Space Mechanics: A Beginner's Guide to Programming Fundamentals

Anh Tang
Department of Information Technology, Georgia Gwinnett College
United States
atang1@ggc.edu

Julissa Valdez-Ramos
Department of Information Technology, Georgia Gwinnett College
United States
jvaldezramos@ggc.edu

Anca Doloc-Mihu
Department of Information Technology, Georgia Gwinnett College
United States
adolocmihu@ggc.edu

Cindy Robertson
Department of Information Technology, Georgia Gwinnett College
United States
crobertson2@ggc.edu

Abstract

This project strives to foster genuine interest in technology and computing in K-12 and non-IT undergraduate students. Using a simple space-themed game as the teaching platform, we aim to introduce students to fundamental programming concepts such as variables, conditional statements, and loops, as well as algorithmic thinking in a hands-on and engaging way.

CCS Concepts

• **Applied computing** → **Interactive learning environments**.

Keywords

Scratch, Makey Makey, project-based learning, teaching, block-based programming, algorithm, critical thinking, sprite animation, CS, IT, education, gamification, outreach

ACM Reference Format:
Anh Tang, Julissa Valdez-Ramos, Anca Doloc-Mihu, and Cindy Robertson. 2024. Space Mechanics: A Beginner's Guide to Programming Fundamentals. In *The 25th Annual Conference on Information Technology Education (SIGITE '24), October 10–12, 2024, El Paso, TX, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3686852.3687068

1 Introduction

The Technology Ambassador Program (TAP) at Georgia Gwinnett College (GGC) is a service-learning course dedicated to boosting student interest and participation in computing through numerous outreach activities led by students at GGC. TAP faculty and students strive to increase curiosity and enthusiasm in technology through interactive classroom workshops in introductory Information Technology (IT) courses on campus and through extracurricular activities for K-12 students, such as the Atlanta Science Festival.

SIGITE '24, October 10–12, 2024, El Paso, TX, USA

ACM ISBN 979-8-4007-1106-0/24/10
https://doi.org/10.1145/3686852.3687068

We developed our project while enrolled in the TAP course. It was built on results presented in a prior publication [2] focused on the effect of our service-learning courses for underrepresented minority students and how these courses encouraged them to explore more opportunities to engage in STEM.

The objective of this project is to introduce the concept of algorithms and to explain how to develop an algorithm using variables, conditional statements, and loops. It is geared toward an audience that has little to no prior programming experience such as K-12 and non-IT undergraduate students. We accomplish our goal through a hands-on, beginner-friendly platform: *Space Mechanic*, a space-themed role-playing game designed to turn the learning process into something engaging and entertaining. Through this project, we hope that we are able to give the students a crash course in programming and inspire them to seek out more opportunities to participate in technology-related activities in the future. Most importantly, we hope to demonstrate to the students themselves that they are more than capable of learning new technical skills.

The following sections include an introduction to our project and the results of our study designed to gauge student interest in IT after participating in a classroom workshop. Discussion and Future Work sections will conclude this work.

2 Methodology

Our project aims to give a crash course on programming concepts such as variables, conditional statements, and loops, along with basic algorithmic thinking. For teaching programming, we used Scratch [4]. To make the project more fun and interactive, we incorporated a Makey-Makey [5] into the game-play.

2.1 Block-Based Programming via Scratch

Scratch [4] is a block-based programming language created by the MIT Media Lab. We chose Scratch for our project because of its beginner-friendly features and colorful user interfaces that turn the learning process into an experience similar to playing a video game. For students who are not familiar with programming, a block-based language may seem less daunting and intimidating than a text-based language (see Figure 1). Since Scratch has pre-made blocks of code that the students can drag and drop and then organize to

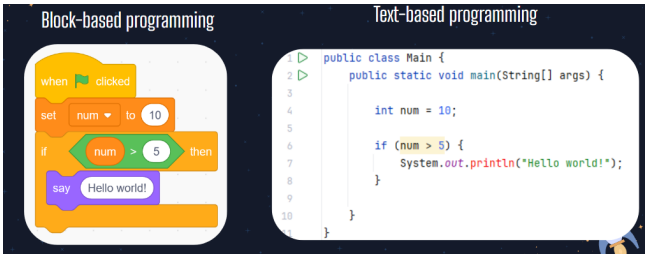

Figure 1: Example of simple code showing a text-based program and its corresponding block-based counterpart.

build an algorithm, they can focus on learning the concepts without being overwhelmed by the syntax.

2.2 Boosting Engagement through Makey-Makey

Makey Makey [5] is an invention kit designed by the MIT Media Lab that turns everyday objects like bananas, bread, and Play-Doh into a controller as long as they are conductive. Students can use these conductive items in place of the arrow keys on the keyboard. It is a fun and simple way to demonstrate the flexibility and creativity of technology.

2.3 Integrating Asset Packets and Creating Game Graphics

The game's visuals were created using an asset pack called *Tech Dungeon: Rougelite* purchased from an indie developer, Trevor Pupkin [7]. This asset pack can be used in both free and commercial projects and all credits belong to its creator Trevor Pupkin [7]. The asset pack consists of a 32x32 tile set for level designs, a character sprite with multiple idle/walking animations, and sprite sheets for props such as treasure chests, computer monitors, workstations, etc.

Scratch does not have a feature for integrating asset packs or sprite sheets. Therefore, we created several game levels externally and then imported them into the game on different layers to translate depth and allow for collision.

3 Our Space Mechanic Game Project

For our project, we created a game called *Space Mechanic* in Scratch where the player role-plays as a mechanic working on a spaceship. The player can move around using the arrow keys or the Makey Makey. The game also includes wall collision, which means the character won't be able to defy physics and walk through walls or obstacles in the way.

Before playing the game the participants are taught fundamental programming concepts such as variables, conditional statements, and loops. Then, the game quizzes students on these concepts. The spaceship has two rooms as shown in Figure 2. While the player can freely move between the rooms at any time, we suggest they finish the first room and then go to the second room. The player must observe their surroundings and identify problems hidden within the sprites.

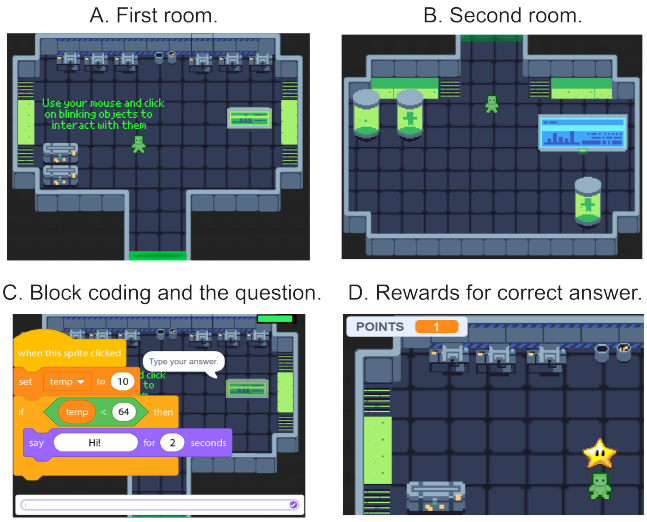

Figure 2: Spaces Mechanic game developed in Scratch. (A) Shows the first room where the player answers coding questions. (B) Shows the second room where the player animates sprites via simple code. (C) Shows an in-game programming question and its block coding. (D) Shows the point accumulated and the star animation for a correct answer.

First room, Figure 2A: The player, role-playing as the mechanic of the spaceship, is tasked with fixing the broken computer monitor and opening treasure chests. There are some chores too, like taking out the trash for example. The first room is designed to quiz the player on their understanding of the programming concepts (variables and conditional statements) they learned during the first part of our workshop prior to playing the game. There are several blinking objects in this room and each of them is a task that the mechanic must complete to earn points. Every time the player clicks on a blinking object, a programming question will appear on the screen (see Figure 2C). Players need to read the question and type the answer in the dialog box at the bottom of the screen. A correct answer to the question will earn the player a point; an animated star appears signaling the point gained (see Figure 2D). There are four points in total and the amount of attempts for each question is infinite.

Second room, Figure 2B: The objective of this room is to quiz the player on a deeper understanding of how to apply their newly acquired knowledge of loop statements to animate the sprites. Some of the sprites in this room are not animated. The player needs to spot them, then go inside the code and program animation for these sprites.

Our project is freely available on the TAP website at https://tapggc.org//. We presented our project at several outreach events, including classroom workshops, which we describe next.

4 Outreach Workshops

Our hour-long workshop comprises of three learning sections:

(1) A lesson on character movement in Scratch where students learn about concepts such as variables, conditional statements, and loops. They also play with the sprite movements and are encouraged to experiment with other ways of programming character movement.
(2) A lesson on sprite animation followed by hands-on practice time during which students create their own sprite animations.
(3) A session where students enter *Space Mechanic* and answer programming questions that help solidify the concepts they previously learned. In addition to that, they are also tasked with debugging sprite animations in the second room of the game. To complete this task, students must be able to read the code and identify the problem, which encourages their critical thinking and problem-solving skills.

During the workshop, we provided hand-outs to students, which included a brief introduction to every concept that we were going to cover and a PDF file with step-by-step instructions for all workshop activities. We also administered pre and post-surveys at the beginning and the end, respectively, of our classroom workshops. The next section provides our analysis of the results obtained from these pre and post-surveys.

5 Results

We conducted a total of four classroom workshops in our Introduction to Computing general education course at Georgia Gwinnett College. We analyzed the data obtained from students who participated in the workshop activities to assess the results. We had a total of 56 workshop participants; however, only 37 of them responded to both pre and post-surveys. Therefore, we will include only the results obtained from these 37 participants in these results.

5.1 Demographics

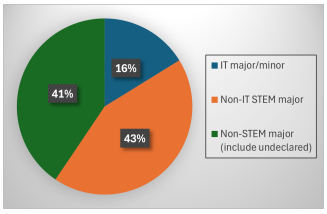

Figure 3: Majors of the workshop participants.

67.6% of the workshop participants were female (25) and 32.4% were male (12). As shown in Figure 3, the workshop participants were mostly non-IT STEM majors (43% or 16) and non-STEM majors (41% or 15). We had only 6 IT majors and minors participating in our workshops (16%).

Figure 4 shows that majority (62%) of our workshop participants did not have any prior programming experience, 27% considered themselves a novice and only 11% chose the intermediate option. No participants picked the advanced option.

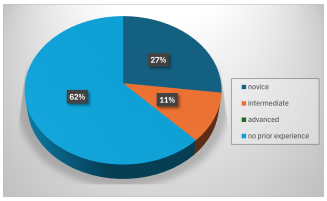

Figure 4: Workshop participants' experience in programming.

We wanted to find out how effective the classroom workshops were at teaching the fundamental programming concepts, so we quizzed the participants on different types of questions, including general and programming questions.

5.2 Assessing General Concepts

Figure 5 shows the results from the pre and post-surveys for the multiple choice questions *What is block-coding?* in panel A and *What is an algorithm?* question in panel B. Figure 5A shows a significant difference between the number of participants that were able to answer correctly the *What is block-coding?* question before and after our workshop, with a increase from 27% in the pre-survey to 91.9% in the post-survey.

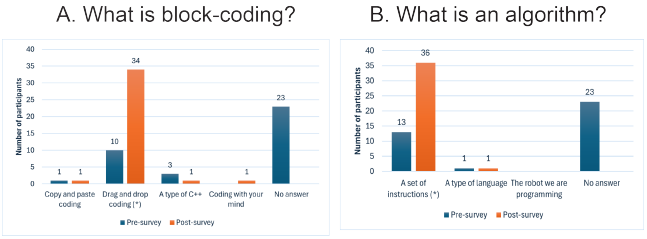

Figure 5: Pre and post-survey results for general concepts questions. (*) indicates the correct answer. (A) What is block coding? and (B) What is an algorithm?

We used the χ^2 test [1] to analyze the difference between our control and experimental data results to see if there was any significant difference between these group results. To test the independence of the two survey results, pre and post, we used the chi-square test of independence. The p-values were obtained using *scipy.stats.chi2_contingency* function in SciPy library [10] in Jupyther Colab. We obtained $\chi^2 = 38.09$, $df = 4$, and a p-value $p = 1.073198591987177e-07$ which is less than ($p < 0.05$) showing that there was a significant difference between the answers of the pre-survey versus the answers of the post-surveys.

Similarly, the results shown in Figure 5B for the multiple choice question *What is an algorithm?* reflect a noticeable improvement as well, increasing from 35.1% of participants answering correctly in the pre-survey to a whopping 97.3% in the post-survey. Since the chi-square test relies on comparing observed frequencies to expected frequencies and having zero in the expected frequencies can lead to division by zero errors or inaccurate results, we added a small constant of 0.5 to all of our pre and post-data to avoid

divisions by zero. We obtained $\chi^2 = 32.62$, $df = 3$, and a p-value $p = 3.8702317919715245e-07$ which is less than ($p < 0.05$) showing that there was a significant difference between the answers of the pre-survey versus the answers of the post-surveys.

5.3 Assessing Programming Concepts

Figure 6 shows the results from the pre and post-surveys for the three programming questions modeled after their in-game counterparts. In these questions, students are expected to state the output of a given program using previously learned concepts such as variables and conditional statements.

According to the results in Figure 6, students demonstrated a significant improvement on all programming questions. Figure 6B shows that 97.3% of the participants were able to answer the first programming question (Figure 6A) correctly in the post-survey as opposed to the previous 35.1% in the pre-survey. We obtained $\chi^2 = 32.62$, $df = 5$, and a p-value $p = 4.473726525527954e-06$ which is less than ($p < 0.05$) showing that there was a significant difference between the answers of the pre-survey versus the answers of the post-surveys. Note that to avoid division by zero, we adjusted the survey values obtained for this question by 0.5.

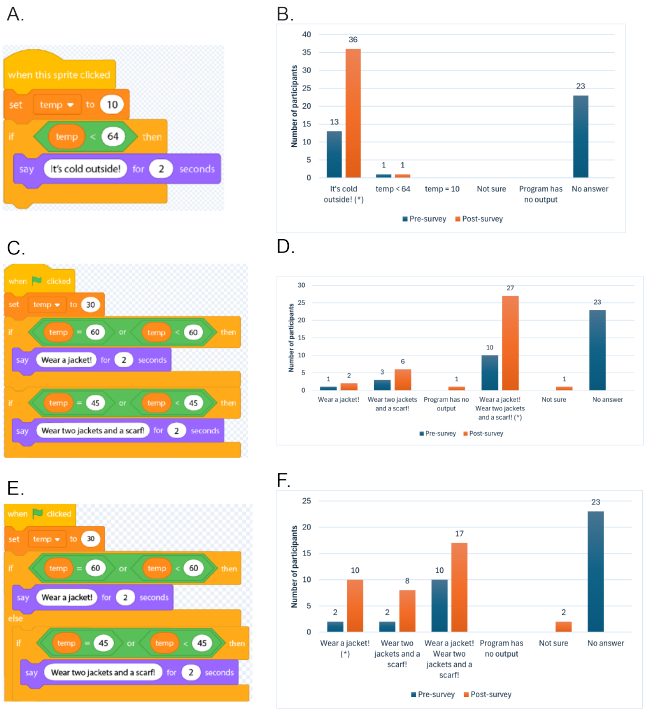

Figure 6: Three coding questions asking the output of the block-code shown in the left panels (A, C, and E). On the right we present the pre and post-survey results for each of the block-code questions. (*) indicates the correct answer.

Figure 6D indicates a 45.9% increase in the number of participants answering the second question (shown in Figure 6C) correctly. For this question, we also obtained $\chi^2 = 34.14$, $df = 5$, and a p-value $p = 2.2287377590205416e-06$ which is less than ($p < 0.05$) showing that there was a significant difference between the answers of the pre- and post-surveys.

A similar trend is evident in Figure 6F as well, where the percentage goes from 5.4% in the pre-survey to 27% in the post-survey. We obtained $\chi^2 = 33.32$, $df = 5$, and a p-value $p = 3.2498014687655327e-06$ which is less than ($p < 0.05$) also showing a significant difference between the answers of the pre-survey versus the answers of the post-surveys. Note that, as above, to avoid division by zero, we adjusted the survey values obtained for this question by 0.5.

However, it is important to note that in the third programming question (Figure 6E), the most popular answer choice is not in fact the correct answer. This indicates that a large percentage of students were confused about the difference between two if-statements and an if-else statement. This information will help us make some necessary adjustments for future workshops.

5.4 Workshops Effectiveness

In our post-surveys, we also asked our students several questions about their thoughts on the workshops. On a scale of 1 to 5, students gave us a score of 4.35 regarding how fun the workshop was and how easy it was to follow the instructions. On a scale of 1 to 10, students gave us a score of 2.89 out of 10 in terms of workshop difficulty (with 10 being the most difficult) and an 8.92 regarding how enthusiastic the presenters were.

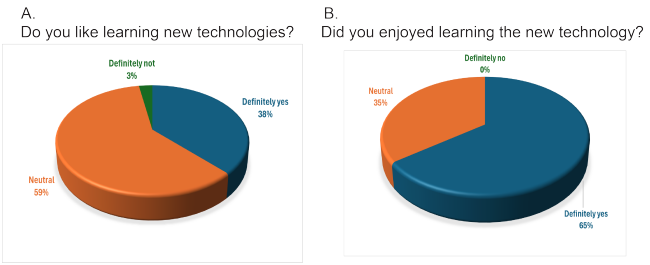

Figure 7: Participants' interest in learning new technology. (A) Pre-survey. (B) Post-survey answers about the technology learned during the workshop.

In our pre-surveys, we asked the participants if they liked learning new technologies. As shown in Figure 7A, 38% of the participants answered yes, 59% were neutral, and 3% answered no. After the workshop, we also asked them if they enjoyed learning the new technology and received the following results depicted in Figure 7B: 64.9% answered yes, 35.1% were neutral, and nobody answered no. This indicates that the workshops were successful in convincing more students to learn new technologies (26.9% shifted their answer to yes after our workshop). Even the participant who answered no in the pre-survey enjoyed learning the new technology via our workshop. We obtained $\chi^2 = 5.94$, $df = 2$, and a p-value $p = 0.05115$ which is slightly higher than ($p < 0.05$) showing that there was not a significant difference between the answers of the pre-survey versus the answers of the post-surveys. However, we consider that even if one participant got to see that learning new technology could be fun after our workshop as a success story. Our results show that several participants were influenced by our workshop, which was a great success for us.

When asked, 33% of the students said the workshop sparked their interest in programming, 51% said maybe, and only 16% gave a definite no (shown in Figure 8).

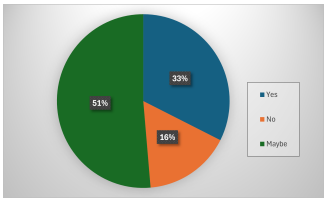

Figure 8: Participants' interest in programming after the workshops.

When asked what was the most difficult item they learned during the workshop, many participants stated "coding", which is a good indicator that coding is actually quite challenging to learn and requires more time. Overall, the participants rated our workshops a score of 4.78 out of 5.

6 Discussion

As stated in the Introduction section, the primary objective of this project was to promote interest in computing and increase participation in IT and STEM fields. According to the data obtained from our surveys, a third of our students said the classroom workshops sparked their interest in programming. Among the students, no one answered negatively. Most workshop participants, however, were not decisive in their opinions, which tells us that we need to put more effort into making the workshops more engaging for this group of participants.

Moreover, initially, in the pre-surveys, only 38% of the participants answered yes to the question *Do you enjoy learning new technologies?*. However, in the post-surveys, there was a significant shift in preference (an approximate 27% increase) and two-thirds of the participants said they enjoyed learning the new technology taught in the workshop in the post-surveys, which demonstrated to us that we are heading in the right direction in our mission to encourage more students to engage in technology-related fields. These results showed us that we should continue our commitment to service learning and outreach work to reach even more students in the future.

The difficulty rating shown in the Workshops Effectiveness subsection indicated that the majority of participants regarded the materials as being too easy. For future workshops, we may consider increasing the difficulty level by adding more complex concepts/questions. However, at the same time, we also noticed that a large percentage of students answered the question shown in Figure 6 E incorrectly, which was most likely due to them not understanding how an if-else statement works. This concept was a crucial part of our workshops. Therefore, we must allocate more time to explain this concept, perhaps with more examples, in future workshops.

The results from the Assessing Programming Concepts subsection also indicate that our methods of teaching variables and if statements were successful, as demonstrated by the large shift in the number of correct answers (see Figures 6B and 6D). However, we are aware that our pool of survey responses is rather small (37 participants who responded to both pre and post-surveys) so we cannot draw any definitive conclusions. Thus, we take these results as suggestions and we would like to hold more workshops to gather more feedback and adjust our materials in the future.

Some of the feedback from the participants about the classroom workshops was:

- "Everything was very fun I would not change anything"
- "Great job lots of fun"
- "Everything was great! Very fun and interesting!"
- "Nothing, you did great :)"
- "Nothing. The way the project was presented was easy for me to understand"
- "The project was pretty fun and interesting !"

All these comments are very encouraging to us. We also received this comment "More of an in-depth explanation on how to work scratch", which shows us that we need to make some adjustments on the way we introduced the Scratch workspace and features to the students.

From chatting with the students immediately after the workshops, we learned that for many of the students, their favorite part of the workshops was drawing their own sprite and programming its animation. They loved seeing their work in action and being creative with technology. To assess the learning of this lesson, we asked the students a general multiple-choice question about the type of loop they used to build their animation, and we gave four choices to select from (results not shown). 84% of the participants chose the correct answer, demonstrating that perhaps a better way to teach programming is to allow students to create their own visual and then ask them to program it to do some action.

Our study here was built upon previous work [6, 8, 9]. However, unlike [6] which taught only block coding via Scratch, we incorporated two technologies into our workshops. Thus, our participants had less time to learn the coding concepts. We think they might have needed more time to thoroughly understand the coding concepts. Similar to [6], we asked simple block-coding questions in both pre and post-surveys to assess the difference in learning created by our workshops. While this might not be the only factor, it definitely shows that some level of learning happens during the course of the workshop. We believe we will need to conduct more in-depth studies to assess what other factors play a role in influencing the process of learning programming concepts.

7 Conclusion and Future Work

The method of using a hands-on, interactive platform like a video game to teach programming fundamentals is efficient and effective. Despite the majority of the participants having little to no coding experience, the students in introductory IT courses were able to complete all workshop activities and in-game programming challenges with ease after just learning these concepts only a few minutes earlier. This shows that our outreach workshops are effective and useful tools for not only teaching basic programming skills but also enticing students to change their views on programming and IT in general.

Recently, we presented our project at the Teachers Technology Workshop in front of teachers from several local elementary, middle,

and high schools. We learned from them what technologies they are allowed to use in the classroom. For example, they are not allowed to use Scratch in the classroom, but instead they use a version of the Scratch editor made specifically for K-12 students called CSFirst Scratch [3], which allows students to work on the game without needing to leave the CS First website. In the future, we plan to adapt our game and workshop to CS First as well as other requirements the teachers mentioned during the workshop and give them full access to our work.

Acknowledgments

Our team wants to express our most profound gratefulness toward the IT Department Chair and the Dean at Georgia Gwinnett College for supporting our project. Sponsored via NSF 2315804.

References

[1] William G Cochran. 1952. The χ^2 Test of Goodness of Fit. *The Annals of Mathematical Statistics* 23, 3 (1952), 315 – 345. https://doi.org/10.1214/aoms/1177729380
[2] Sonal Dekhane et al. 2018. *Journal of Computing Sciences in Colleges* 34, 2 (2018), 147–153. https://doi.org/10.5555/3282588.3282609
[3] MIT. 2020. *CSFirst.* https://scratch.mit.edu/users/CSFirst/
[4] MIT. 2020. *Scratch.* https://scratch.mit.edu/
[5] MIT. 2024. *Makey Makey.* https://makeymakey.com/
[6] Valentina Mosquera-Reina, Ryan Cunico, Josiah Williams, Matthew Bauer, Anca Doloc-Mihu, and Cindy Robertson. 2021. Introducing Programming Concepts through Interactive Online Workshops. In *SIGITE 2021*. ACM, 71–72. https://doi.org/10.1145/3450329.3478319
[7] Trevor Pupkin. 2024. *Tech Dungeon: Roguelite - Asset Pack by Pupkin.* https://trevor-pupkin.itch.io/tech-dungeon-roguelite
[8] Cindy Robertson and Anca Doloc-Mihu. 2021. Assessing the Effectiveness of Teaching Programming Concepts through Online Interactive Outreach Workshops. In *SIGITE 2021*. ACM, 123–128. https://doi.org/10.1145/3450329.3476861
[9] Cindy Robertson and Anca Doloc-Mihu. 2023. Understanding College Level Student Learning of Basic Programming at an Open Access Institution. In *Proceedings of the 2023 ACM Southeast Conference*. ACM, 26–32. https://doi.org/10.1145/3564746.3587007
[10] SciPy.org. [n. d.]. *SciPy.* https://projects.scipy.org/getting-started.html