# Improving Attitudes to Computer Programming in a First-Year Interdisciplinary Science Course

Amy A.Z. Zhao, Sara Davies, Ava Greenwood, Timothy J. McIntyre

Corresponding author: Sara Davies (sara.davies@uq.edu.au)
School of Mathematics and Physics, The University of Queensland, Brisbane, QLD 4072, Australia

**Keywords**: interdisciplinary science; computer programming; attitudes

## Abstract

Over five years, we measured students' attitudes and perceptions of computer programming in a compulsory first-year interdisciplinary science course. During this period, we collected 1057 matched survey responses at the beginning and end of the semester. We found a significant improvement in students' attitudes toward programming over the course of the semester, and there was no difference in the extent of improvement between students with and without prior programming experience. We conclude that an interdisciplinary course in which computer programming instruction is embedded with problem-based learning activities is effective in introducing computer programming to science students, regardless of their initial level of programming experience.

## Introduction

Computer programming is a skill that can offer many benefits in the context of scientific learning. These benefits include abstract representation, structured problem decomposition, iterative and recursive thinking, and conditional logic (Jenson & Droumeva, 2016). There has been substantial research investigating the relationship between students' attitudes towards science, technology, engineering, and mathematics (STEM) topics and their career interests (e.g. Wiebe, Unfried, & Faber, 2018). Specifically, there is evidence that attitudes towards STEM subjects have been linked to STEM knowledge achievement (Han, Kelley, & Knowles, 2021; Mao, Cai, He, Chen, & Fan, 2021; Else-Quest, Mineo, & Higgins, 2013). This paper investigates first-year students' attitudes toward computer programming before and after taking an interdisciplinary science course with computer programming embedded into the content.

There is a consensus in the literature that teaching and learning computer programming can be challenging, especially for novice learners (Robins, Rountree, & Rountree, 2003; Teague, 2011; Medeiros, Ramalho, & Falcão, 2019). In a systematic review of approaches to teaching programming, Vihavainen, Airaksinen, & Watson (2014) found that, on average, teaching interventions can improve programming pass rates by nearly one-third, with no statistical differences in effectiveness observed between the different types of interventions. Teaching interventions described in their survey included collaboration, contextualisation, group work, game-theme component, and increased support.

In a review of the literature on teaching and learning computer programming in higher education, Medeiros et al. (2019) find that the biggest challenges novice programming students face are related to 'motivation and engagement, problem-solving, and the syntax of programming languages.' A recent study by Krakowski, Greenwald, Roman, Morales, and Loper, (2023), in a middle school context, showed that using real-world problems and authentic integration of computational thinking can contribute to learning gains and improvements in attitudes towards STEM professions. Building on these findings, our study looks at the effects of having computer programming embedded within a first-year interdisciplinary science course, where the aim is to improve attitudes and perceptions toward computer programming.

The courses in consideration for this paper are SCIE1000 (Theory and Practice in Science) and SCIE1100 (Advanced Theory and Practice in Science) at The University of Queensland (UQ). UQ is an Australian research-intensive institution that is ranked in the top 100 universities worldwide (for example in the Times Higher Education World University Rankings) and enrols approximately 1500 new Bachelor of Science (BSc) students each year. Students enrolled in science degrees at UQ can major in a range of scientific disciplines and all students in the BSc program are required to take SCIE1000. The course is offered in all three semesters: Semester 1, 2, and 3 (summer). Students enrolled in a Bachelor of Advanced Science (BAdvSc) are required to take SCIE1100, offered only in Semester 1. Both courses are typically taken by students in the first year and semester of their degrees. SCIE1000 and SCIE1100 cover the same topics and have the same learning objectives, but the SCIE1100 stream has a deeper focus on research applications, slightly differentiated assessment, and provides a cohort experience. For brevity, we will refer to both courses collectively as SCIE1000/1100.

SCIE1000/1100 has been offered at UQ since 2008. Matthews , Adams, and Goos (2010) previously investigated attitudes among biology students enrolled in SCIE1000, finding that while students gained a positive appreciation for mathematics, their appreciation for computer programming was unchanged (comparing the 2008 cohort to the 2009 cohort). In 2018, SCIE1000/1100 became a compulsory course for science students and new course coordinators were hired to redevelop the learning materials and approaches.

The current learning objectives of SCIE1000/1100 cover a broad range of foundational scientific skills such as the creation and use of mathematical models in science, writing computer programs for a variety of scientific contexts, philosophical reasoning, and science communication to various audiences. SCIE1000/1100 has been designed to cater to the diverse interests of students by incorporating a mixture of science and mathematics with an unusual teaching model. The course content is delivered by a multidisciplinary team, with most lectures co-delivered by a mathematician and a scientist. A four-lecture unit on the philosophy of science is jointly delivered by two philosophy lecturers. Students learn basic skills in scientific computing in Python from a computer scientist in the Python Class, and Workshops are led by tutors with various science, mathematics, and computer programming backgrounds. More details regarding the course content and the pedagogical approach of SCIE1000/1100 are detailed in Matthews, Adams, and Goos(2009).

Python programming is embedded in SCIE1000/1100 in various ways. The course workbook includes a Python Appendix that covers key concepts and examples of key topics such as variables, inputs, debugging, conditionals, loops, arrays, writing functions, and plotting graphs. In addition, the weekly Python Class explores these topics in a lecture setting with coding demonstrations and provides an opportunity for students to ask questions about the topics. The Python Class is optional but students with no prior computer programming experience are highly encouraged to attend. These classes are recorded and made available for enrolled students to access at any time.

Each week, the topics covered in the Python Class are put into practice in Workshops, which provide students with contextualisation, group work and support. Workshop tasks ask that students interact with Python problems in Jupyter notebooks. The problems are based on different scientific contexts each week and the Workshops offer the opportunity for students to work collaboratively on programming tasks within those contexts, with activities and group discussions facilitated by a team of tutors. Workshop participation contributes to 10% of a student's overall course grade.
The summative assessment of Python topics primarily occurs in the Python and Communication Assignment, which was first developed and added to the course in 2018. The assignment asks students to create code that could be used in a museum exhibit. Students need to demonstrate each of the key topics of programming mentioned above and are required to communicate the scientific context of the assignment to both the end user of the exhibit as well as through comments within the code. Some examples of scientific contexts used for this assignment include: the echolocation of whales, the size and shape of penguin huddles, daylight hour models, detection of exoplanets, a Serengeti chase, and plastic pollution modelling. The Python and Communication Assignment contributes to 15% of a student's overall course grade.

Piggott, Herke, McIntyre, and Bulmer (2019) conducted an initial study investigating the attitudes and perceptions of science, mathematics, and computer programming in SCIE1000/1100 measured in a single semester (2019 Semester 1). For students with prior programming experience, there was an observed improvement in overall perceptions of programming in the last week of semester compared to the first week. However, a limitation of this study was that initial attitudes toward computer programming were not recorded for those with no prior programming experience.

In the present study, we expand on Piggott et al. (2019) by collecting data across a broader time frame from 2019 to 2023. This extended data collection captures changes in attitudes beyond a single semester. In addition, we measure the baseline attitudes and perceptions of programming in students with no prior programming experience, acknowledging that these students may hold pre-existing views about programming. In addition to reporting on attitudes to computer programming, we also measure and report on attitudes toward science and mathematics.

## Aims

Our aims extend on those of Piggott et al. (2019). We consider the following three research questions:

Q1. Is there a measurable change in student attitudes to and perceptions of computer programming before and after SCIE1000/1100?

Q2. Does prior programming experience have any impact on changes in attitudes to and perceptions of computer programming?
Q3. Which learning activities (and to what extent) contribute to changes in perceptions?

## Methods

### Participants and Procedure

Students were given anonymous surveys at the beginning (Week 1) and end (Week 12 or 13) of semester; we refer to these as the pre- and post-survey, respectively. Data was collected across 14 semesters from 2019 Semester 2 to 2023 Semester 3. Data analysis was performed only on those participants who were both at least 18 years of age and who gave informed consent for their data to be used in this study. Human ethics approval was obtained for this project (UQ Human Ethics Approval #2019000164).

After removing data for respondents who were under 18 or did not give informed consent, we obtained 3951 pre-survey responses, and 1901 post-survey responses. We matched participants' pre- and post-survey responses within each semester based on the following non-identifying information: their gender[1], the last three digits of their phone number, the state in which they completed secondary schooling, and their first initial. We successfully matched 1057 participants. Specifically, 352 had prior programming experience (182 man or male responses, 162 women or female responses, and 8 non-binary/ I use a different term) and 705 had no prior programming experience (245 man or male responses, 449 woman or female responses, 10 non-binary/ I use a different term, and 1 prefer not to say). We note that the percentage of matched responses out of the total number enrolled after census date[2] ranged between 7% to 30% across semesters (full table in Supplementary materials).

### Measures

Attitudes and perceptions of science, mathematics, and computer programming were measured using 12 items from Piggott et al. (2019) which were adapted from Parsons, Adler, and Meece (1984). These items, detailed in Table 1, assessed self-efficacy, usefulness, and enjoyment for each discipline. Internal consistency for each subgroup across the matched pre- and post-survey observations (Table 2) ranged from 'acceptable' to 'excellent' (George & Mallery, 2016). We note that there is slightly lower internal consistency among the science measures. This may be because the survey asks students to reflect on “science” as a general

[1] Prior to 2022, students were asked to select their gender from the following options: male, female, other, or prefer not to say. From 2022 onwards, the gender response options were changed to: woman, man, non-binary, prefer not to say, or I use a different term. Those who responded “other” prior to 2022 have been reclassified as “non-binary/ I use a different term”.

[2] The last day the student can make changes to enrolment without financial penalty. This is typically one month after the beginning of the semester.

concept and given the broad scope of scientific majors represented among the cohort, students' prior exposure to science and conception of science is likely considerably more variable than their conception of mathematics or programming. In the pre-survey, students were asked, 'Have you done any computer programming before?', responding with either 'Yes' or 'No'. Both the pre- and post-survey included the question, 'What grade do you expect to achieve in SCIE1000/SCIE1100?' with responses on a scale of 1 to 7, corresponding to the UQ grading system.

Table 1. Survey items assessing student attitudes and perceptions toward computer programming. We note that Science and Mathematics attitudes and perceptions are assessed by replacing 'computer programming' with 'science' and 'mathematics', respectively. Taken from Piggott et al. (2019).

| Subgroup | Item | Descriptor of 1 | Descriptor of 4 | Descriptor of 7 |
|---|---|---|---|---|
| Self-efficacy | How good at **computer programming** are you? | Not good at all | O.K. | Very good |
| | Some students find that they are better at one subject or activity than another. Compared to most of your other activities, how good are you at **computer programming**? | Not as good as other activities | About the same | A lot better than other activities |
| | If you were to list all the students in your class from the worst to the best in **computer programming**, where would you put yourself? | One of the worst | In the middle | The best |
| | How good would you be at learning something new in **computer programming**? | Not good at all | O.K. | Very good |
| Usefulness | In general, how useful is what you learn in **computer programming**? | Not useful at all | | Very useful |
| | Some students find what they learn in one subject or activity more useful than what they learn in another. Compared to most of your other activities, how useful is what you learn in **computer programming**? | Not as useful as what I learn in other activities | About the same | A lot more useful than what I learn in other activities |

| | | | | |
|---|---|---|---|---|
| | For me, <u>being good</u> at **computer programming** is | Not important at all | | Very important |
| | Some students believe that it is more important to be better at one subject or activity than another. <u>Compared to most of your other activities</u>, how <u>important</u> is it to you <u>to be good</u> at **computer programming**? | Not as important as being good in other activities | About the same | A lot more important than being good in other activities |
| Enjoyment | How much do you <u>like</u> doing **computer programming**? | A little | Some | A lot |
| | Some students find that they like one subject or activity much more than another. <u>Compared to most of your other activities</u>, how much do you <u>like</u> **computer programming**? | Not as much as other activities | About the same | A lot more than other activities |
| | How good do you think you would be in a career requiring **computer programming** skills? | Not good at all | O.K. | Very good |
| | In general, you find working on **computer programming** problems | Very boring | O.K. | Very interesting |

Table 2. Internal consistency (Cronbach's alpha, Cronbach, 1951) for each subgroup and discipline.

| | Computer programming | Science | Mathematics |
|---|---|---|---|
| Self-efficacy (pre-survey) | 0.89 | 0.78 | 0.93 |
| Usefulness (pre-survey) | 0.91 | 0.75 | 0.89 |
| Enjoyment (pre-survey) | 0.92 | 0.79 | 0.92 |

| | | | |
|---|---|---|---|
| Self-efficacy (post-survey) | 0.93 | 0.83 | 0.95 |
| Usefulness (post-survey) | 0.94 | 0.84 | 0.91 |
| Enjoyment (post-survey) | 0.93 | 0.86 | 0.94 |

Students were asked to rate the extent to which specific learning activities changed their perceptions of computer programming in the post-survey, e.g. 'The following things in SCIE1000/SCIE1100 changed my perceptions of computer programming', with the following learning activities listed: Python Class, Workshops, and the Python and Communication Assignment. Students could respond: 'Negatively', 'No change', 'Positively', or 'Did not use'. While other activities were included in the post-survey for certain semesters, we focus on these three activities due to their consistent inclusion across all 14 semesters.

**Data Analyses**
Data were analysed in R (R Core Team, 2021). Overall attitudes and perceptions (OAP) for each discipline (science, mathematics, and computer programming) were calculated for each student by averaging across each set of 12 survey items. The change in OAP for each discipline was calculated by subtracting the pre-survey OAP from the post-survey OAP.

We calculated the effect size of the change in OAPs using Cohen’s d (Cohen, 1988). Independent groups t-tests were used to investigate whether there was a significant difference between the change in computer programming OAPs between those with and without prior programming experience. We used single sample t-tests to investigate whether there was a significant change in OAPs within each discipline. We use the coefficient of determination ($R^2$) to describe the percentage variance explained by ratings of learning activities with changes in computer programming OAP.

## Results

Overall, we observed that introducing programming to students through this interdisciplinary and context driven science course resulted in a significant positive change in computer programming OAP in students who reported prior programming experience with a moderate effect size, and we observed similar results for those with no prior programming experience (Table 3). We found no evidence that there was a difference in the change in programming OAP between those with and without prior programming experience (t(853.41) = 1.64, p = 0.102, d = 0.10). In other words, we observed the same improvement in students’ attitudes toward programming regardless of whether students had prior programming experience. We also found no significant change in mathematics OAP and a small but statistically significant negative change in science OAP.

Table 3. Mean and standard deviation for pre- and post-survey OAP for matched students.

| | Pre-survey M(SD) | Post-survey M(SD) | Change M(SD) | t | df | p | Cohen’s d |
|---|---|---|---|---|---|---|---|
| Programming OAP (n = 1021) | 3.46(1.22) | 3.99(1.35) | 0.53(1.14) | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Prior experience (n = 352) | 4.07(1.16) | 4.53(1.32) | 0.45(0.98) | 8.65 | 351 | <0.001 | 0.46 |
| No prior experience (n = 669) | 3.14(1.12) | 3.72(1.28) | 0.57(1.22) | 12.09 | 668 | <0.001 | 0.47 |
| Science OAP (n = 1057) | 5.44(0.68) | 5.33(0.86) | -0.11(0.67) | -5.45 | 1056 | <0.001 | -0.17 |
| Mathematics OAP (n = 1057) | 4.58(1.17) | 4.57(1.27) | -0.02(0.75) | -0.72 | 1056 | 0.469 | -0.02 |

When we visualise changes in programming OAP over time (Figure 1), we observe several semesters of significant positive change in both students with and without prior programming experience (Figure 1). There was no significant change in mathematics OAP and only two semesters with a significant negative change in science OAP. Covid-19 led to classes being moved fully online in Semester 1, 2020. From Semester 2, 2020, until Semester 1, 2023, students were able to complete SCIE1000/1100 either in-person or externally (fully online), with this option also available in the subsequent summer semester. We have not specifically investigated differences in the responses between students enrolled in-person and those enrolled externally.

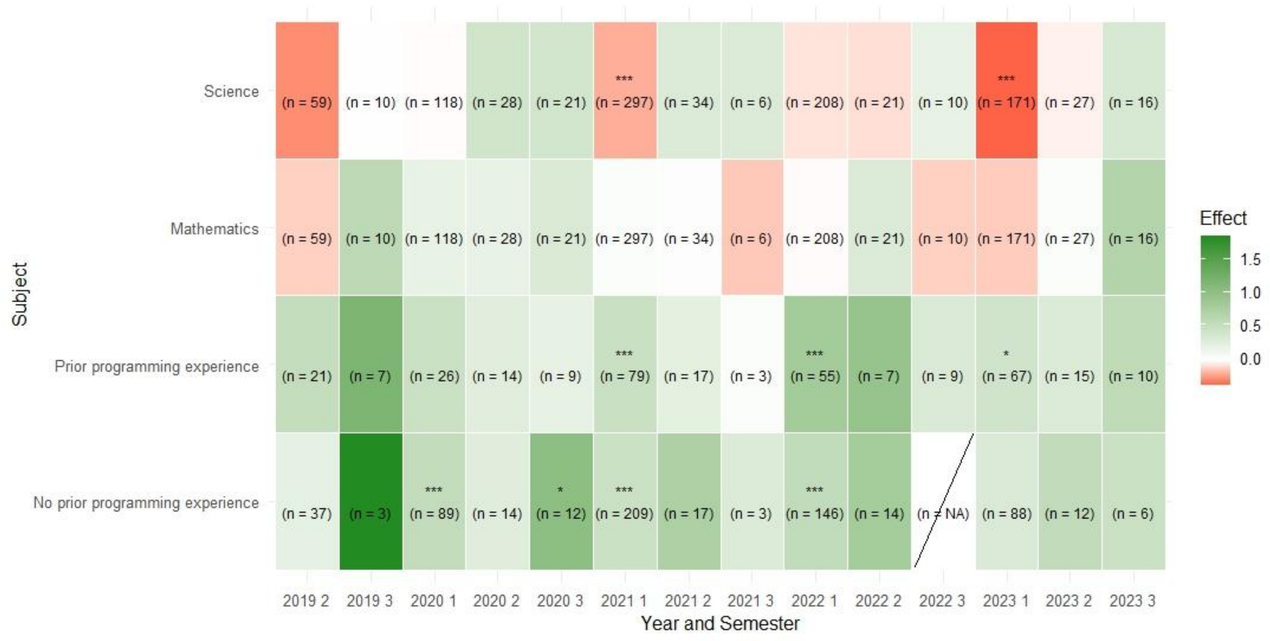

Figure 1. Heatmap of the change in OAPs over each measured semester. Green indicates a positive effect size while red indicates a negative effect size. We adjusted p values to account for multiple tests using the Bonferroni-Holm method. Note: * < .05, ** < .01, *** < .001

In Figure 2, we compare the average improvement in each of the 12 individual OAP items between students who reported prior and no prior programming experience. Similar to Piggott et al. (2019), we saw little to no change in mathematics and science attitudes, but we saw substantial positive changes in items measuring the change in self-efficacy and enjoyment of programming. In contrast, OAP changes under the category of usefulness were much smaller. Students generally entered SCIE1000/1100 with a higher OAP for usefulness than for self-efficacy and enjoyment of programming. We hypothesise that the programming activities impacted their ability to program but weren't sufficiently in-depth to further highlight the usefulness of programming in their own discipline area.

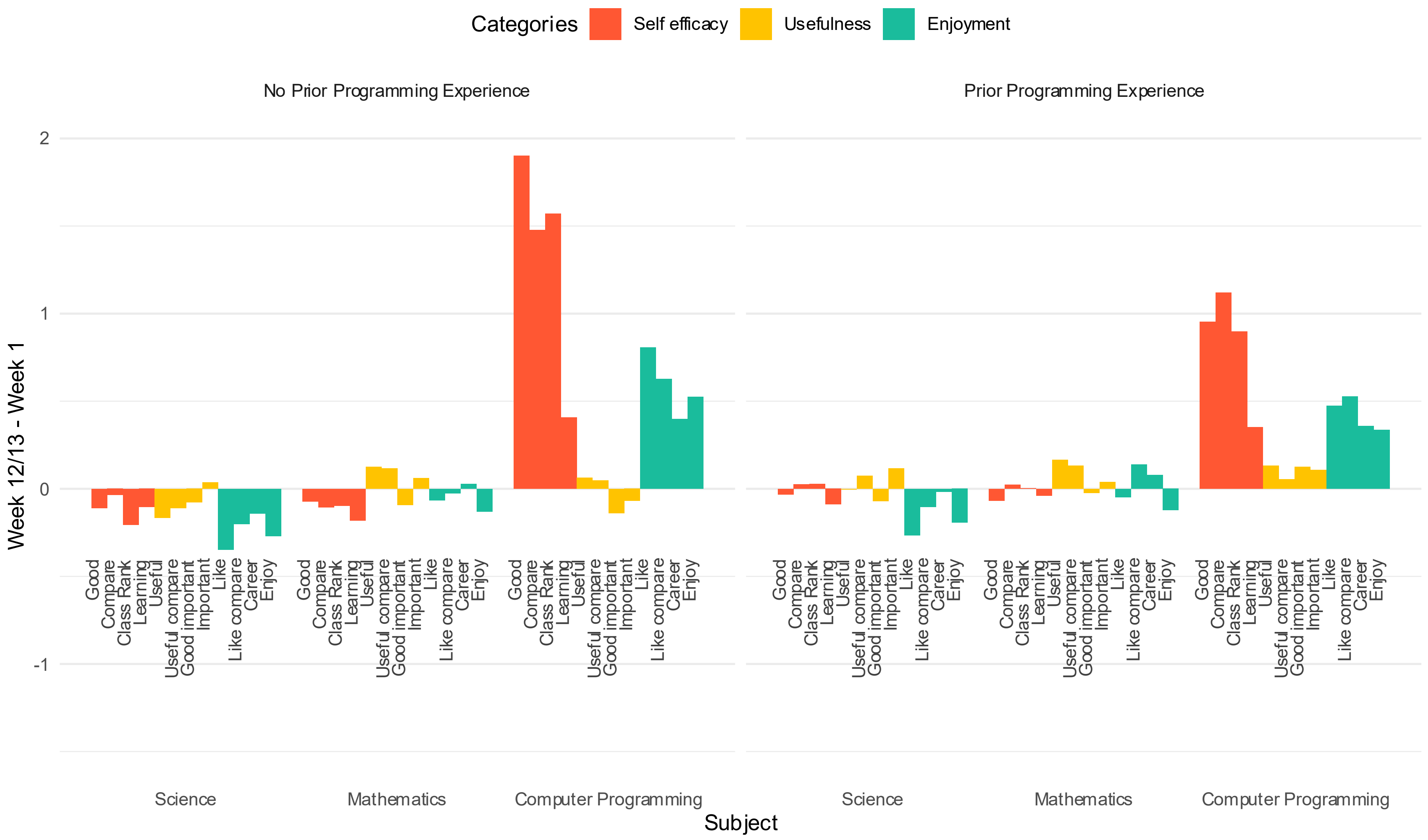
Categories
Self efficacy
Usefulness
Enjoyment
No Prior Programming Experience
Prior Programming Experience
Week 12/13 - Week 1
2
1
0
-1
Good
Compare
Class Rank
Learning
Useful
Useful compare
Good important
Important
Like
Like compare
Career
Enjoy
Science
Mathematics
Computer Programming
Subject

Figure 2: Changes in programming OAP items compared across students with no prior and prior programming experience.

A majority of students reported that Workshops and the Python and Communication Assignment positively impacted their perceptions of computer programming (Figure 3). The Python Classes had the highest percentage of non-engagement, particularly for those with prior programming experience (31%). The Python and Communication Assignment had the highest percentage of negative ratings out of the three activities, with 18% of students with prior programming experience and 13% of those with no prior programming experience reporting that the assignment had a negative impact on their perceptions of programming.
Figure 3. Proportions of student responses (positive, no change, negative, did not use) regarding the impact of specific learning activities on programming perceptions.

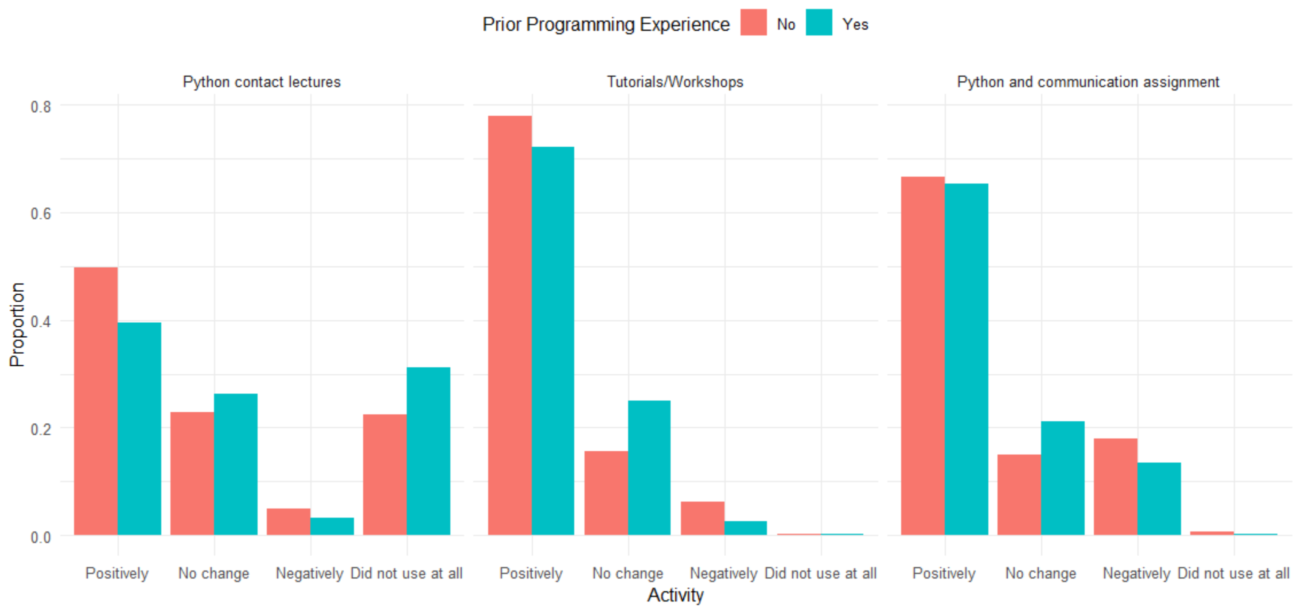

Figure 3. Proportions of student responses (positive, no change, negative, did not use) regarding the impact of specific learning activities on programming perceptions.

We found that self-reported responses to the Python and Communication Assignment explained the most variance in change in programming OAP ($R^2 = 0.10$). A Tukey HSD test revealed that students who positively rated the Python and Communication Assignment had a significantly higher improvement in programming OAP ($M = 0.78$) than those who gave a neutral rating ($M = 0.26$) ($p < 0.001$), and both those who gave positive and neutral ratings had significantly larger improvements than students who gave negative ratings ($M = -0.14$) ($p < 0.001$). Responses to Workshops explained the second highest variance in change in programming OAP ($R^2 = 0.06$) followed by the Python Class ($R^2 = 0.02$). We also found that there were no interactions between programming experience and any of the learning activities on programming OAP ($p \geq 0.054$).

**General differences between those with and without prior programming experience**
We investigated whether there were general differences between those with and without programming experience. Prior programming experience was associated with the gender of the student ($\chi^2(1) = 27.32$, $p < .001$). Specifically, students with prior programming experience were more likely to be men (52% men, 46% women, 2% non-binary/ I use a different term) while students without programming experience were more likely to be

women (35% men, 64% women, 1% non-binary/ I use a different term / prefer not to say). Students with no prior programming experience also tended to expect a significantly lower grade at the beginning of semester (M = 5.54) compared to students with prior experience (M = 5.76, t(747.25) = -3.76, p < .001).

## Discussion

### On research questions Q1 and Q2

Across five years, we found that embedding computer programming in a first-year interdisciplinary science course significantly improved students' attitudes and perceptions of programming over the course of the semester, shown by a moderate effect size. Specifically, we observed large improvements in attitudes relating to self-efficacy and enjoyment of programming (Figure 2). Our findings over a five-year period replicated those reported by Piggott et al. (2019) regarding 2019 Semester 1 observations, giving us a reliable indicator that the positive changes in attitudes to computer programming are consistent over time. This motivates us to conclude that embedding computer programming into an interdisciplinary science course can be effective at improving attitudes towards computer programming.

### On research question Q3

We identified that students' evaluations of the Python and Communication Assignment were the strongest predictors of changes in programming OAP, compared to other learning activities, highlighting the benefits of this assessment item on improving programming attitudes. Considering the percentage of students who reported various learning activities as having a positive effect on their perceptions of computer programming (see Figure 3), we observed results consistent with those reported by Piggott et al. (2019). The highest percentage of positive evaluations of learning activities were of the Workshops (Prior: 72%, No prior: 78%), followed by the Python and Communication Assignment (Prior: 65%, No prior: 67%), and Python Class (Prior: 39%, No prior: 50%). When we use these evaluations to predict changes in programming OAP, we found the Python and Communication Assignment evaluations explained 10% of the variance in changes in programming OAP, which is a medium effect; this was followed by responses to Workshops (6%), and the Python Class (2%). Specifically, positive evaluations of these learning activities were associated with higher improvements in programming OAP.

The Python and Communication Assignment is a summative piece of assessment intended to challenge students and give students agency, creative freedom, and ownership of the code they produce. Thus, it is not surprising that a student who self-reports a positive impact of the Python and Communication Assignment on their perceptions of programming would indeed see an improvement in their programming OAP. The higher percentage of positive evaluations of Workshops and the Python and Communication Assignment may be explained by these two learning activities involving active engagement, while the Python Class is more passive. Our findings suggest that including a significant authentic assessment piece that involves applying computer programming to an applied, interdisciplinary context is beneficial in changing students' attitudes when students engage meaningfully with the task.

### Comparing OAPs across computer programming, mathematics, and science

While SCIE1000/1100 was effective in improving programming attitudes, we found no change in OAP for mathematics and a small significant negative change for science. We contrast this finding with an earlier study by Matthews et al. (2010) which found that biology students enrolled in SCIE1000 gained a positive appreciation for mathematics. However, we note that the methodology used by Matthews et al. (2010) involved asking students a single question about their attitudes towards mathematics ('Rate the level to which you feel enthusiastic about math') with a 3-point Likert score and the results were obtained by comparing the 2008 cohort to the 2009 cohort.

Overall, the average post-survey OAPs for science and mathematics were high (above the midpoint of 4 out of 7), demonstrating that students' attitudes at the end of semester remained positive (Table 3), and post-survey OAPs for computer programming (with and without prior experience) fell below pre-survey OAPs for mathematics and science. We believe that the slight negative change in science OAP could be attributed to higher initial expectations from first-year students coming from high school. In particular, Crisp et al. (2009) reported that first-year students at the University of Adelaide expected university to be different to high school, but some of their expectations of university were unrealistic (such as getting personalised feedback from their lecturer on their draft work). Successful transition from high school to university does not only rely on academic ability but also the ability to adapt and apply greater autonomy and responsibility than what students initially expect (Brinkworth, McCann, Matthews, & Nordström,, 2009). Another possible explanation for a decrease in science attitudes could be that students who are interested in science would not necessarily choose to enrol in a course that requires study outside their discipline of interest. Since SCIE1000/1100 is a compulsory course in the Bachelor of Science and covers a broad range of scientific contexts, this broader focus may contribute to a lower perception of science. Students studying a Bachelor of Mathematics or Bachelor of Computer Science (or similar degrees with a significant programming component) are not required to take SCIE1000/1100, which may explain why a similar effect was not resolved in the mathematics or programming OAPs. However, we also acknowledge that factors contributing to negative change in OAP may be evident across the themes of mathematics and programming but masked by positive gains.

**The role of gender**

While the focus of the current study is on the role of an interdisciplinary science course on programming perceptions, we have identified many important variables in our data that are associated with gender. Firstly, women gave significantly lower incoming grade expectations (M = 5.49) than men (M = 5.81) ($p < .001$). Across the semester, overall grade expectations significantly decreased (from M = 5.62 to M = 5.24, $p < .001$). However, we find that women experience a significantly greater decrease in grade expectations (M = -0.44) compared to male students (M = -0.28), p = .006. We also observed that women (M = -0.17) had a significantly more negative change in science OAPs compared to males (M = -0.03), p = .016.

These gender differences may be explained by findings from Grimalt-Álvaro, Couso, Boixadera-Planas, & Godec, (2022), who investigated high school students who reported a positive STEM self-identity (i.e. those who were more likely to identify as a 'STEM person') and found that there was a gendered difference among how this identity is constructed. Defining identity through measures of interest, competence, self-efficacy, and aspiration. They found boys were overrepresented when identity aligned closely with the technology and engineering disciplines within STEM, compared to girls who were overrepresented when identity was aligned with science. Among our SCIE1000/1100 students, it appears reasonable to assume that these students would also see themselves as 'STEM people', but it is possible that the programming component in this course could challenge the identity of students who were more science than technology-aligned, and thus could reflect the observed gender differences in science OAP. We did not observe any gender differences for change in mathematics OAP ($p = .116$), which is also consistent with Grimalt-Álvaro et al (2022), who did not find mathematics to be a primary driver in forming STEM identity.

We also observed that those who reported no prior programming experience were more likely to be women. This is consistent with Australian Year 12 enrolments reported in the 2024 STEM Equity Monitor data summary, which shows that women are underrepresented in Information Technology (IT) subjects, making up just 26% of IT enrolments in 2022 despite equal or greater representation having been achieved in the chemical, natural, and physical sciences. Furthermore, low engagement with technology subjects at the senior secondary level also appears to have been relatively consistent across the past decade (Australian Department of Industry, Science and Resources, 2024). Likewise, at the tertiary level, in 2022, women accounted for only 22% of undergraduate and postgraduate enrolments in information technology, compared to 53% of enrolments in the natural and physical sciences (Australian Department of Industry, Science and Resources, n.d.). Krakowski et al. (2023) warn of the potential for these gendered trends already apparent in IT to spread to other scientific disciplines as technology becomes more ubiquitous in science. To address this inequity, Krakowski et al. (2023) advocate for the integration of computational thinking into science education.

We note that both men ($M = 0.62$) and women ($M = 0.45$) had significant positive changes in OAP for computer programming. The positive change in OAP experienced by women suggests that an interdisciplinary approach to teaching programming, delivered to a broad cohort of science students where women are well represented, could be a valuable strategy for improving equity of access to the programming fundamentals required to succeed in an increasingly computational scientific landscape.

## Limitations

One limitation of our findings was that the post-survey occurred close to or immediately after the Python and Communication Assignment due date, meaning that post-survey responses may have been impacted by the experience of completing a dedicated authentic assessment task that would have seemed impossible before taking the course. While both science and mathematics are required to complete the Python and Communication Assignment, this is not explicitly signposted to the students. Consequently, it is possible that students are less likely to feel 'neutral' about computer programming in the days close to the assignment due date and may underestimate their growth in mathematical and scientific self-efficacy as a result of completing this task.

We acknowledge the inherent challenge of measuring students' attitudes and self-efficacy at multiple time points, particularly since these surveys occurred during in-person Workshops. While our data set is large, it is not necessarily representative of the student cohort. It is possible that selection bias occurred in our sample; students who completed both the pre- and post-surveys must have attended the Workshops in both Week 1 and Week 12/13, meaning that students in our sample were at least reasonably well engaged in the course. Specifically, we found weak evidence that students who attended at the start of semester but did not attend at the end of semester had a higher pre-survey programming OAP (M = 3.54) compared to students who attended at both the beginning and end of the semester (M = 3.46) ($p = .068$). In contrast, there was no difference in pre-survey mathematics OAP between students who were matched (M = 4.58) and unmatched students (M = 4.59) ($p = .936$). We also found that matched students had a significantly higher pre-survey OAP for science (M = 5.44) compared to unmatched students (M = 5.32) ($p < .001$). These findings suggest that students who were in our matched sample had more positive attitudes toward science but were less confident in computer programming. It makes sense that a diligent student who is more enthusiastic about science would recognise the importance of the programming component in SCIE1000/1100, which would motivate Workshop attendance. In contrast, students who lack positive attitudes toward science or have greater confidence in their programming ability may lack the motivation to attend the last Workshop of semester and may be less invested in the general aims of the course.

A further limitation was that our measure of prior programming experience was not explicitly defined, and it was left up to the student to interpret what it meant to have prior programming experience. It is possible that those with low self-confidence may have a higher threshold for what would be considered prior programming experience, resulting in a self-selection bias. To address this, future studies could specify an example of prior programming experience, e.g. independently writing one line of code.

## Conclusions

We find that embedding computer programming within an interdisciplinary science course improves students' overall computer programming attitudes and perceptions, regardless of students' initial level of programming experience. We identify that students' self-reported attitudes toward the Python and Communication Assignment – where students apply programming in a scientific context – most explained changes in programming OAPs, with positive evaluations of the assessment associated with higher improvements in OAP. While we found general improvements in programming OAP, we identified that gender plays a role in students' perceived self-efficacy in a STEM context. Both men and women experienced improved attitudes to programming, and hence we suggest that incorporating programming authentically in a generalist science course may provide an equitable pathway for women, who are otherwise highly underrepresented in the information technology disciplines, to gain exposure to and confidence in programming while simultaneously providing positive outcomes for their male peers. However, care must be taken to ensure that the benefits of this integrated programming experience are not achieved at the cost of diminishing students' enjoyment and appreciation of science. Future studies should aim to explore these gender differences, to gain deeper insight into how interdisciplinary programming courses are experienced and can be further developed to support all students.

## Acknowledgements

The second author was supported by an internal UQ grant and the first author was employed as a research assistant under this grant.

## Ethics statement

Human ethics approval was obtained from University of Queensland Human Research Ethics Committee (#2019000164).

## References

Australian Department of Industry, Science and Resources. (2024). STEM equity monitor: data report 2024, Department of Industry, Science and Resources, Canberra, viewed 08 Nov 2024, https://www.industry.gov.au/publications/stem-equity-monitor.

Australia Department of Industry, Science and Resources. (n.d.). University enrolment and completion in STEM and other fields, viewed 08 Nov 2024, https://www.industry.gov.au/publications/stem-equity-monitor/higher-education-data/university-enrolment-and-completion-stem-and-other-fields .

Brinkworth, R., McCann, B., Matthews, C., & Nordström, K. (2009). First year expectations and experiences: student and teacher perspectives. *Higher Education*, *58*(2), 157-173. https://doi.org/10.1007/s10734-008-9188-3

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Routledge. https://doi.org/doi.org/10.4324/9780203771587

Crisp, G., Palmer, E., Turnbull, D., Nettelbeck, T., Ward, L., LeCouteur, A., Sarris, A., Strelan, P., & Schneider, L. (2009). First year student expectations: Results from a university-wide student survey. *Journal of University Teaching and Learning Practice*, *6*(1), 16-32. https://doi.org/10.53761/1.6.1.3

Cronbach, L.J. (1951) Coefficient alpha and the internal structure of tests. *Psychometrika*, 16, 297–334. https://doi.org/10.1007/BF02310555

Else-Quest, N. M., Mineo, C. C., & Higgins, A. (2013). Math and Science Attitudes and Achievement at the Intersection of Gender and Ethnicity. *Psychology of Women Quarterly*, *37*(3), 293-309. https://doi.org/doi.org/10.1177/0361684313480694

George, D., & Mallery, P. (2016). 18 Reliability Analysis. In Routledge (Ed.), *IBM SPSS Statistics 23 Step by Step : A Simple Guide and Reference* (14 ed.).

Grimalt-Álvaro, C., Couso, D., Boixadera-Planas, E., & Godec, S. (2022). " I see myself as a STEM person": Exploring high school students' self-identification with STEM. *Journal of Research in Science Teaching*, 59(5), 720–745. https://doi.org/10.1002/tea.21742.

Han, J., Kelley, T. & Knowles, J.G. (2021) Factors Influencing Student STEM Learning: Self-Efficacy and Outcome Expectancy, 21st Century Skills, and Career Awareness, *Journal for STEM Education Research* 4:117–137. https://doi.org/10.1007/s41979-021-00053-3

Jenson, J., & Droumeva, M. (2016). Exploring Media Literacy and Computational Thinking: A Game Maker Curriculum Study. *The Electronic Journal of e-Learning*, *14*(2), 111-121.

Krakowski, A., Greenwald, E., Roman, N., Morales, C., & Loper, S. (2023). Computational Thinking for Science: Positioning coding as a tool for doing science. *Journal of Research in Science Teaching*, *61*(7), 1574-1608. https://doi.org/10.1002/tea.21907

Mao, P., Cai, Z., He, J., Chen, X., & Fan, X. (2021). The Relationship Between Attitude Toward Science and Academic Achievement in Science: A Three-Level Meta-Analysis. *Frontiers in Psychology*, *12*, 784068. https://doi.org/10.3389/fpsyg.2021.784068

Matthews, K. E., Adams, P., & Goos, M. (2009). Putting it into perspective: mathematics in the undergraduate science curriculum. *International Journal of Mathematical Education in Science and Technology*, *40*(7), 891-902. https://doi.org/10.1080/00207390903199244

Matthews, K. E., Adams, P., & Goos, M. (2010). Using the principles of BIO2010 to develop an introductory, interdisciplinary course for biology students. *CBE Life Sciences Education*, *9*(3), 290-297. https://doi.org/10.1187/cbe.10-03-0034

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2019) A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Transactions on Education*, Vol.62 (2), p.77-90.

Parsons, J. E., Adler, T., & Meece, J. L. (1984). Sex differences in achievement: A test of alternate theories. *Journal of Personality and Social Psychology*, *46*(1), 26-43. https://doi.org/10.1037/0022-3514.46.1.26

Piggott, A., Herke, S., McIntyre, T. J., & Bulmer, M. (2019). The effect of an interdisciplinary science course on student perceptions of computer programming. *Proceedings of the Australian Conference on Science and Mathematics Education*, 162-168.

R Core Team. (2021). *R: A Language and Environment for Statistical Computing*. In R Foundation for Statistical Computing. https://www.R-project.org/

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13:2, 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Teague, D. (2011) Pedagogy of introductory computer programming: A people-first approach, M.S. thesis, Dept. Inf. Syst., QUT, Brisbane, QLD, Australia.

Vihavainen, A., Airaksinen, J., & Watson, C. (2014) A systematic review of approaches for teaching introductory programming and their influence on success. *ICER 2014 - Proceedings of the 10th Annual International Conference on International Computing Education Research*, p.19-26.

Wiebe, E., Unfried, A. & Faber, M. (2018) The Relationship of STEM Attitudes and Career Interest. *Eurasia Journal of Mathematics, Science and Technology Education*, Vol.14 (10), p.1-17. https://doi.org/10.29333/ejmste/92286