way to provide one-to-one mentorship. It interleaves formative assessment with student discussion as follows:

1. Give a brief introduction to the topic, either in class or in out-of-class reading.
2. Give learners a multiple choice question (MCQ).
3. Have all the students vote on their answers to the MCQ.
   (a) If the students all have the right answer, move on.
   (b) If they all have the same wrong answer, address that specific misconception.
   (c) If they have a mix of right and wrong answers, give them several minutes to discuss those answers with one another in small groups (typically 2–4 students) and then reconvene and vote again.

The questions posed to learners don't have to be MCQs: matching terms to definitions can be equally effective, as can Parsons Problems (in which they are given the jumbled parts of a solution and must put them in the right order [23]). Whatever mix is used, the lesson must build toward them, and the question must probe for conceptual understanding and misconceptions (rather than check simple factual knowledge).

Group discussion significantly improves students' understanding because it forces them to clarify their thinking, which can be enough to call out gaps in reasoning. Re-polling the class then lets the teacher know if they can move on, or if further explanation is necessary. A final round of additional explanation and discussion after the correct answer is presented gives students one more chance to solidify their understanding.

But could this be a false positive? Are results improving because of increased understanding during discussion, or simply from a follow-the-leader effect? [24] tested this by following the first question with a second one that students answer individually and found that peer discussion actually does enhance understanding, even when none of the students in a discussion group originally knew the correct answer.

It is important to have learners vote publicly so that they can't change their minds afterward and rationalize it by making excuses to themselves like "I just misread the question". Some of the value of peer instruction comes from having their answer be wrong and having to think through the reasons why. This is called the *hypercorrection effect* [25]. Most people don't like to be told they're wrong, so it's reasonable to assume that the more confident someone is that the answer they've given in a test is correct, the harder it is to change their mind if they were actually wrong. However, it turns out that the opposite is true: the more confident someone is that they were right, the more likely they are not to repeat the error if they are corrected.

6 Use Worked Examples and Concreteness Fading

A worked example is a step-by-step demonstration of how to solve a problem or do some task. By giving the steps in order, the instructor reduces the learner's cognitive load, which accelerates learning [26, 27].

However, worked examples become less effective as learners acquire more expertise [28, 29], a phenomenon known as the *expertise reversal effect.* In brief, as learners build their own mental models of what to do and how to do it, the detailed step-by-step breakdown of a worked example starts to get in the way. This again is a reason to develop learner personas before writing lessons, and is also the reason why tutorials and manual pages both need to exist: what's appropriate for a newcomer is frustrating for an expert, while what jogs an expert's memory may be incomprehensible to a novice.

One powerful way to use worked examples is to present a series of *faded examples* [30]. The first example in the series is a complete use of a problem-solving strategy; each subsequent example gives the learner more blanks to fill in. TThe material that isn't blank is often referred to as *scaffolding*, since it serves the same purpose as the scaffolding set up temporarily at a building site.

Faded examples can be used in almost every kind of teaching, from sport and music to contract law. Someone teaching Python programming might use them by first explaining how to calculate the total length of a list of words:

```
# total_length(["red", "green", "blue"]) => 12
define total_length(list_of_words):
    total = 0
    for each word in list_of_words:
        total = total + word.length()
    return total
```

and then asking learners to fill in the blanks in this (which focuses their attention on control structures):

```
# word_lengths(["red", "green", "blue"]) => [3, 5, 4]
define word_lengths(list_of_words):
    list_of_lengths = []
    for each ____ in ____:
        list_of_lengths.append(____)
    return list_of_lengths
```

The next problem might be this (which focuses their attention on updating the final result):

```
# join_all(["red", "green", "blue"]) => "redgreenblue"
define join_all(list_of_words):
    joined_words = ____
    for each ____ in ____:
        ____
    return joined_words
```

Learners would finally be asked to write an entire function on their own:

```
# make_acronym(["red", "green", "blue"]) => "RGB"
define make_acronym(list_of_words):
    ____
```

At each step, learners have one new problem to tackle, which is less intimidating than a blank screen or a blank sheet of paper. Faded examples also encourage learners (and instructors) to think about the similarities and differences between various approaches.

Worked examples are themselves an example of *concreteness fading* [31, 32], which describes the process of starting lessons with things that are specific or tangible and then explicitly and gradually transitioning to more abstract and general concepts. Concreteness fading:

1. helps learners understand abstract symbols in terms of well-understood concrete objects,
2. lets them leverage personal experience to ground abstract thinking,
3. gives them a store of examples and mental images that they can fall back on when abstract symbols and reasoning fail, and
4. help learners figure out what is specific to particular examples and what is generalizable across all problems of a certain kind.

One way to remember this strategy is the acronym PETE (Problem, Explanation, Theory, Example), which encourages instructors to:

- describe an authentic problem that the lesson will solve,
- work through a solution to that problem,
- explain the general theory that underpins that solution, and
- work through a second example so that learners will understand which parts generalize.

7 Show How to Detect, Diagnose, and Correct Common Mistakes

It is almost oxymoronic to say that learners spend a lot of their time trying to figure out what they've done wrong and fixing it: after all, if they knew and they had, they would already have moved on to the next subject. Most lessons devote little time to detecting, diagnosing, and correcting common mistakes, but doing this will accelerate learning—not least by reducing the time that learners spend feeling lost and frustrated.

In Carroll et al's "minimal manual" approach to training materials, every topic is accompanied by descriptions of symptoms learners might see, their causes, and how to correct them [33]. When studying second language acquisition, [34] identified six ways in which instructors can correct learners' mistakes:

Explicit correction: clearly indicating that the learner is incorrect and provide the correct form.

Recasting: repeat the learner's response with the mistake or mistakes corrected.

Clarification request: indicate that the learner's answer is incorrect (e.g., by saying, "Are you sure?") but leave the correction open-ended.

Metalinguistic clues: pose leading questions (e.g., "Do we need the absolute error or the relative error here?")

Elicitation: provide the first part of the correct answer as a prompt and require the learner to fill in the rest.

Repetition: repeats the learner's error, drawing attention to it but leaving the correction up to them.

All of these can be used preemptively during the design of lessons. For example, an introduction to chemical reactions could present an incomplete calculation of enthalpy and ask the learner to fill it in (elicitation) or present the complete calculation with errors, then draw attention to those errors and correct them one by one (recasting). All of these strategies provide retrieval practice by requiring learners to use what they have just learned, and encourage metacognition by requiring them to reflect on the the limits and applicability of that knowledge.

8 Motivate and Avoid Demotivating

One of the strongest predictors of whether people learn something is their *intrinsic motivation*, i.e., their innate desire to master the material. The term is used in contract with *extrinsic motivation*, which refers to behavior driven by rewards such as money, fame, and grades. As [35] describes, the biggest motivators for adult learners are their sense of agency (i.e., the degree to which they feel that they're in control of their lives), the utility or usefulness of what they're learning, and whether their peers are learning the same things. Letting people go through lessons at the time of their own choosing, using authentic tasks, and working in small groups speak to each of these factors.

Conversely, it is very easy for educators to demotivate their learners by being unpredictable, unfair, or indifferent. If there is no reliable relationship between effort and result, learners stop trying (a particular case of a broader phenomenon called *learned helplessness*). If the learning environment is slanted to advantage some people at the expense of others, everyone will do less well on average [36], and if the lessons make it clear that the teacher doesn't care if people learn things or not, learners will mirror that indifference.

One way to tell if learners are motivated or not is to look at the incidence of cheating. In classrooms, it is usually not a symptom of moral failing, but a rational response to poorly-designed incentives. As reported in [37], some things that educators do that unintentionally encourage cheating include:

- setting the cost of failure very high,
- relying on single assessment mechanisms like multiple-choice tests, and
- using arbitrary grading criteria.

Eliminating these from lessons doesn't guarantee that learners won't cheat, but does reduce the incidence. (And despite what many educators believe, cheating is no more likely online than in person [38].)

9 Make Lessons Inclusive

Inclusivity is a policy of including people who might otherwise be excluded. In STEM education, it means making a positive effort to be more welcoming to women, under-represented racial or ethnic groups, people with various sexual orientations, the elderly, the physically challenged, the economically disadvantaged, and others.

One axis of inclusive lesson design is physical: provide descriptive text for images and videos to help the visually challenged, closed captions for videos to help those with hearing challenges, and so on. Another axis is social:

- Use gender-neutral pronouns (e.g., a singular "they") or alternate between male and female pronouns.

- Use culturally varied names in examples (e.g., Aisha and Boris rather than Alice and Bob).
- Avoid examples based on over-simplified or exclusionary views of gender and orientation, such as assuming that there are only two genders, that gender is fixed throughout a person's life, or that marriage is always between people of unlike gender.

Committing fully to inclusive teaching may mean fundamentally rethinking content. For example, [39] explored two strategies for making computing education more culturally inclusive:

Community representation highlights students' social identities, histories, and community networks using after-school mentors or role models from students' neighborhoods, or activities that use community narratives and histories as a foundation for a computing project.

Computational integration incorporates ideas from the learner's community, e.g., reverse engineering indigenous graphic designs in a visual programming environment.

The major risk of the community representation approach is shallowness, e.g., using computers to build slideshows rather than do any real computing. The major risk with computational integration is cultural appropriation, e.g., using practices without acknowledging origins. When in doubt, ask your learners and members of their community what they think you ought to do and give them control over content and direction.

The most important step is to stop thinking in terms of a *deficit model*, i.e., to stop thinking that the members of marginalized groups lack something and are therefore responsible for not getting ahead. Believing that puts the burden on people who already have to work harder because of the inequities they face, and (not coincidentally) gives those who benefit from the current arrangements an excuse not to look at themselves too closely.

10 Design for Collaboration with Other Instructors

Instructors often scour the web for ideas, and it's common for people to inherit courses from previous instructors. What is far less common is collaborative lesson construction, i.e., people taking material, improving it, and then offering their changes back to the community. This model has served the open source software community well, and as [7] describes, it works equally well for lessons—provided that materials are designed to make fine-grained collaboration easy.

The best tool we have today for large-scale ad hoc collaboration is version control. Originally developed by programmers for managing the source code of large projects, it gives each contributor their own working copy of the material, but provides a way for them to submit changes to a central store, have them reviewed, make further modifications, and finally merge them into the core to strengthen it [40].

Version control's greatest strengths are its scalability (some projects have had thousands of contributors) and its openness (anyone can offer changes). Its greatest weakness is that widely-used systems like Git are designed to handle text files, and struggle with structured document formats like Microsoft Word or PowerPoint. As a result, people who wish to use it for lessons usually have to rely on LateX or some variation of HTML or Markdown for their notes and slides, which is a larger barrier to uptake than many experts realize (or are willing to acknowledge).

One key enabler of collaborative lesson construction is licensing. We strongly recommend using one of the Creative Commons family of licenses, since they have been carefully vetted and are widely understood.

Conclusion

Following the ten rules laid out above doesn't guarantee that your lessons will be great, but it will help ensure that they aren't bad. When it comes time to put them into practice, we recommend following something like the reverse design process developed independently by [41–43]:

1. Figure out who your learners are and what their goals are.
2. Create the summative assessment for the lesson to give yourself a target.
3. Itemize the knowledge and skills that assessment relies on, and create formative assessments to check on each while learning is taking place.
4. Order those formative assessments in a way that respects their dependencies, i.e., so that they build on each other.
5. Estimate the time required to cover each topic and perform its related formative assessment, then cut material that there isn't time for.
6. Write lessons to connect each formative assessment to the next (which is usually much easier than writing an entire lesson at once).
7. Double-check your language and examples to ensure that they address your learners' goals and won't demotivate them.
8. Derive learning objectives and key points from the lesson to share with your learners and co-instructors. The former make the lesson findable, while the latter give you and your co-instructors a quick way to check what the lesson actually covers.
9. Put everything online for other people to download, modify, and contribute to.

References

1. Nuthall G. The Hidden Lives of Learners. NZCER Press; 2007.
2. Ambrose SA, Bridges MW, DiPietro M, Lovett MC, Norman MK. How Learning Works: Seven Research-Based Principles for Smart Teaching. Jossey-Bass; 2010.
3. Brown NCC, Wilson G. Ten Quick Tips for Teaching Programming. PLoS Computational Biology. 2018;14(4). doi:10.1371/journal.pcbi.1006023.
4. Huston T. Teaching What You Don't Know. Harvard University Press; 2012.
5. Lang JM. Small Teaching: Everyday Lessons from the Science of Learning. Jossey-Bass; 2016.
6. Rice GT. Hitting Pause: 65 Lecture Breaks to Refresh and Reinforce Learning. Stylus Publishing; 2018.

7. Devenyi GA, Emonet R, Harris RM, Hertweck KL, Irving D, Milligan I, et al. Ten Simple Rules for Collaborative Lesson Development. PLOS Computational Biology. 2018;14(3). doi:10.1371/journal.pcbi.1005963.

8. Wilson G. Software Carpentry: lessons learned. F1000Research. 2016;doi:10.12688/f1000research.3-62.v2.

9. Wilson G. Teaching Tech Together. Lulu; 2018.

10. Rohrer D, Dedrick RF, Stershic S. Interleaved Practice Improves Mathematics Learning. Journal of Educational Psychology. 2015;107(3):900–908. doi:10.1037/edu0000001.

11. Kang SHK. Spaced Repetition Promotes Efficient and Effective Learning. Policy Insights from the Behavioral and Brain Sciences. 2016;3(1):12–19. doi:10.1177/2372732215624708.

12. Miyatsu T, Nguyen K, McDaniel MA. Five Popular Study Strategies: Their Pitfalls and Optimal Implementations. Perspectives on Psychological Science. 2018;13(3):390–407. doi:10.1177/1745691617710510.

13. Weinstein Y, Madan CR, Sumeracki MA. Teaching the Science of Learning. Cognitive Research: Principles and Implications. 2018;3(1). doi:10.1186/s41235-017-0087-y.

14. Weinstein Y, Sumeracki M, Caviglioli O. Understanding How We Learn: A Visual Guide. Routledge; 2018.

15. Miller MD. Minds Online: Teaching Effectively with Technology. Harvard University Press; 2016.

16. Karpicke JD, Roediger HL. The Critical Importance of Retrieval for Learning. Science. 2008;319(5865):966–968. doi:10.1126/science.1152408.

17. Rawson KA, Thomas RC, Jacoby LL. The Power of Examples: Illustrative Examples Enhance Conceptual Learning of Declarative Concepts. Educational Psychology Review. 2014;27(3):483–504. doi:10.1007/s10648-014-9273-3.

18. Mayer RE, Moreno R. Nine Ways to Reduce Cognitive Load in Multimedia Learning. Educational Psychologist. 2003;38(1):43–52. doi:10.1207/s15326985ep3801_6.

19. Mayer RE. Multimedia Learning. 2nd ed. Cambridge University Press; 2009.

20. Crouch CH, Mazur E. Peer Instruction: Ten Years of Experience and Results. American Journal of Physics. 2001;69(9):970–977. doi:10.1119/1.1374249.

21. Vickrey T, Rosploch K, Rahmanian R, Pilarz M, Stains M. Research-Based Implementation of Peer Instruction: A Literature Review. CBE–Life Sciences Education. 2015;14(1). doi:10.1187/cbe.14-11-0198.

22. Porter L, Bouvier D, Cutts Q, Grissom S, Lee CB, McCartney R, et al. A Multi-Institutional Study of Peer Instruction in Introductory Computing. In: Proc. 2016 Technical Symposium on Computer Science Education (SIGCSE'16). Association for Computing Machinery (ACM); 2016.

23. Parsons D, Haden P. Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. In: Proc. 2006 Australasian Conference on Computing Education (ACE'06). Australian Computer Society; 2006. p. 157–163.

24. Smith MK, Wood WB, Adams WK, Wieman CE, Knight JK, Guild N, et al. Why Peer Discussion Improves Student Performance on In-class Concept Questions. Science. 2009;323(5910):122–124. doi:10.1126/science.1165919.

25. Metcalfe J. Learning from Errors. Annual Review of Psychology. 2016;68(1):465–489. doi:10.1146/annurev-psych-010416-044022.

26. Atkinson RK, Derry SJ, Renkl A, Wortham D. Learning from Examples: Instructional Principles from the Worked Examples Research. Review of Educational Research. 2000;70(2):181–214. doi:10.3102/00346543070002181.

27. Paas F, Renkl A, Sweller J. Cognitive Load Theory and Instructional Design: Recent Developments. Educational Psychologist. 2003;38(1):1–4. doi:10.1207/s15326985ep3801_1.

28. Kalyuga S, Ayres P, Chandler P, Sweller J. The Expertise Reversal Effect. Educational Psychologist. 2003;38(1):23–31. doi:10.1207/s15326985ep3801_4.

29. Kalyuga S. Expertise Reversal Effect and Its Implications for Learner-Tailored Instruction. Educational Psychology Review. 2007;19(4):509–539. doi:10.1007/s10648-007-9054-3.

30. Schwonke R, Renkl A, Krieg C, Wittwer J, Aleven V, Salden R. The worked-example effect: Not an artefact of lousy control conditions. Computers in Human Behavior. 2009;25(2):258–266. doi:10.1016/j.chb.2008.12.011.

31. Goldstone RL, Son JY. The Transfer of Scientific Principles Using Concrete and Idealized Simulations. Journal of the Learning Sciences. 2005;14(1):69–110. doi:10.1207/s15327809jls1401_4.

32. Fyfe ER, McNeil NM, Son JY, Goldstone RL. Concreteness Fading in Mathematics and Science Instruction: a Systematic Review. Educational Psychology Review. 2014;26(1):9–25. doi:10.1007/s10648-014-9249-3.

33. Carroll J. Creating Minimalist Instruction. International Journal of Designs for Learning. 2014;5(2). doi:10.14434/ijdl.v5i2.12887.

34. Lyster R, Ranta L. Corrective Feedback and Learner Uptake: Negotiation of Form in Communicative Classrooms. Studies in Second Language Acquisition. 1997;19(1):37–66.

35. Wlodkowski RJ, Ginsberg MB. Enhancing Adult Motivation to Learn: A Comprehensive Guide for Teaching All Adults. Jossey-Bass; 2017.

36. Wilkinson R, Pickett K. The Spirit Level: Why Greater Equality Makes Societies Stronger. Bloomsbury Press; 2011.

37. Lang JM. Cheating Lessons: Learning from Academic Dishonesty. Harvard University Press; 2013.

38. Beck V. Testing a Model to Predict Online Cheating—Much Ado About Nothing. Active Learning in Higher Education. 2014;15(1):65–75. doi:10.1177/1469787413514646.

39. Lachney M. Computational Communities: African-American Cultural Capital in Computer Science Education. Computer Science Education. 2018; p. 1–22. doi:10.1080/08993408.2018.1429062.

40. Blischak JD, Davenport ER, Wilson G. A Quick Introduction to Version Control with Git and GitHub. PLOS Computational Biology. 2016;12(1):e1004668. doi:10.1371/journal.pcbi.1004668.

41. Wiggins G, McTighe J. Understanding by Design. Association for Supervision & Curriculum Development (ASCD); 2005.

42. Biggs J, Tang C. Teaching for Quality Learning at University. Open University Press; 2011.

43. Fink LD. Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses. Jossey-Bass; 2013.