Amsterdam University Press

TIJDSCHRIFT VOOR GENDERSTUDIES 27.2/3 (2024) 196-217
HTTPS://DOI.ORG/10.5117/TVGN2024.2-3.007.GUES

Teaching coding inclusively

If this, then what?

Olivia Guest
Donders Institute for Brain, Cognition and Behaviour, School of Artificial Intelligence, Radboud University
olivia.guest@donders.ru.nl

Samuel H. Forbes
Department of Psychology, Durham University
samuel.forbes@durham.ac.uk

Abstract

We present our stance on teaching programming with the aim of increasing reflexivity amongst university educators through dissecting and destroying pervasive anti-pedagogical gendered framings. From the so-called male geek trope that dominates Global North/Western perceptions of technology to the actively anti-feminist stances such demographics espouse: programming has a sexism problem. Herein, we touch on how and why programming is so gendered in the present; we expound on how we manage this in our classrooms and in our mentorship relationships; and we explain how to keep doing so moving forwards. Through weaving examples of programming into the text, it is demonstrated that basic coding concepts can be conveyed with little effort. Additionally, example dialogues – exchanges between teachers and students and between educators – are worked through to counteract inappropriate or harmful framings. Finally, we list some ground rules, concrete dos and don'ts, for us to consider going forwards. Ultimately, as educators, we have a twofold obligation, for our students to a) learn programming, and for them to b) unlearn problematic perceptions of who can code.

Keywords: programming, inclusivity, pedagogy

Scoping the issue

```
// What is the value of z in the following
// JavaScript code snippet?

var x = 5;
var y = 6;
var z = x + y;
```

Programming has a sexism problem.[1] There is a history of women being not only the first programmers and the first computers, but also of them being actively pushed out and expunged from the historical record (Evans, 2020; Hicks, 2017; Lee Shetterly, 2016). There is also a present day highly masculinised view of the field starting from children's perceptions, such as the so-called male geek trope, which dovetails with masculinist ideologies within the tech sector (Birhane & Guest, 2021; Erscoi, Kleinherenbrink, & Guest, 2023; Hermans, 2024; Lewis, Anderson, & Yasuhara, 2016; Margolis & Fisher, 2001; O'Mara, 2022; Salter & Blodgett, 2017; White, 2020). Furthermore, there is the view that the tech sector is the only place where coding skills are relevant – a caricature akin to 'writing is only useful if one wants to be a novelist' (Hermans & Aldewereld, 2017). In turn, this translates into dissuading girls and women from learning to code (Busjahn & Schulte, 2013; De Wit, Hermans, Specht, & Aivaloglou, 2024; Hermans, 2021). The interrelated dynamics of Global North/Western gendered and racialised perceptions of technology and the pedagogical situations in which these perceptions are relevant is our focus herein. Teaching programming must take on these issues against the backdrop of their latent cause: patriarchy.

Given all this, no wonder that women do not appreciate the fun or usefulness of learning coding, and that the layperson has no idea that programming is for everybody. If girls, women, and the feminised generally, are made to believe they are not intelligent or do not belong, or they are sexually harassed and assaulted (Dresden, Dresden, Ridge, & Yamawaki, 2018; Essanhaji, 2023; McKinley, 2018), the problem lies with the educator, institute, and society at large for not challenging gender apartheid (Blum & Frieze, 2005; Goffman, 1977; Lind-Guzik, 2023).

While by no means is it true that everybody needs to code, we make a different case: everybody already knows how to code at least with respect to some basic concepts and everybody who wants to learn should be afforded the same respect to do so. Because our pedagogical focus herein is programming, we are at odds with an existing hyper-masculinised culture, that does not set us on a fair footing to foster deep care for each other's experiences in educational and pastoral contexts. Quite the opposite, as shall be seen: technology culture is interwoven with white supremacy,

1969: 'Space age needleworker "weaves" core rope memory for [Apollo missions'] computers.' (Raytheon, 1969, p. 18)

1962: Mathematicians and programmers, Patsy Simmers, Gail Taylor, Milly Beck, Norma Stec, holding parts of the first computers.

c. 1972: African-American woman computer operator at the Office of Personnel Management.

1969: Margaret Hamilton with the code she and her staff wrote for the Apollo 11 mission.

Figure 1. Historical depictions of programmers and computer technicians serve to (re) claim such activities. License (in order): a) Raytheon (1969); b), c), and d) Public Domain.

capitalism, competition-as-virtue; and it defaults to extremely polarised gendered archetypes (Erscoi et al., 2023; Hicks, 2017; Little & Winch, 2020).

In this piece, we – under the guidance of our students' feedback, through examination of our own (inter)relationships and lived experiences, and as a function of other academics' views (Abbiss, 2008; Anderson, 2016; Hermans, 2021; Kramarae, 1988; Light, Nicholas, & Bondy, 2015; Mayer, 1981; McCracken et al., 2001; Mitcho, 2016; Sheard et al., 2014; Shrewsbury, 1987; Turkle, 2005; Webb, Allen, & Walker, 2002) – speak to fellow educators who ignore these issues at their students' peril. We aim to probe, dissect, and destroy anti-pedagogical framings that aid few and harm many, especially the racialised and feminised, in the academy and beyond. We posit that such a journey is fraught with danger because of the framing of technology as masculinised and inappropriate for the feminised, people of colour, and any minoritised group. In other words, we face the double bind of being (seen as) both internally and externally hostile to anyone who falls outside the standard white male geek archetype. Freeing ourselves, at least within the confines of the classroom, is a big ask that we lay bare.

Dysfunctional programming

```
// In this Scala snippet, what does the function
// main() print?

@main def main() = println("Hello, World!")
```

The current state of teaching coding minimally ignores the backdrop of sexism and maximally plays into it. How do we include those whom their classmates, other educators, and society at large are excluding? To answer such questions, the wider context of learners needs to be understood.

From a young age, children learn that tinkering, reverse engineering, video games, and other such activities, are seen as prototypically masculine (Lien, 2013). Worse, these activities or hobbies are not recognised as such when the feminised generally take part in them or when these activities occur in less male-coded settings (Scott, 2019), e.g. puzzle video games are not seen as true video games, and HTML is not seen as a true programming language. This facilitates statements, e.g. about women's programming capacity, such as in Dialogue 3 (see below), to be seen as uncontroversial.

Obfuscating women's interests in and contributions to technology echoes through the ages (Erscoi et al., 2023); see Figure 1. Culinary recipes are archetypal algorithms, but cooking is not seen as related to programming (cf. Shore, 1985). Jaccard looms, machines that weave cloth, are the original use case for punchcards – physical pieces of paper that were used to program computers (Harlizius-Klück, 2017). Ada Lovelace invented the first computer program (Aiello, 2016). Women mathematicians and programmers worked with the first digital computer, the ENIAC (Kleiman, 2022). Grace Hopper invented the compiler (Beyer, 2012). Core rope memory was created through knitting copper wires by women for NASA's Apollo missions (Rosner, Shorey, Craft, Remick, 2018). Margaret Hamilton was Director of the Software Engineering Division that, inter alia, took humans to the Moon (Hamilton & Hackler, 2008).

The contradictions rising evermore give us framings such as 'women are good at language and men are good at logic and maths' which fail to notice logic, maths, and programming languages are all languages. All these framings are not only sexist, but discombobulating to our students and false (Kelly, Wang, & Mizunoya, 2022; O'Dea, Lagisz, Jennions, & Nakagawa, 2018; Voyer & Voyer, 2014). Ultimately, these are typical trends within capitalist patriarchy, where women's – and all minoritised people's contributions – are systematically hidden from the historical retelling of humanity's achievements (also known as cryptogyny, the Matilda effect: Connell & Janssen-Lauret, 2022; Evans, 2020; Gage, 1883; Hicks, 2017; Kleiman, 2022; Lee Shetterly, 2016; Pozo & Padilla, 2019; Pozo-Sánchez & Padilla-Carmona, 2021; Rossiter, 1993; Van den Brink & Benschop, 2012). Part of a good teacher's repertoire is this fact, which both drives a more expansive appreciation of their own field and results in a broader and more interesting syllabus.

Abeba Birhane: a cognitive scientist who specialises in inter alia responsible and ethical AI, at the Mozilla Foundation and Trinity College Dublin.

Naomi Wu: a DIY maker (a subculture that creates though hacking, tinkering, and otherwise innovating tech, arts, and crafts) and expert in 3D printing.

Regina Honu: a tech entrepreneur, who founded Soronko Academy, a school for girls to learn coding.

Adele Goldberg: a computer scientist and one of the creators of the programming language Smalltalk.

Figure 2. Displaying the breadth of what one can do with coding skills, or what a person with computational skills looks like, dissolves masculinist assumptions. License (clockwise): a) and b) used with permission, c) used with permission from the Computer History Museum, d) CC BY-SA 4.0, by Chickymaria: https://commons.wikimedia.org/wiki/File:Regina_Honu_01.jpg.

In the Dutch setting, direct relationships are proposed between what specialisation in high school students chose, i.e. between so-called hard sciences or so-called soft sciences, prior to entering our classrooms – and this prior exposure is taken as predictive of their aptitude (Scheerens, Timmermans, & Van der Werf, 2019). The fact that programming languages are so Anglocentric rears its head from childhood, interlocking with class and educational attainment (Hermans, 2024; Swidan & Hermans, 2023). Ironically, this is not taken as a part of our jobs as educators, but as a deficit or an essential characteristic of the student (Abbiss, 2008; O'Dea et al., 2018).

In the Anglosphere setting, these relationships to technology or so-called hard sciences are traced further back to childhood (Lien, 2013; Margolis & Fisher, 2001; O'Mara, 2022; Scott, 2019). Statements from mentors recruiting childhood exposure are said lightly without reflexivity.[2] Adding to the irony, these are the same mentors who do little to no coding on a daily or weekly basis. If any property of a skill is uncontroversial, it is that frequent exercise of said skill is likely indicative of current aptitude.

However, and much more importantly, there is no critical window for learning coding. There is no biological clock that starts ticking, counting down from birth to childhood when exposure to code sets one on a course to being adept at coding for life (cf. Forbes, Aneja, & Guest, 2022). There is nothing stopping anybody at any age from having fun with code or retraining

as a programmer. Basic coding is not radically different from adding a little formal veneer to basic literacy and numeracy all students have already been exposed to and have mastered (see Dialogue 1 below).

Dialogic deprogramming

```
/* What is on the left of the :-  is true if what
is on the right is true in Prolog. What does this
mean? */

grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
```

in computer science it is heard that 'women are stupid' and 'women cannot code as well as men' – and these are taken as facts of the matter, not open to debate or questioning, let alone unpacking as forms of abuse (see Dialogue 3). This continues in our proximal academic environments to this day, as reported by our own students in an artificial intelligence bachelor's degree and by women students in computer science in Margolis and Fisher (2001) and Yates and Plagnol (2022).

On the other hand, in psychology, educators are not only reluctant to teach these highly prized skills, but are also outspoken and defensive about their reluctance (see Dialogues 4–7 below). This reluctance is in contrast with the facts on the ground, where the book *Learning Statistics with R: A Tutorial for Psychology Students and Other Beginners* by Danielle Navarro (2013) was averaging 90 downloads per day in 2014. She also notes:

> I've been pleasantly surprised at just how little difficulty I've had in getting undergraduate psych students to learn R. It's certainly not easy for them, [...] but they do eventually get there. [...] So if the students can handle it, why not teach it?
> (Danielle Navarro, 2013, p. xii)

This makes clear the backdrop of negativity she is reacting to: students are expected to be reluctant or even unable to learn coding. Academics are rarely on record claiming women in psychology cannot (learn to) code (cf. Long, 2018), but this is no reason not to address these claims (Tupas & Tarrayo, 2024). Relatedly, BSc degree programmes, including psychology, can have negative consequences on women's academic careers: 'there is often a smaller percentage of women than men among doctoral graduates even in domains in which they are in the majority at the undergraduate level' (Aelenei, Martinot, Sicard, & Darnon, 2019, p. 4).

Dialogue 1

👩 I am too old to learn to code; others started in childhood.

👩‍💻 Many learn to code, and many other complex skills, later in life. But also, programming basics are not novel concepts: *if-statement*, 'if you are hungry, then you can have a snack'; *while-loop*, 'while there are slices of pizza left, offer them to your guests'; *object-orientation*, 'a dog is a type of mammal, so you can typically expect it to have four legs. An orca is also a mammal, but has modified legs for an aquatic life'. Age is not relevant. Those who use their prior, much younger, exposure to coding are gatekeeping you; they are not warning you of real educational dangers, but using this false excuse to stop you learning. Unlike some facets of cognitive development, *there is no critical window for learning to code*; this is a myth.

What is typified here is a classic framing that takes perhaps useful notions from education like zones of proximal development (Vygotsky & Cole, 1978) or critical windows (Burrill, 1985) and misapplies them in a self- and other-harmful way.[3] The self-harm is the product of the active hostile enculturation that is tacitly and directly promoted by those who want to gatekeep (i.e. keep women out of coding; Hicks, 2017; Yates & Plagnol, 2022). This framing is also found outside undergraduate courses that require coding in their first year, since there the students in many ways have no choice: the programme they are enrolled in, e.g. computer science, requires programming. Such settings are prevalent in psychology when advanced methods courses cause the learners to confront their (perceived) minimal technical or coding skills. The men, while dramatically fewer than the women in psychology, nonetheless end up being the 'code guy' (Johnson, Madill, Koutsopoulou, Brown, & Harris, 2020; White, 2020).

Dialogue 2

👩 I do not know how to code, even though I am in my final year of an artificial intelligence BSc.

👩‍💻 Why do you think this, given you have passed all your programming classes?[4] That is the university's definition of knowing something, which might not mean much, but what metric are you using? Stereotypically polluted perceptions of women coders taken from your classmates, teachers, or society at large? Remember, the coding you do is real coding.

Not believing in our skills as women coders is a typical academic journey (recall Yates and Plagnol, 2022; also see Lehtinen, Lukkarinen, & Haaranen,

2021). Most women seem to think, due to years of denigration, that they are not as good as the men. In other words, 'in the academic contexts in which women are less certain that they belong, they may consequently feel less academic self-efficacy' (Aelenei et al., 2019, p. 6). Even the kind of men who will choose to read this article should reflect on how there have been moments where they have also held biases, made throwaway denigratory comments, or allowed such attitudes or behaviours to pass uncritically.

Dialogue 3


Women are not good at programming.


If you think this, then can you explain how women: invented the compiler (Grace Hopper), wrote the code that took humans to the moon (Margaret Hamilton), and on and on? If these are somehow exceptions to some baseline where most women are not that good at coding, I can grant it if you also grant that most people are not that good. Not everybody is an expert writer or mathematician, but girls excel at these subjects (Kelly et al., 2022; O'Dea et al., 2018; Voyer & Voyer, 2014).

These self- and other-harming comments are a reaction to perhaps their world view shattering. Notably Hicks (2017) and Lee Shetterly (2016) might be useful materials for such students to help set them on a better footing. This might also be an opportune moment to remember the vicious cycle at play here. Sexism drives and is driven by many of these assumptions, that is to say, 'women on GitHub [a social code-sharing website] may be more competent overall, bias against them exists nonetheless' (Terrell et al., 2016, p. 1).

Dialogue 4


Students want or need graphical user interfaces (GUIs), otherwise they do not enjoy the course and/or they cannot learn as well.


Pupils from a young age for example learn complex enough linguistic, mathematical, and artistic skills without the use of GUIs. Besides, we already expect undergraduates to have the ability to navigate complex statistical or conceptual work without multimedia.

This is a myth that is pushed in part by the technology industry, e.g. vendors such as MathWork's MATLAB and IBM's SPSS. While GUIs might seem easy to use, they are not conducive to (and in fact may harm) so-called 'computational thinking' – the skill we are trying to teach (Anderson, 2016; Angeli & Giannakos, 2020; Wing, 2006; cf. Basman, 2017). Learning to use a

GUI is a different skill set and not a necessary core subject in a programming course.

Dialogue 5

👨 We cannot teach them to code because scoping (or any other programming concept) is difficult and time-consuming to learn.

👩‍💻 If the goal is to make every course as easy as possible – where easy means the teacher is purposefully avoiding what they perceive as difficult, but they have this knowledge – then our students will rightfully complain because this is both elitist and anti-pedagogical.

These are frames that are premised on the idea that somehow computational knowledge is more difficult to learn in the abstract and not difficult to learn as a function of the teacher or social contexts (Birhane & Guest, 2021; Gould, 1981; Hampshire, Highfield, Parkin, & Owen, 2012). Which is to say: 'The problem with women in technology isn't the women' (Ford, 2015).

We can even invert the paradigm that difficulty with respect to coding is somehow unique or gendered in essence (Abbiss, 2008). Learning anything of value is a serious but not impossible commitment. To say one cannot teach something because that something (e.g. computational thinking: Brennan & Resnick, 2012; Wing, 2006) is difficult is to say one is not qualified to do so. Furthermore, to say that what one teaches is driven merely by (perceived) difficulty is a serious error. This becomes doubly erroneous if claimed by a member of staff in a psychology department from the perspective of understanding human cognitive capacities (Fine, 2010; Kyndt, Dochy, Struyven, & Cascallar, 2010).

Dialogue 6

👨 Teaching students to code is zero-sum, so that means removing other parts of the course.

👩‍💻 Coding can be taught alongside other things. In psychology this can be during experimental design, since academics use programming language(-derived tools), and during statistics courses (see also Anderson, 2016). These are packaged together as 'Research Methods,' which span the full breadth of a multiyear degree in psychology, meaning there is ample time to teach relevant programming practice as you use in your research.

This zero-sum property may hold. In the United Kingdom, the British Psychological Society (BPS) controls curricula, which means that to be accredited certain criteria must be met. A welcome change is the flexibility in teaching Research Methods. Furthermore, the BPS states statistics taught in R is an

appropriate method of teaching students some of the required research skills (The British Psychological Society, 2017). This means programming can be folded into teaching Research Methods without adding overhead to students who may otherwise not have the curriculum space.

On the other hand, in the Netherlands 'students experience a higher workload than 28 hours per EC' (Faculty Student Council, 2022) and 'the number of students taking longer than 3 years to complete their studies [for a 3 year degree] is relatively large' (QANU, 2020, p. 16). Students need stretches of uninterrupted time to manage their time, so this is undesirable (Kyndt, Berghmans, Dochy, & Bulckens, 2013).

Dialogue 7


You cannot teach them how to code during a stats class because some students will have a 'handicap' if they have not coded before.[5]


Isn't the real disadvantage leaving university without ever learning how academics do their work? If some students already know how to program, which is your contention, the problem is that imbalance. You can address this by having a class that weaves these concepts into their statistical training, or indeed a separate class.

As mentioned above, this is a type of zero-sum framing (Kyndt et al., 2010), which we may need to navigate. Educators like Navarro (2013) have done it, so others can draw inspiration from her materials.

Additionally, we would like to problematise the assumption that the educator in a university setting where the learning goals comprise 'learning how to program' has to do much for students who already code. These students often perpetuate masculinist notions (Margolis & Fisher, 2001; McCracken et al., 2001). As educators, we have a responsibility to protect our students from anti-pedagogical framings. We must use our judgement to decide if a student who already achieves the learning goals can be safely taught to promote our values in class.

In the classroom

```
# What does the following R code print?

a <- 33
b <- 200

if (b > a) {
  print("b is greater than a")
}
```

Large classrooms have their own sets of difficulties. Some students will be learning this skill for the first time, others may have had previous exposure. The role of the teacher is to make sure those students who have had the least exposure learn to code, and to recognise they have learned to code (Dialogue 2).

In contrast, perceived-to-be experienced coders are not the responsibility of the educator to keep entertained (Dialogue 7). Unlike other earlier stages of education, university classes are often not mandatory. A student who feels un(der)stimulated by classes on topics they already know can either not attend and pass the exam based on their previous skills, or request to be given an exemption. An educator's job is to teach those who do not know how to code, and not to keep experienced coders highly stimulated (which perhaps might be the case in prior stages of education; Angeli, 2022; Angeli & Valanides, 2020; De Wit, Hermans, Specht, & Aivaloglou, 2023). Herein, we propose something that seems radical to some of our colleagues, that a teacher's role is to help those who meet the entrance criteria of their course and take them on a pedagogical journey to meet (or indeed surpass) the learning objectives.

Workgroups or practical sessions also present a series of difficulties (Lehtinen et al., 2021; Morrison, Margulieux, & Guzdial, 2015). This is especially true if the teacher is unable to spread their attention over all groups, pairs, or individual students at all times. Teaching assistants can be trained to spot the deployment of statements in Dialogues 1–3 and act appropriately. Lacking these interventions, educational contexts can easily be derailed into significant emotional labour, wherein feminised students are traumatised by framings of their (perceived) inability to code (Lewis et al., 2016; Lind-Guzik, 2023; Terrell et al., 2016; Yates & Plagnol, 2022).

In mentoring

```
# What will this Ruby snippet print?

n = -1
if n > 0
  puts "n is positive"
elsif n == 0
  puts "n is zero"
else
  puts "n is negative"
end
```

In contrast to the larger setting of the classroom, one-on-one mentoring relationships allow for a more direct examination of problematic baggage

our mentees may carry. Such an intimate setting requires even more due care and attention, as things said in such contexts may have the deepest impact. Situations such as those uncovered in Dialogue 2, where mentees disclose deeply held (harmful) beliefs about their skills, arise when the mentor-mentee relationship is one where divulging such self-images is seen as safe.

A healthy mentoring relationship is required for a flourishing mentee (Phillips-Jones, 2003). In such settings – part of academics' pastoral and managerial responsibilities – we should strive to elevate our mentees to heights that they alone may not yet be ready to reach (due to trauma visited upon them by previous experiences; recall Dialogues 1 & 2). Importantly, however, not all our mentees will be feminised – it is unlikely most of them will be, given the current gendered landscape if the context is programming. Relatedly, some may be more likely to express or believe sentiments such as those captured by Dialogue 3, and this is much more likely if we ourselves are not feminised. In other words, men mentors, for example, may have differing opportunities for intervention. We implore our colleagues to take such opportunities for changing perspectives.

In the Dutch setting, PhD candidates may take classes to hone their technical skills. In the United Kingdom setting, PhD students are not employees and are more actively mentored, allowing for personalised pastoral care. In both settings, as supervisors/mentors, we have a responsibility to investigate if their presence is safe for other learners. If our mentee is gendered and/or racialised, we should allow space for them to report to us what tensions or problems may arise in these spaces. Graduates learning to program provides a fertile environment to collaboratively address biased or lacking educational experiences.

How not to go loopy

```
# Can you guess what a for-loop does in the
# following example in Python?

fruits = ["lemon", "banana", "pineapple"]
for fruit in fruits:
  print(fruit)
```

Given the above, what concrete steps may an educator take? We have an obligation to deradicalise our masculinist students, both for their own benefit and for the safety and educational success of their peers (also see Abbiss, 2008; Berry, McKeever, Murphy, & Delany, 2022). Intertwined with

this, we have an obligation to support our vulnerable students through learning concepts and skills that are embedded in a minefield of distractors and punishments. Below are some basic points to avoid or promote in your learning spaces:

Avoid catering to the most competent students other than to give them (if asked) work on diversity, inclusivity, and equity issues within programming. For example, essays on historical programmers or organising events like viewings of *Hidden Figures* (based on Lee Shetterly, 2016). Recruit them to help other students sparingly – or not at all – and ensure they do not recapitulate that certain demographics are inherently more skilled.

Remember there is no one way of teaching other than your own way of imparting knowledge and nurturing the students. If you inherit materials, question them. If you have high student attrition, look at the demographics and ask why. The answer is generally socially unjust forces are at work, but your unique case may need specific interventions; new teaching methods or more women staff might not be enough. Think deeply and take your time.

Avoid assuming you are a good teacher – do not take your students' word on this as final. Be ready to grow. While student evaluations are indispensable, they are not experts on what/how you should teach. Pedagogy comprises many academic fields, and students are not trained in them. It is your responsibility to seek out experts on teaching programming (see Bibliography).

Avoid pretending sexism is absent from your classroom, from daily interactions with other students, from students' educational histories. Neglecting it leaves the door open for masculinist radicalisation to harm the feminised programmers, and for racist, or otherwise socially unjust, notions of who can code (e.g. ableism; Bocconi, Dini, Ferlino, Martinoli, & Ott, 2007; Van der Meulen, Hartendorp, Voorn, & Hermans, 2023).

Avoid deploying individualistic framings such as so-called stereotype threat, impostor syndrome, or implicit associations. Telling students that, all else equal, the problem is within their head constitutes an improper basis on which to build a functional learning environment, and is tantamount to victim-blaming in this context. Sexism and racism are out there in the world and not something women or people of colour are creating in our classrooms to subvert our pedagogy.

Ask students to care about each other and each other's learning experiences. Warn them away from gendered dynamics in which the masculinised students are typically explaining things to the feminised students, but

nonetheless empower them to help, support, and care for each other. Competition-as-virtue and individualism are not useful paradigms in a pedagogical safe space.

Remind mentees of humility because it is important to remember that nobody knows everything about everything. Technology is constantly changing, and so current knowledge becomes outdated faster than people realise. Relatedly, being wrong, e.g. introducing bugs to code accidentally, is part of the learning process.

Promote reflexivity – there is value in looking back at and thinking deeply about both how far learners have come in terms of the direct learning goals and with respect to overcoming sexist, racist, or other framings. Learning how to code, exploring their ability to teach others, if they are mentoring their peers, or assisting you with teaching, as well as surpassing maladaptive social conditioning are all valuable achievements.

On this final note, the above suggestions are meant to inspire educators' reflexivity. They are not meant to be used as a way to be catastrophically self- or other-critical (Okun, 2021). Cultivating healthy learning spaces is a work-in-progress; definitionally in flux.

OO, so what now?

```
# A class is a blueprint for creating objects.
# What is the output in the following Python
# snippet?

class Fruit:
  def __init__(self, name, colour):
    self.name = name
    self.colour = colour

f = Fruit("strawberry", "red")

print(f.colour)
```

Looking forwards, we ask that educators who are not able to carry the whole classroom take a step back and question why: what holds you back? Presenting in class materials such as those presented here (Figures 1 and 2) can go a long way. If you are a woman, feminised, or gender diverse, showing up has impact; from an artificial intelligence undergraduate in her final year:

[Women professors and educators] inspire me to maybe pursue an academic career. I used to think not seeing many women didn't bother me,

but apparently it really helps. I just realised this week we only have had 1 female teacher for all the compulsory courses.

Humans learn and continue to learn, and we can also choose to forget. We can collectively decide to leave behind toxic framings that only certain types of people can learn to code. This constitutes a zeroth step towards reclaiming and rehabilitating programming as skill and as profession – invented by women erased from mainstream history – and it presents unique challenges to both learner as student and learner as educator.

Notes

1. The authors would like to thank Iris van Rooij and Kirstie Whitaker for their feedback on an earlier draft of this manuscript. We would also like to thank the CCS group for their helpful reflections on the contents, and specifically Todd Wareham for useful references. Parts of this paper are based on a blog post by the first author (Guest, 2018).
2. While our focus is not one that lies outside the Global North/West, the first author grew up shielded from many such framings (although with exposure to computers indeed at a young age) in Cyprus.
3. See Angeli and Georgiou (2023), Gilsing et al. (2022), Macrides et al. (2022), and van der Werf et al. (2022) for teaching programming in childhood.
4. Perhaps unexpectedly, this was heard from one of our most academically successful women students in artificial intelligence, while doing a thesis project that involved coding.
5. For example, Wagenmakers (2018).

Bibliography

Abbiss, J. (2008). Rethinking the 'problem' of gender and IT schooling: Discourses in literature. *Gender and Education*, *20*(2), 153–165.

Aelenei, C., Martinot, D., Sicard, A., & Darnon, C. (2019). When an academic culture based on self-enhancement values undermines female students' sense of belonging, self-efficacy, and academic choices. *The Journal of Social Psychology*, *160*(3), 373–389.

Aiello, L.C. (2016). The multifaceted impact of Ada Lovelace in the digital age. *Artificial Intelligence*, *235*, 58–62.

Anderson, N.D. (2016). A call for computational thinking in undergraduate psychology. *Psychology Learning & Teaching*, *15*(3), 226–234.

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction*, *31*, 100329.

Angeli, C., & Georgiou, K. (2023). Investigating the effects of gender and scaffolding in developing preschool children's computational thinking during problem-solving with bee-bots. *Frontiers in Education*, *7*.

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, *105*, 106185.

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, *105*, 105954.

Basman, A. (2017). If what we made were real: Against imperialism and cartesianism in computer science, and for a discipline that creates real artifacts for real communities, following the faculties of real cognition. *PPIG*, 23.

Berry, A., McKeever, S., Murphy, B., & Delany, S.J. (2022). Addressing the 'leaky pipeline': A review and categorisation of actions to recruit and retain women in computing education. *arXiv preprint arXiv:2206.06113*.

Beyer, K. W. (2012). *Grace Hopper and the invention of the information age*. MIT Press.

Birhane, A., & Guest, O. (2021). Towards decolonising computational sciences. *Kvinder, Køn & Forskning*, *1*, 60-73.

Blum, L., & Frieze, C. (2005). The evolving culture of computing: Similarity is the difference. *Frontiers: A Journal of Women Studies*, *26*(1), 110–125.

Bocconi, S., Dini, S., Ferlino, L., Martinoli, C., & Ott, M. (2007). ICT educational tools and visually impaired students: Different answers to different accessibility needs. *Universal Access in Human-Computer Interaction. Applications and Services*, 491–500.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, *1*, 25.

Burrill, C. (1985). The sensitive period hypothesis: A review of literature regarding acquisition of a native-like pronunciation in a second language. *Paper presented at a meeting of the TRI-TESOL conference, Bellevue, WA*.

Busjahn, T., & Schulte, C. (2013). The use of code reading in teaching programming. *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*.

Connell, S.M., & Janssen-Lauret, F. (2022). Lost voices: On counteracting exclusion of women from histories of contemporary philosophy. *British Journal for the History of Philosophy*, *30*(2), 199–210.

De Jonge Akademie. (2021). *A smarter academic year.* Amsterdam. Retrieved from https://www.dejongeakademie.nl/documenten/handlerdownloadfiles.ashx?idnv=2043173

De Wit, S., Hermans, F., Specht, M., & Aivaloglou, E. (2024). Gender, social interactions and interests of characters illustrated in scratch and python programming books for children. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1.*

De Wit, S., Hermans, F., Specht, M., & Aivaloglou, E. (2023). Exploring the effects of the Hedy user interface on the development of cs interest in girls. *10th ACM Celebration of Women in Computing womENcourage: Computing Connecting Everyone.*

Dresden, B.E., Dresden, A.Y., Ridge, R.D., & Yamawaki, N. (2018). No girls allowed: Women in male-dominated majors experience increased gender harassment and bias. *Psychological reports*, *121*(3), 459–474.

Erscoi, L., Kleinherenbrink, A.V., & Guest, O. (2023). Pygmalion displacement: When humanising ai dehumanises women. *SocArXiv. February*, *11*.

Essanhaji, Z. (2023). The (im)possibility of complaint: On efforts of inverting and (en)countering the university. *Gender and Education*, 1–16.

Evans, C.L. (2020). *Broad band: The untold story of the women who made the internet.* New York: Penguin.

Fine, C. (2010). *Delusions of gender: How our minds, society, and neurosexism create difference.* New York: WW Norton & Company.

Forbes, S.H., Aneja, P., & Guest, O. (2022). The myth of normative development. *Infant and Child Development, 33*(1), e2393.

Ford, P. (2015). What is code? if you don't know, you need to read this. Retrieved from https://www.bloomberg. com/graphics/2015-paul-ford-what-is-code/

Gage, M.J. (1883). Woman as an inventor. *The North American Review*, *136*(318), 478–489.

Gilsing, M., Pelay, J., & Hermans, F. (2022). Design, implementation and evaluation of the Hedy programming language. *Journal of Computer Languages*, *73*, 101158.

Goffman, E. (1977). The arrangement between the sexes. *Theory and society*, *4*(3), 301–331.

Gould, S. J. (1981). *The mismeasure of man.* New York: Norton.

Guest, O. (2018). Why women in psychology can't program. Retrieved from http://neuroplausible.com/ programming

Hamilton, M.H., & Hackler, W.R. (2008). Universal systems language: Lessons learned from apollo. *Computer*, *41*(12), 34–43.

Hampshire, A., Highfield, R.R., Parkin, B.L., & Owen, A.M. (2012). Fractionating human intelligence. *Neuron*, *76*(6), 1225–1237.

Harlizius-Klück, E. (2017). Weaving as binary art and the algebra of patterns. *Textile*, *15*(2), 176– 197.

Hermans, F. (2021). *The programmer's brain: What every programmer needs to know about cognition*. Shelter Island, NY: Manning Publications.

Hermans, F. (2024). [onward23] creating a learnable and inclusive programming language. Retrieved from https://www.youtube.com/watch?v=VzXiup5Gm7Y

Hermans, F., & Aldewereld, M. (2017). Programming is writing is programming. *Companion Proceedings of the 1st International Conference on the Art, Science, and Engineering of Programming*, 1–8.

Hicks, M. (2017). *Programmed inequality: How Britain discarded women technologists and lost its edge in computing*. Cambridge, MA: MIT press.

Johnson, J., Madill, A., Koutsopoulou, G.Z., Brown, C., & Harris, R. (2020). Tackling gender imbalance in psychology. *Psychologist*, *33*, 5–6.

Kelly, P., Wang, Y., & Mizunoya, S. (2022). How do the educational experiences of girls and boys differ? Retrieved from https://data.unicef.org/data-for-action/how-do-educational-experiences-ofgirls-and-boys-differ/

Kleiman, K. (2022). *Proving ground: The untold story of the six women who programmed the world's first modern computer*. Hurst Publishers.

Kramarae, C. (1988). *Technology and women's voices: Keeping in touch*. Routledge.

Kyndt, E., Berghmans, I., Dochy, F., & Bulckens, L. (2013). 'Time is not enough': Workload in higher education: A student perspective. *Higher Education Research & Development*, *33*(4), 684–698.

Kyndt, E., Dochy, F., Struyven, K., & Cascallar, E. (2010). The perception of workload and task complexity and its influence on students' approaches to learning: A study in higher education. *European Journal of Psychology of Education*, *26*(3), 393–415.

Lee Shetterly, M. (2016). *Hidden figures: The American dream and the untold story of the Black women mathematicians who helped win the space race*. New York: William Morrow.

Lehtinen, T., Lukkarinen, A., & Haaranen, L. (2021). Students struggle to explain their own program code. *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*.

Lewis, C.M., Anderson, R.E., & Yasuhara, K. (2016). 'I don't code all day': Fitting in computer science when the stereotypes don't fit. *Proceedings of the 2016 ACM conference on international computing education research*, 23–32.

Lien, T. (2013). No girls allowed. Retrieved from https://www.polygon.com/features/2013/12/2/5143856/nogirls-allowed

Light, T. P., Nicholas, J., & Bondy, R. (2015). *Feminist pedagogy in higher education: Critical theory and practice*. Waterloo, Ontario: Wilfrid Laurier University Press.

Lind-Guzik, A. (2023). You should give a sh*t about: Gender apartheid. Retrieved from https : / / conversationalist.org/2023/05/18/end-gender-apartheid-today-open-letter-interviewgissou-nia-human-rights-lawyer/

Little, B., & Winch, A. (2020). Patriarchy in the digital conjuncture: An analysis of Google's James Damore. *New Formations*, *102*(102), 44–63.

Long, K. (2018). Why don't women code? a UW lecturer's answer draws heat. Retrieved from https://www.seattletimes.com/seattle-news/education/why-dont-women-code-a-uw-lecturersanswer-draws-heat/

Macrides, E., Miliou, O., & Angeli, C. (2022). Programming in early childhood education: A systematic review. *International Journal of Child-Computer Interaction*, *32*, 100396.

Margolis, J., & Fisher, A. (2001). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: MIT Press.

Mayer, R.E. (1981). The psychology of how novices learn computer programming. *ACM Computing Surveys*, *13*(1), 121–141.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.-D., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, *33*(4), 125–180.

McKinley, K. S. (2018). What happens to us does not happen to most of you. Retrieved from https://www.sigarch.org/what-happens-to-us-does-not-happen-to-most-of-you/

Mitcho, S.R. (2016). Feminist pedagogy. *Encyclopedia of Educational Philosophy and Theory*, 1–5.

Morrison, B.B., Margulieux, L. E., & Guzdial, M. (2015). Subgoals, context, and worked examples in learning computing problem solving. *Proceedings of the eleventh annual International Conference on International Computing Education Research*.

Navarro, D. (2013). *Learning statistics with R*. Lulu.com. Retrieved from http://compcogscisydney.org/learning-statistics-with-r

O'Dea, R.E., Lagisz, M., Jennions, M.D., & Nakagawa, S. (2018). Gender differences in individual variation in academic grades fail to fit expected patterns for stem. *Nature communications*, *9*(1), 3777.

Okun, T. (2021). Retrieved from https://www.whitesupremacyculture.info/

O'Mara, M. (2022). Why can't tech fix its gender problem? Retrieved from https://www.technologyreview.com/ 2022/08/11/1056917/tech-fix-gender-problem/

Phillips-Jones, L. (2003). *Skills for successful mentoring: Competencies of outstanding mentors and mentees*. CCC/The Mentoring Group.

Pozo, B., & Padilla, C. (2019). Criptogıń ia: Una paraula nova per a un fenomen antic. *eldiario. es*, *5*, 2019.

Pozo-Sánchez, B., & Padilla-Carmona, C. (2021). Criptoginia: Una palabra nueva, un concepto para investigar. *Quaderns de Filologia-Estudis Lingüístics*, *26*, 175–192.

QANU. (2020). Report on the bachelor's and the master's programmes artificial intelligence of Radboud University. Retrieved from https://publicaties.nvao.net/ACCR_009468_21PM-56945_ Artificial_Intelligence_Rapport_2020.pdf

Raytheon. (1969). Apollo 11 Press Kit Raytheon. Retrieved from https://www.apollopresskits.com/apollopresskit-directory

Rosner, D.K., Shorey, S., Craft, B.R., & Remick, H. (2018). Making core memory: Design inquiry into gendered legacies of engineering and craftwork. *Proceedings of the 2018 CHI conference on human factors in computing systems*, 1–13.

Rossiter, M.W. (1993). The Matthew Matilda effect in science. *Social studies of science*, *23*(2), 325–341.

Salter, A., & Blodgett, B. (2017). *Toxic geek masculinity in media: Sexism, trolling, and identity policing*. Cham: Palgrave Macmillan.

Scheerens, J., Timmermans, A., & Van der Werf, G. (2019). Socioeconomic inequality and student outcomes in the Netherlands. In Volante, L., Schnepfe, S., Jerrim, J., & Klinger, D. (Eds.), *Socioeconomic inequality and student outcomes: Cross-national trends, policies, and practices* (pp. 111–132). Singapore: Springer.

Scott, S. (2019). *Fake geek girls*. New York: New York University Press.

Sheard, J., Simon, J., Dermoudy, J., D'Souza, D., Hu, M., & Parsons, D. (2014). Benchmarking a set of exam questions for introductory programming. *Proceedings of the Sixteenth Australasian Computing Education Conference* (*ACE2014*), *Auckland, New Zealand*.

Shore, J. (1985). *The sachertorte algorithm and other antidotes to computer anxiety* (Vol.17). New York: ACM New York.

Shrewsbury, C.M. (1987). What is feminist pedagogy? *Women's Studies Quarterly*, *15*(3/4), 6–14.

Swidan, A., & Hermans, F. (2023). A framework for the localization of programming languages. *Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E*.

Terrell, J., Kofink, A., Middleton, J., Rainear, C., Murphy-Hill, E., Parnin, C., & Stallings, J. (2016). Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science*.

The British Psychological Society. (2017). *Supplementary guidance for research and research methods on Society accredited undergraduate and conversion programmes.* Retrieved from https://cms.bps.org.uk/sites/default/files/202207/Research%20Methods%20-%20Undergraduate%20Programmes%20WEB.pdf

Tupas, R., & Tarrayo, V.N. (2024). The violence of literature review and the imperative to ask new questions. *Applied Linguistics Review*, (0).

Turkle, S. (2005). *The second self.* Cambridge, MA: MIT Press.

Van den Brink, M., & Benschop, Y. (2012). Gender practices in the construction of academic excellence: Sheep with five legs. *Organization*, *19*(4), 507–524.

Van der Meulen, A., Hartendorp, M., Voorn, W., & Hermans, F. (2023). Observing the computational concept of abstraction in blind and low vision learners using the bee-bot and blue-bot. *Computer Science Education*, 1–23.

Van der Werf, V., Aivaloglou, E., Hermans, F., & Specht, M. (2022). (how) should variables and their naming be taught in novice programming education? *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2*, 53–54.

Voyer, D., & Voyer, S.D. (2014). Gender differences in scholastic achievement: A meta-analysis. *Psychological bulletin*, *140*(4), 1174.

Vygotsky, L. S., & Cole, M. (1978). *Mind in society: Development of higher psychological processes*. Cambridge, MA: Harvard University Press.

Wagenmakers, E.-J. (2018). Agreed, but I am not sure that combining these in a single course is wise, and I worry about students without coding experience feeling that they start with a handicap (because they do). In their first stats course, I like students to grasp the concepts, not "tapply". Retrieved from https://twitter.com/EJWagenmakers/status/1066680953534328832

Webb, L.M., Allen, M.W., & Walker, K.L. (2002). Feminist pedagogy: Identifying basic principles. *Academic Exchange*, *6*(1), 67–72.

White, S.K. (2020). Women in tech statistics: The hard truths of an uphill battle. *CIO online*.

Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Yates, J., & Plagnol, A.C. (2022). Female computer science students: A qualitative exploration of women's experiences studying computer science at university in the UK. *Education and Information Technologies*, *27*(3), 3079–3105.

About the authors

Olivia Guest is an Assistant Professor of Computational Cognitive Science at Donders Institute for Brain, Cognition and Behaviour and the School of Artificial Intelligence, at Radboud University, Netherlands. Her research interests comprise (meta)theoretical, critical, and radical perspectives on the neuro-, computational, and cognitive sciences broadly construed. She emigrated from Cyprus to the UK in 2006 to pursue an undergraduate degree in Computer Science (2009; University of York, UK), a masters in Cognitive and Decision Sciences (2010; University College London, UK), and a PhD in Psychological Sciences (2014; Birkbeck, UK). She has worked

in labs at the University of Oxford, University College London, and as an independent scientist at a research centre in Cyprus. In 2020, she moved to the Netherlands where she still lives and works. Additionally, Olivia is an Associate Editor-in-Chief for ReScience C and for the Journal of Open Source Software.

Samuel Forbes is an Assistant Professor in Psychology at Durham University in the United Kingdom. He completed his DPhil in Experimental Psychology at the University of Oxford in 2018, where he researched how infants learn colour words and what that means for their colour perception. He worked the University of East Anglia as a postdoc, working on early visual working memory with Professor John Spencer. His research at Durham focuses on the interplay between early cognition and word learning in infants, looking particularly at the links between cognition, environmental factors and early word learning, using a mixed-methods approach. He also carries out work in metascience and methods development, developing pipelines and software for researchers to use in their own research, and runs workshops in coding for psychologists.