Effect of Gender Stereotypes in Programming Assessments on Student Performance

Janet Jiang
University of Washington
Seattle, WA, USA
jcjiang@uw.edu

Amy J. Ko
University of Washington
Seattle, WA, USA
ajko@uw.edu

Mo Zhang
Educational Testing Service
Princeton, NJ, USA
mzhang@ets.org

Kefan Yu
University of Washington
Seattle, WA, USA
ky285@uw.edu

Min Li
University of Washington
Seattle, WA, USA
minli@uw.edu

Abstract

Instructors often contextualize programming problems to increase student motivation and help students relate to the problems. While prior research has pointed out many trade-offs to contextualization, it has not considered whether contextualization can introduce stereotype threat into the classroom. Our study examines how referencing gender stereotypes in programming problems impacts students' performance. We recruited N = 97 students to complete programming problems that referenced gender stereotypes, measuring the number of test cases they passed, how long it took them to read the problems, and how much time they spent working on the problems. We found that neither reinforcing nor subverting gender stereotypes in the context of programming problems introduced gender disparities in performance. However, we noted that a pre-existing gender disparity in the number of test cases passed disappeared on problems that *subverted* gender stereotypes. We also observed a large effect on the time participants spent reading those problems. Overall, our findings suggest that referencing gender stereotypes in programming problems can impact some aspects of performance, but may not result in significant differences overall.

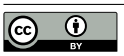
CCS Concepts

• **Social and professional topics → Computing education**.

Keywords

Contextualization, Programming Assessments, Stereotype Threat

ACM Reference Format:
Janet Jiang, Amy J. Ko, Mo Zhang, Kefan Yu, and Min Li. 2027. Effect of Gender Stereotypes in Programming Assessments on Student Performance. In *Proceedings of the 58th ACM Technical Symposium on Computer Science Education V.1 (SIGCSE TS 2027), February 17–20, 2027, Sacramento, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3857222.3860128



This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGCSE TS 2027, Sacramento, CA, USA
© 2027 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3075-7/2027/02
https://doi.org/10.1145/3857222.3860128

1 Introduction

Learning to program is difficult and requires practice [6]. Assessments contribute to this learning by providing students with feedback and helping instructors identify when students need support. Designing good assessments is thus central to teaching computing.

Instructors often try to improve their assessments by adding context to programming problems. For example, a problem where students must sum elements of a list may be accompanied by a narrative where a barista calculates the total tips they received. Contextualization is beneficial because it can motivate students [2, 6, 15] and help students relate to the problems [9, 11, 30]. However, it can also add to students' cognitive load [7, 27] because they must process extra information and determine which parts of the context are relevant. This does not mean contextualization is bad. Specifically, contextualizing programming problems in assignments and assessments allows students to practice and demonstrate their ability to apply computing concepts in social contexts. This cognitive load is necessary for learning, and is therefore germane load [27].

Even so, there may be other unintended consequences to contextualizing programming problems, such as introducing stereotype threat into the classroom. Stereotype threat occurs when a person is "concerned about being judged or treated negatively on the basis of a stereotype" [24]. It can impede students' performance by distracting them from demonstrating their knowledge [24].

Prior research has found that indirectly reminding students about performance stereotypes before an assessment can trigger stereotype threat [12, 13, 25, 26]. It is therefore likely that explicitly referencing, and therefore *reinforcing*, negative stereotypes in the context of programming problems may also trigger stereotype threat. Women, Black, and Hispanic/Latinx students may feel they are at risk of conforming to negative stereotypes about their identity, while Men, White, and Asian students may feel additional pressure to succeed and struggle to focus [12, 13]. In contrast, it is possible that *subverting* negative stereotypes in the context of programming problems might improve performance. Women, Black, and Hispanic/Latinx students may find the context encouraging and develop more confidence in their skills, while Men, White, and Asian students may feel less pressure to succeed.

However, we do not know whether referencing stereotypes in programming problems can trigger stereotype threat and therefore influence students' performance. Here we investigate this possibility for gender stereotypes in particular. We recruited students

who recently completed an introductory programming course to solve programming problems whose context either reinforced or subverted gender stereotypes. We used students' performance on the problems to investigate the following research question (**RQ**): Does referencing gender stereotypes in programming problems introduce gender disparities in performance?

2 Related Work

2.1 Contextualization and Performance

Prior research in psychology [9], math [22], and computer science (CS) has yielded mixed results on whether the contextualization[1] of problems helps or undermines students' performance. In this section, we focus on papers situated in CS contexts. This is because CS has a higher intrinsic cognitive load than other domains [8, 20], so their findings likely cannot be directly extrapolated to CS [2, 14].

A 2016 ITiCSE working group [2] demonstrated that contextualization did not impact novice programmers' performance on a task similar to Soloway's Rainfall Problem [23]. Lovellette *et al.* [16] found no differences in either the structure of students' code or its output correctness on the contextualized and not contextualized versions of Soloway's Rainfall Problem. A more recent study by Lovellette *et al.* [15] concluded that context (and its authenticity) does not influence novices' performance.

Leinonen *et al.* [14] studied whether the contextualization of math-oriented programming tasks impacted novice programmers' performance. They found that students who were assigned a non-context variant of a task passed less test cases and made more algebraic mistakes, but spent less time coding. This suggests that contextualization can both help and harm students and that future research should aim to identify which aspects and kinds of context can affect students' performance. Craig *et al.* [6] evaluated students' performance on contextualized problems from domains introduced earlier in the course. They found that students who completed contextualized problems did not outperform the others.

Our work adds to the literature by examining whether and how a specific kind of context, i.e. context that reinforces or subverts gender stereotypes, impacts students' performance.

2.2 Stereotypes in Computer Science

Stereotypes are "beliefs about the characteristics, attributes, and behaviors of members of certain groups" [10]. Stereotypes are prevalent in CS. Specifically, computer scientists are often thought of as singularly focused, highly intelligent men who lack interpersonal skills and are interested in video games and science fiction [4, 18]. These stereotypes can negatively impact students' performance (see Section 2.3). Furthermore, they can undermine students' interest in CS and deter their participation in the field [3, 4, 18].

2.3 Stereotype Threat

Stereotype threat threatens students' self-integrity and imposes additional pressure on them to succeed [24]. Although this can serve as motivation and improve performance on unchallenging tasks, it impedes performance on challenging tasks. Specifically, it can prevent students from focusing on a task because they need to redirect some of their working memory towards suppressing negative emotions and thoughts that arise [24]. It may also cause students to pay conscious attention to well-learned behaviors that are otherwise conducted automatically [24]. These additional efforts can be considered extraneous cognitive load, which results from less-than-optimal instruction and not the intrinsic difficulty of the material [27]. Lastly, stereotype threat may cause students to lower their self-expectations and place less effort into preparing for a task so that they have an excuse for what they believe is impending failure. As a result, high-achieving individuals who care more about their performance are most impacted by negative stereotypes [24].

The extent to which stereotype threat affects an individual's performance depends on the relevance of the stereotype to the assessment, how strongly the individual identifies with the domain of the assessment, and the difficulty of the assessment [21]. Prior works indicate that the effects of stereotype threat are pronounced on harder assessments [12, 21, 25]. They also suggest that moderately explicit cues tend to result in the largest stereotype threat [21]. It is thus possible that referencing gender stereotypes, a moderately explicit cue, may trigger stereotype threat on difficult assessments.

Much of the literature on stereotype threat focuses on threats targeting students' race and gender [3, 5, 12, 13, 18, 21, 29]. However, some have shown that stereotype threat can also target other identities, such as whether students attended high schools that are well represented at their university [1]. Below, we highlight the few works situated in computing contexts.

Kumar [12] evaluated the impact of stereotype threat in computing by asking students for their demographic information either before or after an online assessment. He found that asking for students' sex and race before the assessment resulted in decreased scores for female and non-Caucasian students *and* male and Caucasian students who had not mastered the course material. In a follow-up study [13], Kumar found that all students scored lower and spent more time on an assessment when they were asked to report their sex and race prior to its start. This suggests that stereotype threat affects both positively- and negatively-stereotyped students.

To our knowledge, no studies have investigated whether *assessment content* can trigger stereotype threat and therefore impact students' performance. Our work begins to fill this gap in the computing education literature. Like Kumar [12, 13], we use students' performance on items as indicators of stereotype threat.

3 Methods

We had students solve Python programming problems ("items") that reinforced or subverted gender stereotypes. The items used content manipulations that were ecologically valid with respect to CS item design norms.

3.1 Positionality

The first author is a PhD student with experience as a teaching assistant for programming courses. She wrote the initial drafts of the items. As an undergraduate student at a U.S. university at the time, she knew what stereotypes may be familiar to the participants and had seen many contextualized programming problems. The items were edited and approved by the other authors, who had expertise in item design and research on educational assessments.

[1]Here, *contextualization* means adding a real-world narrative that motivates a problem.

3.2 Item Design Process

In designing the items, we first compiled a list of concepts we could assess by looking at sample CS1 syllabi. Then, we came up with a list of gender stereotypes that our population, undergraduate students at North American colleges and universities, were likely familiar with. The list had both computing and societal stereotypes because both can generate negative emotions that impede performance.

We then wrote items that reinforced gender stereotypes ("Dominant" items). Around half of the items reinforced the stereotype that men are more interested in and better at computing. The remaining items referenced traditional gender roles and notions of femininity and masculinity. We created items that subverted gender stereotypes ("Disruptive" items) by reversing the direction of the stereotypes in the Dominant items. Each pair of Dominant and Disruptive items could be solved with identical code, and their lengths did not differ by more than one sentence. We provide descriptions of the item pairs in Table 1 and an example item pair below.

> *"A student who was supposed to represent their school for a robotics competition quit the team, and the coaches are issuing an exam to determine who should replace [her/him]. All of the current students on the team are [male ('M')/female ('F')], so the coaches are considering selecting the highest scoring [non-male/non-female] student as the last team member. Write a function that returns the name of the highest scoring [non-male/non-female] student. If there is a tie, select the student whose name comes first alphabetically... [*description of parameters*]."*

The items collectively assessed the following concepts: branching, loops, lambda, and integer, string, list, set, and dictionary manipulation. Most items combined multiple concepts to increase their difficulty. This was because 1) stereotype threat effects are more noticeable on difficult assessments [12, 21, 25], and 2) we wanted to avoid ceiling effects that masked performance differences. For each item, we wrote 7 test cases that empirically defined its correctness.

We piloted the items with 5 undergraduate students at U.S. R1 universities. They gave feedback on how long it took them to solve the items, if the references to the stereotypes sounded forced, and if any language was confusing. We revised the items accordingly, to ensure that the items were reasonably difficult and phrased clearly.

3.3 Study Design

Our study was reviewed and determined exempt by our university's ethics board. The study was conducted in two stages. In the 1st stage, all participants completed 9 "Neutral" items that did not include any narrative (and theoretically would not pose stereotype threat). We used participants' performance on the Neutral items to route them to one of two difficulty levels ("Easy" or "Hard") in the 2nd stage. The routing was necessary because the items needed to be sufficiently difficult in an absolute sense, but also on a per-learner basis. If the items were too easy for participants, we would not be able to observe any stereotype threat effects because they would perform well despite redirecting their working memory [24]. If the items exceeded participants' abilities, we would not be able to distinguish participants who experienced stereotype threat from those who did not. In the 2nd stage, participants completed one of four problem sets (PS1 - PS4). Each problem set contained 9 contextualized items that tested different subsets of skills. This let us test participants' knowledge broadly and not "double-penalize" those struggling with specific skills. To minimize item order effects, we shuffled the order of the items for each participant. We required participants to submit their code for an item before starting another.

The items in PS1 and PS2 were easier than the items in PS3 and PS4 in that they did not assess as many skills and were slightly shorter in length. The items in PS1 and PS3 reinforced gender stereotypes, while the items in PS2 and PS4 subverted gender stereotypes.

We routed participants to a difficulty level ("Easy" or "Hard") based on a combination of item correctness, calculated using the total number of test cases they passed in the 1st stage, and a survey question that participants completed after finishing the 1st stage. Specifically, we asked participants to indicate the extent to which they agreed with the statement that "*The items in the [1st stage] were difficult for [them] to complete.*" This data was collected using a 4-point Likert scale, ranging from "Strongly Disagree" to "Strongly Agree."[2] We used item correctness as the primary routing condition. If a participant passed every test case for all items, they were randomly assigned a "Hard" problem set. If a participant passed an average of <= 2 test cases per item, they were randomly assigned an "Easy" problem set. We routed other participants based on their response to the survey question. If participants agreed or strongly agreed, they were randomly assigned an "Easy" problem set. Otherwise, they were randomly assigned a "Hard" problem set.

3.4 Participant Recruitment

To recruit, we sent an email to computing faculty in higher education and asked faculty who recently taught introductory Python programming courses to share an email with their students. In the email, students were given a link to a pre-survey where they first answered questions about their prior coursework and English fluency. We included these questions to verify that the students had the knowledge to solve the items and could understand the item contexts enough to be impacted by the manipulations. Students who were fluent in English and had taken exactly one introductory Python programming course in the previous year were eligible to participate in our study. We recruited students who took exactly one programming course to ensure that they had sufficient knowledge to solve the items, but not too much knowledge where they would find the items easy and not be susceptible to stereotype threat.

The rest of the pre-survey informed students of the study components, participation guidelines, and compensation. Additionally, it asked students to pledge that they would write all their code in the provided code editor, not use generative AI to solve the items, and "meaningfully attempt" each item by spending at least 5 minutes working on it, unless they passed all the test cases sooner. We did not ask about demographics in the pre-survey to avoid triggering stereotype threat [12, 13] before students began the study.

We note that our goal in recruitment was not to find a representative sample of computing students. Rather, we sought to recruit students who wanted to gain more practice with Python because they are likely high-achieving and are thus more likely to experience stereotype threat [24]. Also, we note that we did not conduct this study in a course because we wanted to reach a diverse group of students from institutions with varying computing cultures.

[2]We excluded the "Neutral" option as it did not provide useful information for routing.

Table 1: Brief descriptions of the item pairs that referenced gender stereotypes. Item pairs labeled with an asterisk (*) referenced computing stereotypes; the rest referenced societal stereotypes.

	Concepts assessed	*Objective (& relevant context)*	*Diff. within item pairs is the gender of the...*
A*	loops, branching	Determine whether a person has a collectible item that can be traded to another in exchange for help assembling a PC	People seeking and giving help
B	loops, branching, string manipulation	Calculate the cost of a manicure	People seeking and providing the manicure
C*	integer manipulation, branching	Calculate whether a student is predicted to pass an introductory CS class that they are currently failing	Student failing the class
D	integer manipulation	Calculate amount a person's spouse will spend getting gas for them	Person
E	list manipulation, branching, loops	Tally up votes to determine if a family will be attending an event	Head of the household
F*	dictionary/list manipulation, branching, loops	Identify what airline has the cheapest flight for a trip that will be paid for by a relative who is a software engineer	Relative
G*	dictionary/list manipulation, branching, loops	Identify who a student should hire as a tutor	Student needing a tutor
H*	nested branching, integer/list manipulation, loops	See example quote.	See example quote.
I	integer/list manipulation, branching, loops	Calculate how much money a sports fan would win/lose from a bet	Sports fan
J	lambda, dictionary/integer/list/set/string manipulation, nested loops, nested branching	Calculate how many places one must visit to buy everything on a grocery list	Person going shopping
K*	lambda, dictionary/list manipulation, branching, nested loops	Identify what foods can be present at a celebration dinner for a friend who just got hired as a software engineer	Friend
L*	lambda, list/integer/dictionary manipulation, branching, loops	Identify pens that are "lucky" and can be used for an exam that determines whether a failing student passes a CS class	Failing student
M	lambda, dictionary/list/string manipulation, nested loops, nested branching	Identify the smallest set of ingredients that a person can buy so that they can make any one of their grandparents' recipes	Grandparent

3.5 Data Collection

Participants completed the study on a platform that automated study participation and collected participants' keystroke log data. It recorded the participants' interactions with the platform interface, including every character the participants typed or deleted and when participants opened an item, indicated that they were ready to start coding, switched tabs, copy/pasted code, and tested or submitted their code. These logs included time stamps. Additionally, when participants tested or submitted their code, the platform recorded the test cases they passed and failed. This data let us track the correctness of participants' code *and* estimate how long it took participants to read each item prompt and write each solution.

After participants finished the pre-survey, they completed (in order): a tutorial of the platform, the Neutral items, a 1-question survey,[3] their assigned problem set, and a post-survey. In the post-survey, we collected participants' gender using free response text boxes. Additionally, we asked them about their "*perceptions of how the problems were phrased, described, and contextualized.*"

504 students filled out the pre-survey, and 450 of them were eligible to participate. 192 students completed the study.

3.6 Data Cleaning

We only analyzed data from participants who meaningfully attempted every item and did not paste external code into the code editor while solving the items, as we could not capture their entire problem-solving process otherwise. A total of 97 participants fit this description, see demographics in Table 2.

Table 2: Demographics of participants who completed the study and followed all guidelines. The gender majority includes men. The gender minority includes women, nonbinary, transgender, and gender-fluid participants.

	N	Gender Majority	Gender Minority
All	97	40	57
PS1 (Easy, Dominant)	24	9	15
PS2 (Easy, Disruptive)	28	7	21
PS3 (Hard, Dominant)	20	13	7
PS4 (Hard, Disruptive)	25	11	14

We also excluded participants' code for an item if the time between when they first opened the item and their final keystroke was greater than an hour. This was because some participants took long breaks while working on select items, as indicated by large time gaps between keystrokes, and much of this break was likely not spent thinking about the item. Lastly, our analyses only considered participants' data from items that mentioned a gender stereotype.[4] We did not include a gender stereotype in every item to avoid the items becoming too similar to one another and to make the problem sets more authentic, since most instructors will not design an assessment where every item references a stereotype.

[3]Which was used to determine their assigned problem set, see Section 3.3.

[4]There were 6 items from PS1/PS2 and 7 items from PS3/PS4 that fit this criteria.

Table 3: Interpretation of CLES values [28].

If the CLES is within...	The effect is...
(0.44, 0.56)	Negligible
(0.36, 0.44] or [0.56, 0.64)	Small
(0.29, 0.36] or [0.64, 0.71)	Medium
[0.00, 0.29] or [0.71, 1.0]	Large

We divided participants into two groups using their responses to the post-survey. Participants who identified as a man were classified as belonging to the *gender majority*. All other participants were classified as belonging to the *gender minority*. This was necessary as less than 10% of the participants identified as non-binary, transgender, or gender-fluid.

3.7 Data Analysis

We measured participants' performance in three ways. First, we analyzed the number of test cases a participant passed for each item (*Tests Passed*). We used the score from participants' final submission because it is most likely to reflect their full knowledge. Second, we analyzed the time a participant spent reading the prompt before starting to code (*Time Reading*). We calculated this as the time (in minutes) between a participant first opening the item and their first keystroke. This metric indicates how difficult it was for participants to comprehend the context. Third, we analyzed the total time (in minutes) between a participant's first and last keystrokes (*Between Keystrokes*). This metric represents participants' time on task and can be considered an indicator of their coding effort. Collectively, these metrics serve as proxies for (extraneous) cognitive load faced by participants who experience stereotype threat.

We used the Mann-Whitney U (MWU) Test [17] in our analysis as our data was not normally distributed. Since we ran multiple tests on each dataset, we applied a Bonferroni correction to reduce the risk of false positives. We performed the correction by dividing the threshold (α = 0.05) by the number of tests performed on each dataset; we use the corrected α (0.0167) to determine the statistical significance of our results. We report the common language effect size (CLES) [19] for all MWU test results. It can be interpreted using guidelines set forth by Vargha and Delaney [28], shown in Table 3. We report the *CLES-Maj*, i.e. the proportion of all possible cross-group pairs of participants in which the participant from the gender majority has the larger metric value. The *CLES-Min* is 1 - *CLES-Maj*. For *Time Reading* and *Between Keystrokes*, *CLES-Maj* values greater than 0.5 are in favor of minority participants.

4 Results

Before proceeding to data analysis, we examined the distribution of the three metrics across *all* participants for each of PS1-PS4. This was to confirm that all three metrics were capable of demonstrating difference. The minimum, median, and maximum (*MMM*) metric values for each subgroup of participants are reported in Tables 4 and 5, rounded to the nearest minute. We found slight ceiling effects on *Tests Passed* on PS1 and PS2. This suggests that PS1 and PS2 may have been too easy, and that we may not be able to see the effects of stereotype threat through *Tests Passed*. There were no floor or ceiling effects for *Time Reading* or *Between Keystrokes*.

We also examined participants' responses to the post-survey question on their perceptions of the items. This was to confirm that other aspects of the items did not significantly hinder performance and to gain insight into participants' impressions of the context.

Overall, participants ($N = 56$) found the items to be phrased well and accessible for participants whose first language was not English. Participants' impressions of the items were varied, with 32 participants finding the context to be helpful, and 10 participants finding the context to be distracting or confusing.[5] Their descriptions of their interactions with the context were consistent with the literature on the tradeoffs of contextualization [2, 6, 7, 9, 11, 15, 27, 30].

Not many participants addressed the nature of the context in their responses; 26 participants mentioned (and, mostly, appreciated) that the context incorporated real-world scenarios, and two of them pointed out that the context referenced gender stereotypes. This does *not* imply that the stereotypes were not salient to other participants — we did not mention stereotypes in the question stem to avoid influencing participants' perceptions with leading questions. It is likely that more participants recognized the stereotypes in the items, but did not think to mention them in their response.

4.1 RQ: Stereotyped context did not introduce gender disparities in performance

To answer our RQ, we compared the performance of participants from the gender majority to that of participants from the gender minority. We first examined participants' performance on the Neutral items, to identify pre-existing differences between groups. Then, we compared participants' performance *within* each of PS1 - PS4, averaging each metric for each participant across the items.

When we compared participants' performance on the Neutral items, we found that minority participants who were later assigned PS2 passed significantly fewer test cases on average than majority participants also assigned PS2 ($p = 0.012$). This suggests that gender minority participants from PS2 may have started with less prior knowledge. There were no significant differences in the three metrics for majority and minority participants from PS1, PS3, and PS4 (see Table 4), although there was a large effect of 0.72 in favor of majority participants from PS1 on *Tests Passed*.

When we compared majority and minority participants' performance on their assigned problem set, we found no significant differences in any of the metrics (see Table 5). Notably, this means that majority and minority participants passed a comparable number of test cases on PS2, despite majority participants passing significantly more test cases on the Neutral items. However, we did observe a large effect of 0.73 on *Time Reading* for PS2.

Collectively, these results suggest that neither reinforcing nor subverting gender stereotypes in programming problems introduced gender disparities in performance. However, the elimination of the gender difference in *Tests Passed* for those later assigned to PS2 does indicate that subverting gender stereotypes in item contexts may have helped eliminate performance gaps between majority and minority participants.

[5] The remaining participants did not explicitly discuss the utility of the context.

Table 4: MWU tests on Neutral items. Large effects and significant results are bolded.

Metric	Group	CLES-Maj	p	MMM Maj	MMM Min
Tests Passed	PS1	**0.72**	0.084	2/6/7	1/4/7
	PS2	**0.83**	**0.012**	5/6/7	0/5/6
	PS3	0.55	0.696	6/7/7	6/7/7
	PS4	0.58	0.407	5/7/7	4/7/7
Time Reading	PS1	0.34	0.210	0/1/2	1/2/9
	PS2	0.39	0.426	0/1/4	0/1/6
	PS3	0.62	0.438	1/1/16	0/1/3
	PS4	0.48	0.891	0/1/6	0/1/5
Between Keystrokes	PS1	0.50	1.0	3/9/14	2/8/26
	PS2	0.56	0.640	8/9/12	2/9/25
	PS3	0.69	0.183	4/7/16	4/5/7
	PS4	0.37	0.286	2/5/11	2/6/14

Table 5: MWU tests on PS1 - PS4. Large effects are bolded.

Metric	Data	CLES-Maj	p	MMM Maj	MMM Min
Tests Passed	PS1	0.57	0.544	6/7/7	4/7/7
	PS2	0.65	0.236	6/7/7	5/6/7
	PS3	0.42	0.579	2/5/7	4/6/7
	PS4	0.65	0.228	4/6/7	2/5/7
Time Reading	PS1	0.39	0.404	0/1/2	1/1/9
	PS2	**0.73**	0.080	1/2/3	0/1/5
	PS3	0.59	0.536	1/2/28	1/4/6
	PS4	0.49	0.935	1/2/7	1/2/8
Between Keystrokes	PS1	0.70	0.107	1/4/9	1/2/5
	PS2	0.65	0.249	3/4/7	1/3/8
	PS3	0.42	0.588	4/10/14	4/11/16
	PS4	0.47	0.848	2/10/17	4/10/21

5 Discussion

We did not find evidence that the use of stereotyped context created gender disparities in performance. There were *no significant differences* in majority and minority participants' performance on the four stereotyped problem sets at either $\alpha = 0.05$ or the corrected $\alpha = 0.0167$. We see two likely explanations for this observation. First, the context may have triggered stereotype threat in *all* participants. This explanation is aligned with prior work, which suggests that both positively- and negatively-stereotyped students are vulnerable to stereotype threat. Second, the context may have triggered stereotype threat in *none* of the participants. It is possible that the stereotypes were not salient to participants, or that participants noticed the stereotypes but did not experience stereotype threat because computing is not an important part of their identity [21]. Other factors, including concept familiarity and self-efficacy, may also have influenced students' performance and interacted with the stereotyped context, although our results do not suggest that there were systematic differences in such knowledge.

Interestingly, our results suggest that using subversive contexts may help close performance gaps between students from the gender majority and minority. Specifically, among the participants routed to PS2, those who identified as being from the gender majority passed significantly more test cases on the Neutral items, but not the Disruptive items from PS2. However, since there were no other pre-existing differences in performance, we do not know whether this trend applies more broadly. Also, it is unclear if the gap was closed because the subversive contexts positively impacted minority participants,[6] negatively impacted majority participants, or both. Given that we observed a large effect in *Time Reading* for PS2 in favor of minority participants, it is possible that majority participants were more surprised by the subversive contexts (perhaps because it contradicted their preconceived notions about gender) and needed to re-read the problems to fully comprehend them.

6 Threats to Validity

To our knowledge, this work is the first to explore how stereotyped context can affect student performance on programming problems. While our findings add nuance to the literature on contextualization and stereotype threat, there are threats to internal and external validity. First, the items were not assigned as part of a graded course, so some participants may not have cared enough about their performance to experience stereotype threat. We tried to mitigate this by only including data from participants who meaningfully attempted all items. Second, there was an imbalance in the number of majority and minority participants across problem sets, because we could not factor demographics into the routing mechanism — as discussed in Section 3.4, asking participants for their demographics before/during the study may trigger stereotype threat [12, 13]. Third, we cannot be sure our content manipulations were salient to participants. However, we did center the items in our analyses around a gender stereotype, so participants likely recognized them when reading the prompts. Fourth, our sample sizes are relatively small, so we may not have had enough statistical power to detect subtle differences. Lastly, our study has selection bias. Participants had passed a CS course and wanted more practice with Python; our results may only apply to motivated or high-achieving students.

Future work can use similar logging infrastructure and item-pair methods to repeat this study in a more controlled environment, at a larger scale, and with participants that may be more representative of the overall student population in computing. It can also evaluate how referencing gender stereotypes in programming problems influences students' sense of belonging or self-efficacy. It is possible that the use of subversive item contexts can help students from the gender minority feel seen and improve their affective outcomes.

7 Practical Implications and Conclusion

We find that, when controlling for prior knowledge, gender majority and minority students react similarly to seeing stereotyped context, regardless of whether the context reinforces or subverts gender stereotypes. Yet, when there are pre-existing differences in performance, gender minority students may benefit more from seeing programming problems whose contexts subvert gender stereotypes. Our results do not clearly advocate for or against the use of stereotyped context. Rather, they add to the broader evidence suggesting that stereotype threat effects are context dependent.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2417014, 2100296, 2122950 and gifts from Microsoft, Adobe, and Google.

[6]While minority participants were not all women, they were all from genders marginalized in computing, and they may have also found the subversive context encouraging.

References

[1] Adam L Alter, Joshua Aronson, John M Darley, Cordaro Rodriguez, and Diane N Ruble. 2010. Rising to the Threat: Reducing Stereotype Threat by Reframing the Threat as a Challenge. *Journal of Experimental Social Psychology* 46, 1 (2010), 166–171. https://doi.org/10.1016/j.jesp.2009.09.014

[2] Dennis Bouvier, Ellie Lovellette, John Matta, Bedour Alshaigy, Brett A. Becker, Michelle Craig, Jana Jackova, Robert McCartney, Kate Sanders, and Mark Zarb. 2016. Novice Programmers and the Problem Description Effect. In *Proceedings of the 2016 ITiCSE Working Group Reports* (Arequipa, Peru) *(ITiCSE '16)*. Association for Computing Machinery, New York, NY, USA, 103–118. https://doi.org/10.1145/3024906.3024912

[3] Sapna Cheryan, Victoria C Plaut, Paul G Davies, and Claude M Steele. 2009. Ambient Belonging: How Stereotypical Cues Impact Gender Participation in Computer Science. *Journal of Personality and Social Psychology* 97, 6 (2009), 1045. https://doi.org/10.1037/a0016239

[4] Sapna Cheryan, Victoria C Plaut, Caitlin Handron, and Lauren Hudson. 2013. The Stereotypical Computer Scientist: Gendered Media Representations as a Barrier to Inclusion for Women. *Sex Roles* 69, 1 (2013), 58–71. https://doi.org/10.1007/s11199-013-0296-x

[5] Susan F. Chipman, Sandra P. Marshall, and Patricia A. Scott. 1991. Content Effects on Word Problem Performance: A Possible Source of Test Bias? *American Educational Research Journal* 28, 4 (1991), 897–915. https://doi.org/10.3102/00028312028004897

[6] Michelle Craig, Jacqueline Smith, and Andrew Petersen. 2017. Familiar Contexts and the Difficulty of Programming Problems. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '17)*. Association for Computing Machinery, New York, NY, USA, 123–127. https://doi.org/10.1145/3141880.3141898

[7] Gabriella Daroczy, Magdalena Wolska, Walt Detmar Meurers, and Hans-Christoph Nuerk. 2015. Word Problems: A Review of Linguistic and Numerical Factors Contributing to Their Difficulty. *Frontiers in Psychology* 6 (2015), 348. https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2015.00348/full

[8] Stuart Garner. 2002. Reducing the Cognitive Load on Novice Programmers. In *Proceedings of EdMedia + Innovate Learning 2002*, Philip Barker and Samuel Rebelsky (Eds.). Association for the Advancement of Computing in Education (AACE), Denver, Colorado, USA, 578–583. https://www.learntechlib.org/p/10329

[9] Richard A Griggs and James R Cox. 1982. The Elusive Thematic-Materials Effect in Wason's Selection Task. *British Journal of Psychology* 73, 3 (1982), 407–420. https://doi.org/10.1111/j.2044-8295.1982.tb01823.x

[10] James L Hilton and William Von Hippel. 1996. Stereotypes. *Annual Review of Psychology* 47, 1 (1996), 237–271. https://doi.org/10.1146/annurev.psych.47.1.237

[11] Kenneth R Koedinger and Mitchell J Nathan. 2004. The Real Story Behind Story Problems: Effects of Representations on Quantitative Reasoning. *Journal of the Learning Sciences* 13, 2 (2004), 129–164. https://doi.org/10.1207/s15327809jls1302_1

[12] Amruth N. Kumar. 2010. The Effect of Soliciting Demographic Data on the Performance of Students on Online Tests. In *2010 IEEE Frontiers in Education Conference (FIE)*. T1F–1–T1F–4. https://doi.org/10.1109/FIE.2010.5673114

[13] Amruth N. Kumar. 2012. A Study of Stereotype Threat in Computer Science. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2325296.2325361

[14] Juho Leinonen, Paul Denny, and Jacqueline Whalley. 2021. Exploring the Effects of Contextualized Problem Descriptions on Problem Solving. In *Proceedings of the 23rd Australasian Computing Education Conference* (Virtual, SA, Australia) *(ACE '21)*. Association for Computing Machinery, New York, NY, USA, 30–39. https://doi.org/10.1145/3441636.3442302

[15] Ellie Lovellette, Dennis J Bouvier, and John Matta. 2024. Contextualization, Authenticity, and the Problem Description Effect. *ACM Trans. Comput. Educ.* 24, 2, Article 26 (April 2024), 32 pages. https://doi.org/10.1145/3643864

[16] Ellie Lovellette, John Matta, Dennis Bouvier, and Roger Frye. 2017. Just the Numbers: An Investigation of Contextualization of Problems for Novice Programmers. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 393–398. https://doi.org/10.1145/3017680.3017726

[17] Henry B Mann and Donald R Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* (1947), 50–60. http://doi.org/10.1214/aoms/1177730491

[18] Allison Master, Sapna Cheryan, and Andrew N Meltzoff. 2016. Computing Whether She Belongs: Stereotypes Undermine Girls' Interest and Sense of Belonging in Computer Science. *Journal of Educational Psychology* 108, 3 (2016), 424. https://doi.org/10.1037/edu0000061

[19] Zachary Mastrich and Ivan Hernandez. 2021. Results Everyone Can Understand: A Review of Common Language Effect Size Indicators to Bridge the Research-Practice Gap. *Health Psychology* 40, 10 (2021), 727. https://doi.org/10.1037/hea0001112

[20] Briana B. Morrison. 2015. Computer Science Is Different! Educational Psychology Experiments Do Not Reliably Replicate in Programming Domain. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (Omaha, Nebraska, USA) *(ICER '15)*. Association for Computing Machinery, New York, NY, USA, 267–268. https://doi.org/10.1145/2787622.2787744

[21] Hannah-Hanh D Nguyen and Ann Marie Ryan. 2008. Does Stereotype Threat Affect Test Performance of Minorities and Women? A Meta-Analysis of Experimental Evidence. *Journal of Applied Psychology* 93, 6 (2008), 1314. https://doi.org/10.1037/a0012702

[22] Kurt Reusser. 1988. Problem Solving Beyond the Logic of Things: Contextual Effects on Understanding and Solving Word Problems. *Instructional Science* 17, 4 (1988), 309–338. https://doi.org/10.1007/BF00056219

[23] E. Soloway. 1986. Learning to Program = Learning to Construct Mechanisms and Explanations. *Commun. ACM* 29, 9 (Sept. 1986), 850–858. https://doi.org/10.1145/6592.6594

[24] Steven J Spencer, Christine Logel, and Paul G Davies. 2016. Stereotype Threat. *Annual Review of Psychology* 67, 1 (2016), 415–437. https://doi.org/10.1146/annurev-psych-073115-103235

[25] Steven J Spencer, Claude M Steele, and Diane M Quinn. 1999. Stereotype Threat and Women's Math Performance. *Journal of Experimental Social Psychology* 35, 1 (1999), 4–28. https://doi.org/10.1006/jesp.1998.1373

[26] Claude M Steele and Joshua Aronson. 1995. Stereotype Threat and the Intellectual Test Performance of African Americans. *Journal of Personality and Social Psychology* 69, 5 (1995), 797. https://doi.org/10.1037/0022-3514.69.5.797

[27] John Sweller. 2010. Element Interactivity and Intrinsic, Extraneous, and Germane Cognitive Load. *Educational Psychology Review* 22 (2010), 123–138. https://doi.org/10.1007/s10648-010-9128-5

[28] András Vargha and Harold D Delaney. 2000. A Critique and Improvement of the "CL" Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* 25, 2 (2000), 101–132. https://www.jstor.org/stable/1165329

[29] Margaret Walsh, Crystal Hickey, and Jim Duffy. 1999. Influence of Item Content and Stereotype Situation on Gender Differences in Mathematical Problem Solving. *Sex Roles* 41, 3 (1999), 219–240. https://doi.org/10.1023/A:1018854212358

[30] Peter C Wason and Diana Shapiro. 1971. Natural and Contrived Experience in a Reasoning Problem. *Quarterly Journal of Experimental Psychology* 23, 1 (1971), 63–71. https://doi.org/10.1080/00335557143000068