I. J. Education and Management Engineering, **2024, 5, 46-62**
Published Online on October 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijeme.2024.05.05

Comparative Analysis of Two Programming Platforms for Beginners: Python and Scratch

Aderonke Busayo Sakpere*
Department of Computer Science, University of Ibadan, Ibadan, Nigeria
Email: ab.sakpere@ui.edu.ng
ORCID iD: https://orcid.org/0000-0001-8260-1229
*Corresponding Author

Adedeji Folashade
Department of Computer Science, Lead City University, Ibadan, Nigeria
Email: adedeji.folasade@lcu.edu.ng
ORCID iD: https://orcid.org/0000-0002-6105-4139

Received: 26 February, 2024; Revised: 21 March, 2024; Accepted: 10 April, 2024; Published: 08 October, 2024

Abstract: This research paper conducts a comparative analysis of Python and Scratch, exploring their strengths and weaknesses in introductory programming education. While Scratch serves as an excellent starting point, it has limitations, prompting discussions about its suitability for all learners. Some argue that starting with Scratch facilitates a smoother transition into Python, while others suggest its effectiveness in attracting beginners to computer science. The study, conducted over 12 weeks among beginners in a Nigerian higher institution, aims to assess factors such as ease of learning, versatility, community support, and real-world application on both platforms. The first 4 weeks, participants were introduced to Scratch, then were introduced to Python from week 5 to 8 and finally week 9 to 12 were to work on projects and compare both platforms.

The research delves into the experiences of participants lacking prior programming experience, emphasizing the exploration of thematic analysis, System Usability Scale (SUS) scores and individual responses. A total of four evaluations were carried out. Results from the thematic analysis of the 1st evaluation using thematic analysis reveals that Scratch has the ability to foster computational thinking. The 2nd evaluation reveals that Scratch is preferred for tasks such as game development which has the ability to further deepen their programming experience. In the third evaluation, 46.3% of the participants agreed that experience gained from Scratch was helpful in learning Python while 70% agreed to some or a great extent that knowledge and skills acquired from learning Scratch was transferable to learning Python. The fourth evaluation was to understand the ease of use of Scratch versus Python using SUS. The results from SUS notably reveal that the limited number of female participants showed intriguing preferences, with a lone female participant indicating a higher preference for Scratch. However, examining individual responses revealed a consistent outlier, with all participants expressing a higher preference for using Python more frequently than Scratch, despite their initial exposure to both platforms. This research suggests that the choice between Python and Scratch goes beyond syntax preferences, involving pedagogical strategies and the learning experiences each platform offers.

This research contributes insights into the effectiveness of Scratch and Python in an educational setting, offering a nuanced understanding of the preferences and experiences of beginners. The findings underscore the importance of considering not only platform features but also individual learning experiences and pedagogical strategies in shaping programming education for novices.

Index Terms: Introductory Programming Education, Scratch, Python, Comparative Analysis, Learning Experiences, Computational thinking, Thematic Analysis, Sysyem Usability Scale

1. Introduction

Ongoing research endeavors persistently seek to enhance the support provided to beginners in their journey to learn programming, as documented by scholars such as Robins [1], Sakpere [2], Eteng et al. [3], and Skaarseth [4]. Amidst the plethora of studies, two prominent platforms, Scratch and Python, consistently emerge as noteworthy champions in facilitating the learning experience for novices, as indicated by the works of Balreira [5], Sakpere [6], Kelly et al. [7], Elhaid et. al. [8] and Federici [9].

Python, recognized as a general-purpose programming language renowned for its versatility and widespread applicability has garnered immense popularity, particularly for instructing beginners. The language's ascendancy is attributed to its readability, simplicity, and extensive range of applications, as highlighted by the insights of Muller[10]. The distinctive feature of Python's clear and concise syntax positions it as an optimal choice for novice programmers, a sentiment echoed by Hetland [11]. This readability not only expedites the learning curve but also paves the way for a seamless transition into more intricate programming concepts, fostering an environment that encourages learners to delve into problem-solving and algorithmic thinking, as underscored by Khoirom [12]. In essence, Python's multifaceted appeal positions it as a formidable ally in the quest to facilitate a supportive and constructive programming learning experience for beginners.

Scratch, a visual programming language developed by MIT Media Lab was targeted primarily at young learners and beginners [13]. Scratch uses a drag-and-drop interface with colorful code blocks, eliminating the need to memorize syntax. This visual approach aims to make coding more accessible and enjoyable, fostering creativity and problem-solving skills [14-15]. Even though Scratch was primarily designed to be used in outside-of-classroom settings especially among the underprivileged [16,17] it has since evolved and been used as a primary language for teaching beginners in University and formal setting [18,19].

In this research paper, we endeavor to conduct a comparative analysis, delving into the strengths and weaknesses of Python and Scratch as platforms for introductory programming education. Previous studies have highlighted that while Scratch serves as an excellent starting point for beginners, it possesses certain limitations that render it suboptimal for all learners [9]. Conversely, some researchers argue that commencing with Scratch facilitates a smoother understanding and transition into Python [20,21]. Furthermore, Scratch has been recognized as a platform capable of attracting beginners, particularly in initiating an interest in Computer Science, when compared to Python [21]. Consequently, our objective in this research is to explore factors such as ease of learning, versatility, community support, and real-world application both platforms offer. In addition, we also carried out a comparative study from the angle of inclusiveness and accessibility for students in developing world and finally we seek to understand how these platforms foster computational thinking. These factors aim to assist aspiring programmers in making informed decisions based on their goals and preferences. We achieved this by conducting a study for 12 weeks amongst beginners or first year students of Computer Science in a higher institution in Nigeria. Scratch was firstly introduced for a period of 4 weeks and afterwards, Python was subsequently introduced for another 4 weeks. In the last month, students had to further work on projects that require the use of both Scratch and Python. As we navigate through the features of each platform, it becomes apparent that the choice between Python and Scratch transcends mere syntax preferences, encompassing pedagogical strategies and learning experiences offered by each platform.

2. Literature Review

In today's world, computer programming has become an essential skill in many fields [6]. Whether it's creating software, analyzing data, or developing websites, programming serves as the foundation for these tasks. Programming is defined as the process of writing instructions that can be executed by a computer to perform specific tasks, it is also a means of creating something new in the digital world, solving problems and implementing ideas [22]. It started with simple languages like Fortran and Basic, and now there are more advanced languages like C++, Java, and [23]. A programming platform is like a toolbox for programmers, containing tools and resources to build, test, and run computer programs in specific languages. It has everything they need, like components, interfaces, and libraries. These platforms often include development tools to make it easier for developers to create applications [24]. There are different programming platforms to support programmers, offering a variety of programming languages. Each language has its own way of writing code and special features to meet different needs and complexities. These platforms range from text-based IDEs for experienced programmers to block-based interfaces designed for beginners. Each platform has its unique features and tools.

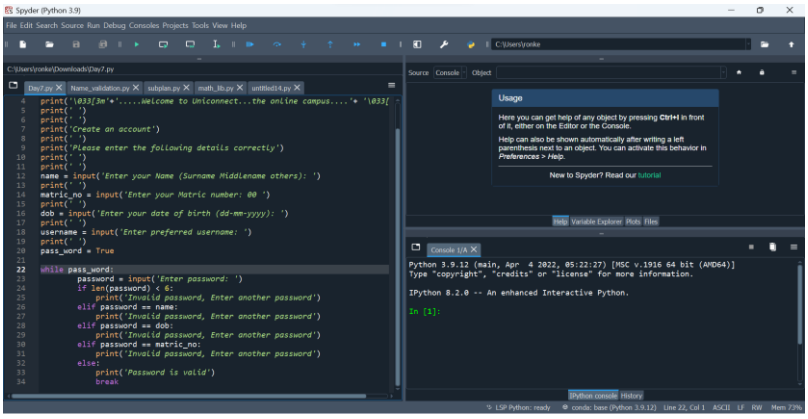


Fig. 1. A typical Python Coding Environment

Learning programming can be a painful process, especially for those seeking to master text-based programming languages such as Python [25]. Fig. 1 shows an illustration of a typical Python Coding Environment. However, there are alternatives available, such as visual programming languages like Scratch, which aim to make the learning process more accessible and enjoyable for beginners. Fig. 2 shows an illustration of a typical Scratch Coding Environemnt. Scratch and Python are two popular programming languages that have been used to support beginners in learning programming Sakpere [6].

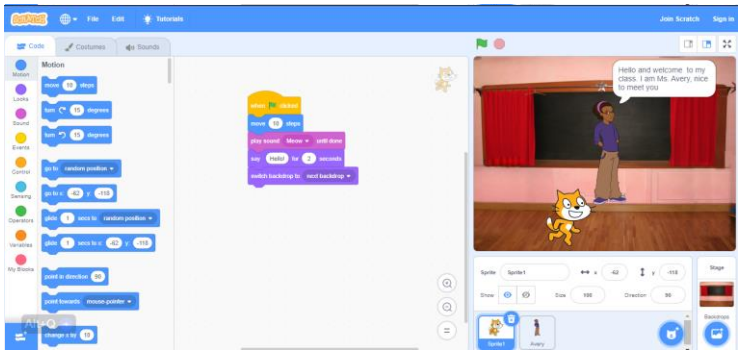


Fig. 2. A Typical Scrtach Coding Environment/IDE

Scratch provides a block-based interface where users can drag and drop blocks of code to create programs. Python, on the other hand, is a high-level, general-purpose programming language that emphasizes readability and simplicity. Both Scratch and Python have their own unique design principles and theoretical foundations. Scratch was explicitly developed for teaching programming and aims to lower the complexity associated with general-purpose programming languages like C, C++, and Java. In terms of research, both Scratch and Python have been extensively studied and utilized as programming platforms for beginners. Studies have shown that Scratch and Python can effectively support beginners in learning programming concepts and developing computational thinking skills. For example, the ScratchJr environment has been shown to enhance problem-solving skills and promote early childhood education [26]. Additionally, research has highlighted the effectiveness of Scratch in promoting creativity and collaboration among students [27]. Oladele et al [28] suggest that employing Scratch in higher education can be engaging and useful, especially for students with no prior programming experience.

On the other hand, Python has been praised for its versatility and widespread use in industry, making it a valuable programming language for students to learn. Several studies have examined the success and failure of using Scratch and Python as programming environments for beginners. Some studies have reported positive outcomes, showing that Scratch and Python can effectively support beginners in learning programming concepts and improving their computational thinking skills [6, 29]. Other studies, however, have identified challenges and limitations in using Scratch and Python. These challenges include difficulties in transitioning from block-based programming such as Scratch to text-based programming such as Python [30], as well as the need for additional scaffolding and support to facilitate the learning process.

In the existing literature, some gaps in research need to be addressed regarding the use of Scratch and Python as programming environments for beginners. For instance, there is a need for more studies that specifically compare the learning outcomes and effectiveness of Scratch and Python in supporting beginners to learn programming. Additionally, more research is needed to explore the potential benefits and challenges of integrating Scratch and Python in a complementary manner. In this study, we conducted a comparative analysis, delving into the strengths and weaknesses of Python and Scratch as platforms for introductory programming education.

3. Methodology

To achieve the objective of this research, participants consisted of first-year students at a Nigerian University who enrolled in an introductory course in computing for the 2023 academic session. A total of 67 students enrolled for the course. Prior to the start of the course, an open call and recruitment was made for students to fill in their prior programming experience. Participation was voluntary. A total of 57 students (80.8%) took part in the pre-assessment survey. Table 1 shows the demographic information of those who participated.

Table 1. Demographics of Participants

Characteristics	Frequency
Gender	
Male	7
Female	48
Took Computing in High School	
Yes	35
No	20
Prior Programming Experience	
No Experience	21
Yes (Experience)	34 (8 in Python or other HLL) (26 in at least HTML)

Given the high number of the participants who either have no prior programming experience or have ever taken computing as a subject in high school, the researchers decided to explore the use of Scratch and Python as a programming platform for beginners or those new to programming. This decision was based on the recommendation of these platforms as ideal for beginners in previous research [6,31]. The lecture period lasted for a period of 11 weeks. The first class was a general introduction to computing, explaining various fields of Computer Science and Career options available. The hands-on started using Scratch started in the 2nd week and lasted till the 6th week. Python was introduced in the 7th week up to the 9th week. The 10th and 11th weeks were used for working on the final project and covering computing subjects such as networking and Computer & Society.

The pedagogy adopted in the course was project-based and collaborative. Using project-based pedagogy, students were able to work on mini projects and the collaborative pedagogy enabled them to work in a team. In addition, the project helped them develop soft skills such as communication, writing, and presentation. Projects centered on games, animations, ATM simulation, implementation of algorithms such as Zeller Algorithm used for Day Prediction and various computations such as password validation for social media.

Finally, we carried out evaluation using qualitative and quantitative means. The qualitative is based mainly on asking participants to reflect on their experience and quantitatively focused on the usability of the platforms.

4. Analysis, Evaluation and Reflection

- Qualitative Analysis: A reflection on the assignment that they did using both Scratch and Python to design a game and calculator.
- A total of 4 Evaluations were carried out during the course. The first evaluation was based on their positive and negative experience using Scratch to work on their first project.

4.1 First Evaluation: Thematic Analysis on Scratch Experience

The first evaluation was in their 2nd week of using Scratch and their 1st team-based project was a 3 in 1 animation project with a storyline that focused on their school, discipline, and the team/buddy. A total of 30 projects were received with most of them as group/team work except for 2-3 submissions for those who opted out to be assigned to work in a team. A team consists of 2 people typically with similar computing or programming experience. In addition, 5 senior students signed up to provide support as teaching assistants. Fig. 3 is a sample of animation created for the 1st project. To evaluate their learning experience, students were asked to reflect and outline the joy and frustrations experienced programming with Scratch on their first project. Thematic Analysis was used to classify their responses. In total, 5 themes were identified. Table 1 and 2 present these themes, further explanation about them and the number of occurrences (frequency).

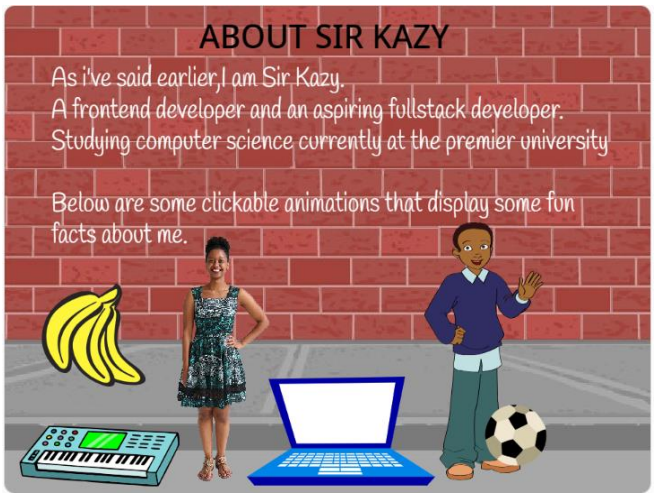


Fig. 3. Sample of Project 1 (Animation on Buddy Introduction)

Table 2. Thematic Analysis of the Joy Experience of using Scratch

Theme	**Explanation and Extracts/Quotes from Responses**	**Frequency (Number of People that expressed theme)**
Creativity, & improvement	The participants attested that working on an animation project using Scratch was useful to sharpen their creativity and generally improve with respect to their computing and problem solving skills. Some responses on this, "**The Scratch programming language forced us to be creative in our thinking and application of the blocks.**" It also helped them make new discoveries of how to program or make an object interactive. Some responses on this, "**We were happy to discover we had inbuilt talents in coding."** It is also worth noting that even though it enhanced their creativity, a lot of mental work was involved. Some responses verbatim, **"The experience was hectic but educative", "The creative aspects were mentally exhausting".**	13
Bonding/Team-Spirit Enabled	Team work on a Scratch project provides a platform for bonding and even brings about joy, closeness to each other as a team, Fun, Brilliant ideas, New soft skills. "**It helps to think creatively, work collaboratively, and reason systematically**."	9
Simplicity & Learnability	The simplicity of the platform enabled easy navigation, spotting of errors. It also improves learning and promotes better understanding of algorithms, problem solving and coding. Some responses **"Building with a scratch compared with using a programming/markup language is lot easier and faster." "We were able to understand algorithms better."**	8
Sense of achievement	Scratch was able to help them bring their ideas to life and this gave them a sense of achievement. **"The thrilling feel of exploring different aspects of the scratch interface."**	6
Good Features & General good user experience	The help features helped students to recover from errors or get insights into problems or issues experienced. "**The user interface was good and the user experience was great and made the work easier."** "**It was a great experience.I was happy when I started making progress."**	4

Table 3. Thematic Analysis of the Frustration (Negative) Experience of using Scratch

Theme	**Explanation and Extracts/Quotes from Responses**	**Frequency (Number of People that expressed them)**
Rigidity	Participants find that Scratch is not flexible essentially because blocks are pre coded. The rigidity also extends to the number of sprites available to choose from and also the fact that some events could require too long lengthy codes. Also, participants realized Scratch is not great when it comes to correcting mistakes. Some comments **"Making a single mistake made the whole animation to malfunction."**	10

Limitation in Resources	The participants are students who live in Africa, precisely Nigeria and are faced with challenges such as irregular electricity, unstable or limited internet access, state of the art computer laboratory or equipment etc. This seems to have affected many of them from using Scratch optimally. Some of them had to use their mobile phones and this didn't give them a good experience since Scratch is designed to work mainly on desktop or tablet. Some comments: **"Network connection was the major issue faced. It set us back a couple times."; "neither of us currently has a fully functional laptop." "Internet glitches sometimes caused a loss of uploaded backdrops and coding. It consumes much data." "it was kinda stressful because I was using my phone to do it."**	8
Frustration as a result of first time experience	Many of the participants were using Scratch for the first time. So they found it a bit challenging learning to use the platform and also working on 3 different animation projects. Lack of exposure. Some comments, "**Since it was basically my first project using scratch it took a while to get used to the software**"."**Due to limited experience in working with Scratch, it wasn't easy coming with project ideas.**"	5
Compatibility Issues & Extra Features Required	Some participants found out that unavailability of extra features on the Scratch didn't make working or collaboration easy. their browser wasn't compatible with Scratch and so had to Some comments **"It was difficult working on the same project without a group account that was accessible to both of us at the same time." "Had issues with accessing the project on my scratch account, but was able to access it eventually with another browser.". "The text-to-speech function was very limiting so I had to use tools outside scratch to generate audio."**	3

4.2 Second Evaluation: Further Analysis on Scratch for Game Development

From literature, Scratch Gaming has been identified as instrumental to helping learners develop computational thinking [32]. Computational thinking is a 21st century skill needed for problem solving at all stages and ages of life [33]. As a result, in this study, development of game was integrated into the projects participant were asked to work on. Though none of the team had prior experience in game programming using Scratch, 5 out of the 30 teams indicated they have used other platforms such as Javascript, Python, ConstructD and StuckD to develop games. So in a way, all teams were using Scratch for the first time to develop games. The project was to create 3 different games using Scratch. Prior to this task, videos of 5 different games and tutorials created using Scratch were shared with the participants. After the task, learners were asked to test their games with users and get feedback. In addition, they were asked to share new skills and experiences gained using Scratch to work on their second project.

4.2.1 User feedback

The essence of this evaluation was to get feedback from users' perspective and not necessarily the programmer about how applications developed using Scratch feels in terms of aesthetics, ease of use and general user experience. According to Riihiaho [34], usability evaluation is key to user-centered product development. Previous researches have also reported the immense benefits of usability evaluation [35]. As a result, the participants carried out usability evaluation on their games. Each team tested their games with different numbers of users. The team with the minimum number of users testing their games had 2 participants while the team with the highest number of users testing their games had over 10 users. On average, each game was tested by 5 users. Fig. 4 shows the total number of users who tested the game. The total number of games developed is 90 as each team had 3 games and the total number of teams was 30.

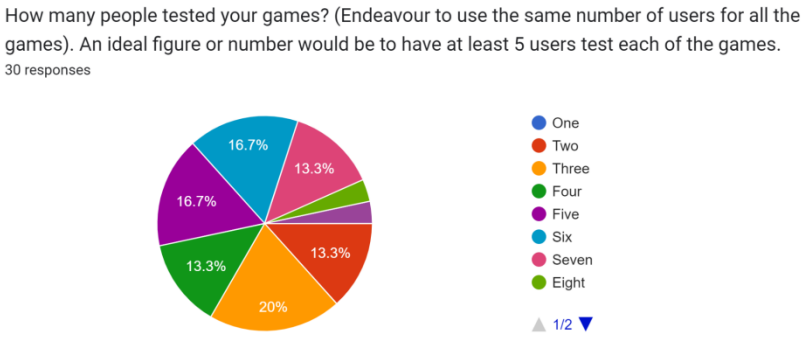


Fig. 4. Distribution of the number of users who tested the application

Observation technique has been identified as a valid tool for usability evaluation [36] and this was used to get users feedback during the usability study. The users are mainly students of a university in the South West of Nigeria and their discipline varies from Tech to Science to Humanities. The responses fall in the category of positive and negative. Positive responses revolve around themes such as interesting, amazing and overall good. Negative responses revolve around the game having a kid look and may not necessarily be attributed to Scratch, the programming platform used to

create the code. In summary, we say that Scratch has the ability to produce games that are interesting, amazing and overall good.

4.2.2 New experience or skills gained

Using thematic analysis we grouped participants' responses on the new skills and experience gained into themes. A total of 4 themes emerged:

(1) Enhanced Programming Proficiency: The most prevalent observation among our surveyed teams, with at least 15 affirmations, was the significant improvement in programming skills resulting from engaging in advanced projects, particularly game development. Teams reported a clearer grasp of fundamental concepts such as control structures, abstraction, code organization, and debugging. For instance, Team 8 noted, "Through this project, I comprehended the mechanics of loops and conditions, executing them proficiently with Scratch." Meanwhile, Team 12 highlighted their acquisition of practical knowledge in event-driven programming, along with a deeper understanding of conditionals and loops. The project emphasized the importance of well-organized and readable code, fostering a more nuanced comprehension of overarching programming principles. In summary, the project provided valuable insights into problem-solving, critical thinking, and honed programming abilities for the teams involved.

(2) Problem Solving Skills: This category was noted by 14 teams, indicating a notable improvement in general problem-solving skills such as logical thinking and creativity due to project involvement. Several teams provided insights into their experiences, highlighting phrases like, - "I gained the ability to improve my thinking skills, " It also helped us develop our critical thinking skills by teaching us how to break down complex problems into smaller, more manageable pieces. And it was really fun to create something that other people could play!" "Creating these games had really improved our skills, and ability to kind of " think outside the box" It was really amazing." Working on this game project significantly improved my problem-solving and critical thinking skills.

(3) User Experience Skills: This topic arose frequently, mentioned a total of 6 times. When crafting an application, prioritizing user-centric design is paramount (Sakpere, 2015). Many teams found that creating a gaming application intended for user testing significantly enhanced their User Experience skills. They gained a deeper understanding of how to captivate users and tailor designs to their needs. Feedback included statements such as: ""The concept of the games did not seem difficult; however, we realized the importance of user experience in video games". "This project helped improve my analytical thinking and allowed me to view a game/program from the users perspective". It required me to analyze and resolve challenges related to game mechanics and user experience."

(4) Unlocking Hidden Potentials: This topic emerged frequently, with a total of 4 mentions. Engaging in a challenging project such as developing three gaming applications per team proved to unlock hidden potentials among participants. They discovered newfound self-reliance by utilizing the internet as a supplementary learning resource, tapping into their subconscious to sustain focus on the project, and even experiencing improvements in other academic areas like mathematics and physics. Paraphrasing participants response, "This project consumes your thoughts, akin to the excitement of a new relationship or an addictive game. It's a constant presence in your mind, driving subconscious ideation. I found myself brainstorming while cooking or tackling math problems, a testament to the project's influence on my subconscious thinking.". Another shared, "Partnering on this project allowed us to expand our knowledge base, leveraging the internet as a secondary educator for programming skills.". Furthermore, participants noted that the project deepened their comprehension of mathematical and physical concepts through the necessity of complex calculations to instill realism in the games.

4.3 Third Evaluation: Scratch Experience/Skill Transferable to Python?

3 weeks after exposing the learners to programming in Scratch. They were then exposed to programming using Python for another 3 weeks. In week 6 of their programming learning, they were given 2 tasks: coding and designing a calculator and game in both Python and Scratch. Then they were asked to reflect on their journey for 6 weeks and also evaluate their experience on both platforms. The first interest was to understand if their knowledge and experience in Scratch was useful and transferable to Python. Fig. 5 and table 3 have their responses. The other interest was also to understand which of the platforms helped them learn programming better and the response to this is illustrated in table 4.

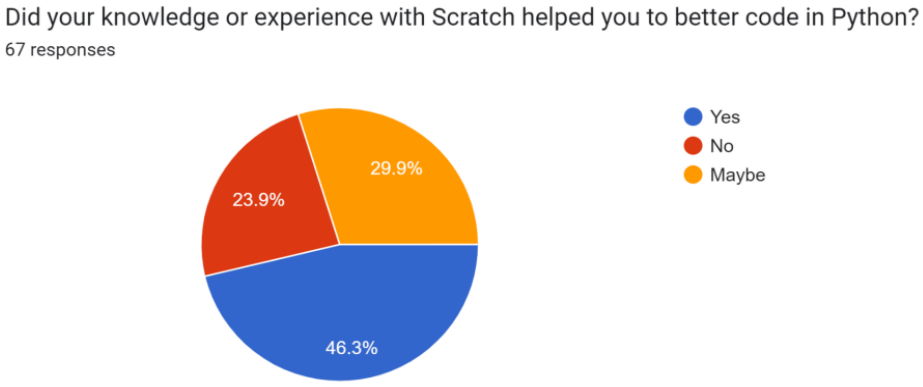


Fig. 5. Responses on how experience in Scratch helped to better code in Python

Table 3. Responses on the transferability of skills from Scratch to improve coding in Python.

Response	Count
A ittle bit (kind of/relatable)	17
No	13
Yes (to some extent or greatly)	36

Table 4. Which of the platforms helped them to learn programming better

Platform	Responses
Scratch	18
Python	22
Both (Python & Scratch)	3
None (Neither Python nor Scratch)	4

4.4 Fourth Evaluation: Python vs Scratch: SUS

Usability refers to how specific users can effectively, efficiently, and satisfactorily use a system, product, or service to achieve intended goals within a given context [37]. To assess the perceived usability of Python and Scratch as programming platforms, students were surveyed regarding their experiences. Google Forms facilitated the collection of responses. The survey utilized the System Usability Scale (SUS), designed by John Brooke for subjective system usability testing [38]. Originally developed as part of the usability engineering program at Digital Equipment Co Ltd., Reading, United Kingdom, SUS comprises ten items providing a holistic perspective on users' subjective assessments of usability.

Following the students' interaction with Python and Scratch, the questionnaire was administered. The ten questionnaire items included five positive opinions (1, 3, 5, 7, and 9) and five negative opinions (2, 4, 6, 8, and 10). Each opinion was rated on a 5-point Likert scale, where 1 indicated strong disagreement and 5 indicated strong agreement. Ten items featured in the SUS questionnaire. The maximum score is 100, and the average established is 68 as a minimum for considering a system as usable [38]. The total score is obtained based on a specific procedure. In the case of odd items the score is calculated subtracting 1 from the obtained value in each item; in the case of the even items, the score is calculated by subtracting a 5 from the value obtained in the item. Finally, the sum of these 10 scores is multiplied by 2.5, thus obtaining the overall usability score [39].

For this study, an electronic version of the questionnaire was created using Google Forms, distributed with the provided URL: https://forms.gle/fbKgixDUGCwTZuZ56. Fig. 6 displays a screenshot of the online questionnaire, adapted from the SUS. At the beginning of the questionnaire some general background questions were included, such as gender and participants' prior experience with programming.

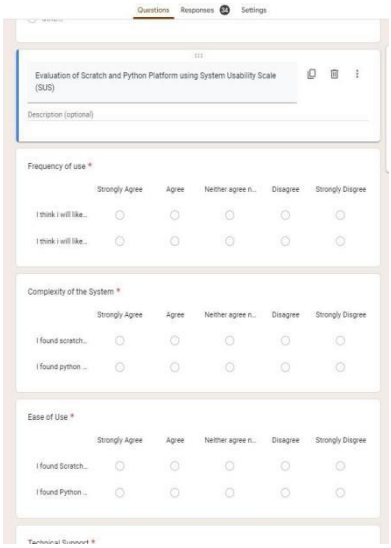


Fig. 6. SUS questionnaire adopted for evaluating Scratch and Python platforms

For a reliable SUS result, a minimum of 8-12 participants are required [40]. 34 participants from the students that took the course interacted with Python and Scratch, the questionnaire was administered. Based on the result of the study, the distribution by gender where 24 out of the 34 responded to the question on gender. Out of the 24 that responded, there were 20 males (83.3%), 2 females (8.3%); 2 respondents (8.4%) preferred not to say their gender. As shown in Fig. 7, 79.4% had prior programming experience, and in Fig. 8, 29.4% had prior experience with python, while 11.8 had prior experience with scratch. As shown in Fig.9 , 38.2% had up to 6 months of programming experience prior to the course, 14.7% had between 13 to 18 months of prior programming experience.

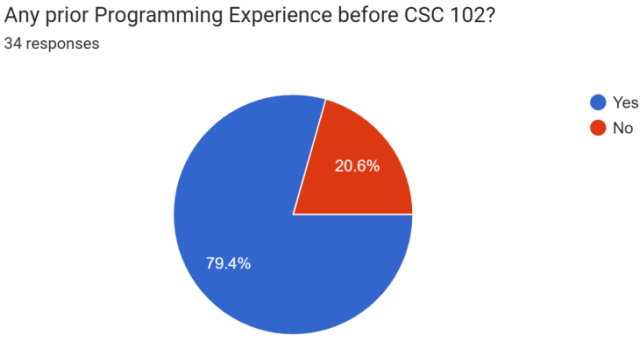


Fig. 7. Participants that had prior programming experience

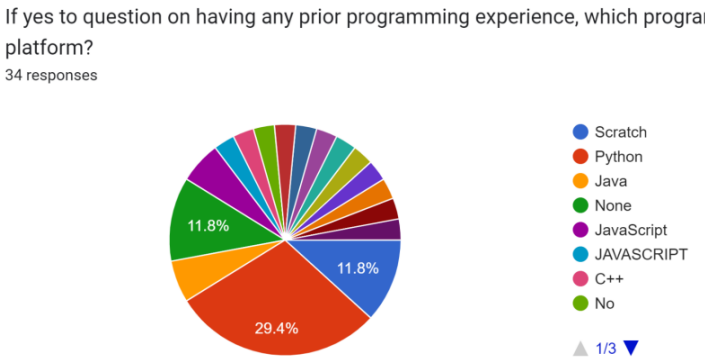


Fig. 8. Participants previous programming platform

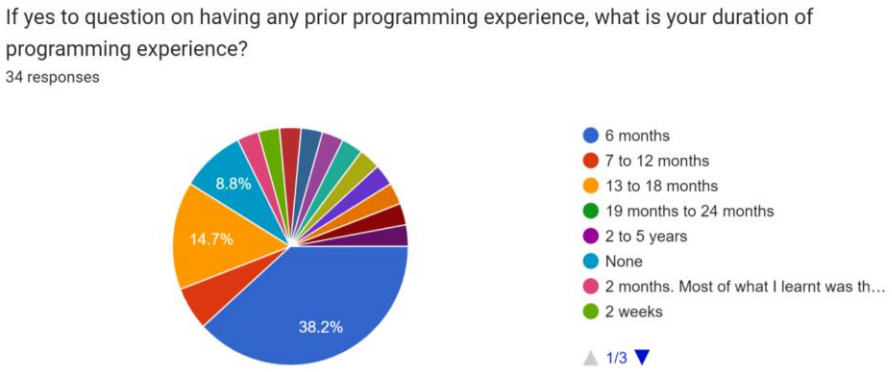


Fig. 9. Duration of programming experience prior to taking the course

The SUS scores for Scratch and Python are 55.07 and 60.5, respectively, positioning both scores below the average SUS score of 68, according to the SUS interpretation scale by Lewis and Sauro [41]. The standard deviation of 14.2 indicates a significant dispersion in the obtained data. In the case of Scratch, participant scores ranged from 17.5 to 90 points, with 16 responses surpassing the average SUS score of 55.07 and 18 responses falling below it. For Python, scores ranged from 17.5 to 82.5, with 18 responses exceeding the 60.5 average and 16 responses falling below it.

Although SUS scores are represented on a 100-point scale, it is essential to emphasize that they are not percentages and should not be construed as such. Lewis and Sauro[41] recommend expressing SUS scores in percentiles for more precise communication. In this study, the average SUS scores for Scratch (55.07) and Python (60.5) categorize them in grade D, indicating that the perceived usability of Scratch and Python as learning platforms for programming beginners is considered satisfactory. The SUS scale comprises ten items, each contributing to the evaluation of distinct usability aspects. These items can be construed to represent various facets of usability. Notably, the standard process does not involve calculating scores for individual SUS questions. Inspired by GitLab's approach to gauging long-term usability and Lewis and Sauro's item benchmark methodology [41,42], where benchmarks were derived from an overall SUS score, we adopt a similar strategy. This approach provides additional insights into specific usability aspects of Scratch and Python, facilitating a comparison of individual SUS questionnaire items for both languages. Our individual question scores align with the scale used for the overall SUS score, enabling a nuanced understanding of how each question performs in the context of Scratch and Python. The process involves obtaining the normalized single question score and multiplying it by 25 to derive the final score.

Table 5 presents a breakdown of SUS scores by item, offering a detailed comparison between Scratch and Python for each aspect. Participants express a clear preference for using Python frequently, with a score of 86 compared to Scratch's 45. Python is perceived as less complex than Scratch. Despite being more user-friendly, Python requires support from a technical person for effective use. Python's functions are more seamlessly integrated, though there is noted inconsistency. Scratch excels in learnability, while it is considered more cumbersome. Users express greater confidence in using Scratch, although it entails learning more before getting started.

Table 5. SUS Score by individual Item

S/N	SUS Questions	Scratch SUS Item Score	Python's SUS item Score
1	I think I will like to use Scratch/Python frequently	45.0	86.0
2	I found Scratch/Python unnecessarily complex	42.75	67
3	I found Scratch/Python easy to use	60.0	65.4
4	I think i will need the support of a technical person to be able to use Scratch/Python	56.0	39.0
5	I found various functions in Scratch/Python well integrated	60.0	68.3
6	I thought there were too much inconsistency in Scratch/Python	57.0	66.2
7	I would imagine that most people will learn to use this Scratch/Python very quickly	74.26	61.7
8	I found Scratch/Python very cumbersome to use	43.0	57.0
9	I felt very confident using Scratch/Python	57.3	60.3
10	I needed to learn a lot of things before I could get going with Scratch/Python	49.0	39.0

5. Discussion

Firstly, we seek to understand the effect of lack of programming experience (prior to taking the introduction to programming course) on Python and Scratch. We explored the SUS scores and individual responses to the 10 SUS questions among participants lacking prior programming experience, which accounted for five (5) out of the thirty-four (34) total participants. Referencing Table 6, among the 5 participants with no prior programming experience, one identified as female, 3 as male, and one preferred not to disclose their gender. Notably, the lone female participant achieved a significantly higher SUS score (70) in Scratch compared to her score (32.5) in Python. However, due to the limited number of female participants, we cannot definitively conclude that females with no programming experience prefer Scratch. Additionally, upon examining individual responses to the 10 SUS questions, a consistent outlier emerged. All 5 participants expressed a higher preference for using Python more frequently than Scratch, particularly evident in their responses to the frequency of use question.

Table 6. Participants without prior programming experience

Participant's S/No.	GEDERS	SUS SCORE		Like to use Frequently		Complex systems		Easy to use		Needed Tech Support		Well Integrated		Inconsistent		Learned to use quickly		Cumbersome		Confident using		Needed to learn a lot to use	
		Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch	Prthon	Scratch
8	Nil	50	55	A	N	N	N	N	N	A	D	A	D	N	N	N	A	A	N	N	N	N	D
13	Male	72.5	55	A	D	D	D	A	N	A	N	A	D	SD	D	A	A	D	A	A	A	D	D
19	Female	32.5	70	A	N	SD	N	D	A	N	D	D	SA	N	D	N	SA	SA	N	SD	A	SA	N
22	Male	52.5	20	SA	D	D	SA	N	SD	SA	A	A	N	N	N	A	D	D	A	D	SD	SA	SA
24	Male	70	50	A	N	N	N	A	N	A	A	A	N	SD	A	A	A	D	N	A	N	D	D

SA=Strongly Agree A=Agree N=Neither Agree Nor Disagree D=Disagree SD= Strongly Disagree

Secondly, we decided to understand the effect of programming experience (prior to taking the introduction to programming course) on Python and Scratch. As referenced in Table 7, we examined the SUS scores of 29 participants who possessed prior programming experience before enrolling in the introduction to programming course. Among them, 5 had experience with Scratch, 7 with HTML/CSS, 11 with Python, 3 with Java, and 2 each with JavaScript and C++/C. There were no notable differences in scores regardless of the programming language participants had learned before the course, except for those with previous experience in JavaScript. Participants with prior JavaScript experience demonstrated significantly higher SUS scores for Python. Figure 10 in the appendix shows the different rating of participants comparing both Python and Scratch with respect to different usability factors such as complexity, ease of use, learnability, consistency amongst others. This figure informed how the SUS values generated on Table 5, 6 and 7 were generated.

Table 7. Participants with prior programming experience

Scratch				Prthon				Java				HTML/CSS				JavaScript				C++			
S/No.	Gender	Prthon	Scratch	S/No.	Gender	Prthon	Scratch	S/No.	Gender	Prthon	Scratch	S/No.	Gender	Prthon	Scratch	S/No.	Gender	Prthon	Scratch	S/No.	Gender	Prthon	Scratch
1	Nil	17.5	50	4	Nil	60	72.5	6	Nil	72.5	72.5	2	Nil	77.5	65	12	Male	72.5	40	3	Nil	57.5	75
17	Male	47.5	32.5	5	Nil	77.5	35	10	Nil	72.5	55	20	Male	42.5	52.5	29	Nil	70	25	31	Male	50	37.5
19	Female	40	90	6	Nil	72.5	27.5	30	Male	55	72.5	25	Male	60	57.5								
23	Male	77.5	67.5	9	Nil	72.5	17.5					26	Male	65	50								
				11	Male	70	62.5					27	Male	55	65								
				14	Male	62.5	55					32	Male	42.5	80								
				15	Male	65	47.5					33	Female	52.5	40								
				16	Male	75	57.5																
				21	Nil	62.5	70																
				28	Male	65	72.5																
				34	Male	60	80																

Thirdly, we try to understand how this study fills gaps in current research, particularly in developing countries. From table 2 and 3, various thematic analyses were generated to show the positive and negative experiences of the participants who are from Nigeria, which is one of the developing countries. The positive experiences reveal that the use of Scratch fosters creativity and team-spirit. Creative Learning Spiral, which is the foundation or the theory behind the design of Scratch attributes to this positive experience [43]. Through a creative learning spiral, participants are able to go through the cycle of imagination of ideas, creation: bringing their ideas into life, Play: constant experimenting and

tinkering with their ideas, Sharing: Collaboration with others through direct project or gaining inspiration from existing projects, Reflection: evaluation their ideas for possibility of improvement and finally go back to the beginning of cycle which is imagination and this spiral is repeated. Challenges prominent in the developing world such as lack of resources, unstable internet access, load shedding or regular power outage [44, 45] also affected the learning experience of the students negatively as reflected in Table 3. Of utmost concern is the fact that many of the learners in the developing world own mobile phones and necessarily do not own a desktop or tablet which Scratch is developed or designed for [2,6]. A recommendation from this study would be that a mobile version of Scratch be designed to foster continuity and learning of students in the developing world.

5.1. Novel methodologies or findings that distinctly contribute to existing literature on programming education

Our study offers insights into the landscape of introductory programming education. By emphasizing the need for tailored approaches and considering the evolving nature of programming education, our findings can inform educators, curriculum developers, and learners in creating more effective and inclusive learning environments. The findings of our research contribute to the existing literature on programming education in several ways,

(a) Comparative Analysis of Python and Scratch: Our study provides a comprehensive comparative analysis of Python and Scratch as platforms for introductory programming education. By exploring the strengths and weaknesses of each platform, we contribute insights that go beyond mere syntax preferences.
(b) Insights for Aspiring Programmers: The insights generated from our research offer guidance for aspiring programmers in making informed decisions based on their goals and preferences. Our study empowers learners to choose the platform that best suits their needs, by acknowledging the benefits and limitations of both Scratch and Python
(c) Pedagogical Approach: Our research highlights the effectiveness of a pedagogical approach focused on project-based and collaborative learning. Engaging students in mini projects spanning diverse areas, we facilitate teamwork and soft skills development while providing a well-rounded exposure to programming concepts.
(d) Consideration of Individual Learning Experiences: Our research emphasizes the importance of considering individual learning experiences and pedagogical strategies in choosing between Python and Scratch. This approach recognizes the diverse needs and preferences of learners, contributing to a more inclusive learning environment.

5.2. Broader Implications for Programming Education

Research has also shown that Scratch is under-utilized or not fully explored in higher education across all disciplines despite its enormous benefits[46]. Typically, the higher curriculum of a Nigerian hgher education for a 1st year student often uses text-based programming such as Python to teach programming. Howbeit, it has not recorded a huge success rate or retention. In this study, findings shows that initially exposing first time learners to Scratch which is a visual platform offers a transferrable skill to a text-based platform such as Python and could be potential for higher retention of students in Computer Science and endear them to programming. Furthermore, findings from our research show that Python offers a mobile IDE which enables learner who do not have access to a desktop or laptop which is typically the situation in developing world to effectively gain hand-ons experience unlike Scratch that is fully optimized for desktop and not necessarily for mobile phones.

Broader implications for programming education beyond the study's context could include

(a) Informing educational policies and curriculum development strategies to cater to diverse learner needs and preferences. This could also influence pedagogical approaches in other educational settings and disciplines, highlighting the importance of tailored instruction and project-based learning methodologies. In addition, the findings may contribute to the ongoing discourse on the integration of technology in education and the promotion of computational thinking skills across various learning domains.
(b) Diversify Programming Platforms: Encourage the adoption of diverse programming platforms, including visual platforms like Scratch, alongside text-based languages such as Python in higher education curriculum. Recognize the potential of Scratch to serve as a bridge for first-time learners, offering transferable skills that can enhance their understanding and retention of programming concepts.
(c) Address Accessibility Challenges: Recognize the limitations faced by students in developing countries, particularly regarding access to desktop or laptop computers. Explore solutions to make programming education more accessible, such as leveraging Python's mobile IDE or adapting Scratch for mobile use. Ensure that educational materials and resources are optimized for various devices to accommodate diverse learning environments.
(d) Holistic Approach to Programming Education: Take a holistic approach to programming education by integrating both visual and text-based platforms into the curriculum. Emphasize the complementary nature of these platforms, highlighting their unique strengths and capabilities. Provide students with opportunities to explore different programming paradigms and tools to broaden their skill sets and deepen their understanding of computer science concepts.

(e) Promote Hands-On Learning: Encourage hands-on learning experiences that allow students to actively engage with programming tasks and projects. Provide access to resources and tools that facilitate practical experimentation and exploration, regardless of the devices available to students. Emphasize the importance of experiential learning in reinforcing theoretical knowledge and developing problem-solving skills.

(f) Support Continuous Professional Development: Invest in professional development opportunities for educators to enhance their proficiency in teaching diverse programming platforms effectively. Offer training programs, workshops, and resources that enable instructors to stay updated on the latest pedagogical approaches, technologies, and best practices in programming education. Foster a culture of lifelong learning among educators to ensure the ongoing improvement of teaching practices and student outcomes.

By implementing these recommendations, higher education institutions in Nigeria and other developing countries can create more inclusive and effective programming education programs that cater to the needs and challenges of diverse student populations.

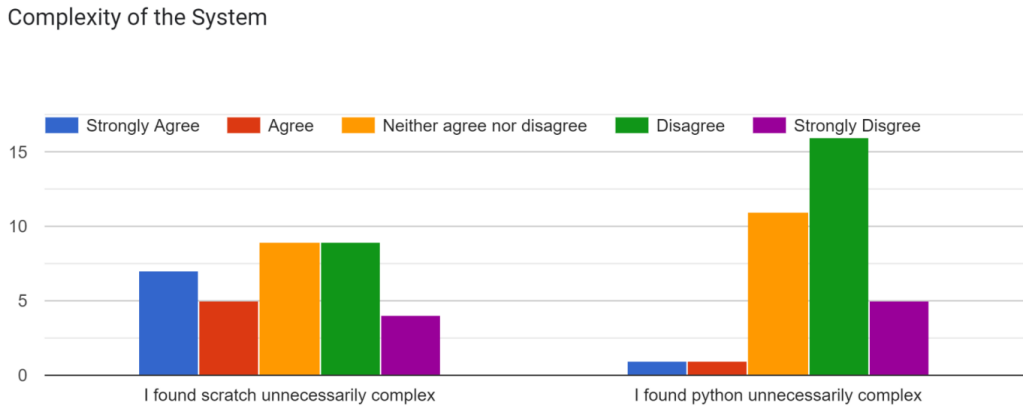
5.3. Conclusion and Further Work

In conclusion, our research delved into the comparative analysis of Python and Scratch as platforms for introductory programming education, exploring their strengths and weaknesses. The study, conducted over 12 weeks with first-year students at a Nigerian University aimed to provide insights for aspiring programmers in making informed decisions based on their goals and preferences. While Scratch was recognized as an excellent starting point, particularly for attracting beginners and fostering interest in Computer Science, it was acknowledged to have limitations that rendered it suboptimal for all learners. On the other hand, Python's ascendancy as a general-purpose programming language, renowned for its versatility and readability, positioned it as an optimal choice for novice programmers. The pedagogical approach adopted, focusing on project-based and collaborative learning, allowed students to work on mini projects, fostering teamwork and soft skills development. Projects spanned diverse areas such as games, animations, simulations, and algorithm implementations, providing a well-rounded exposure to programming concepts.

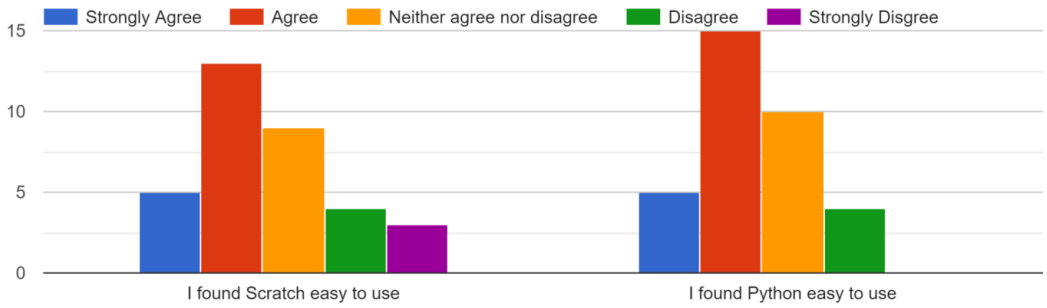
Qualitative analysis, including reflections on assignments involving both Scratch and Python, provided valuable insights into the students' experiences. Four evaluations conducted throughout the course offered continuous feedback on their progress and highlighted areas of improvement. The System Usability Scale (SUS) scores and individual responses among participants lacking prior programming experience revealed intriguing patterns. While the lone female participant showed a higher preference for Scratch, the overall consistency among participants favored using Python more frequently.

Ultimately, our research suggests that the choice between Python and Scratch extends beyond mere syntax preferences. It involves considering pedagogical strategies, individual learning experiences, and the specific goals of the aspiring programmers. The study contributes nuanced insights into the dynamic landscape of introductory programming education, emphasizing the need for tailored approaches to accommodate the diverse needs and preferences of learners. As programming education continues to evolve, these findings can inform educators, curriculum developers, and learners in creating a more inclusive and effective learning environment.

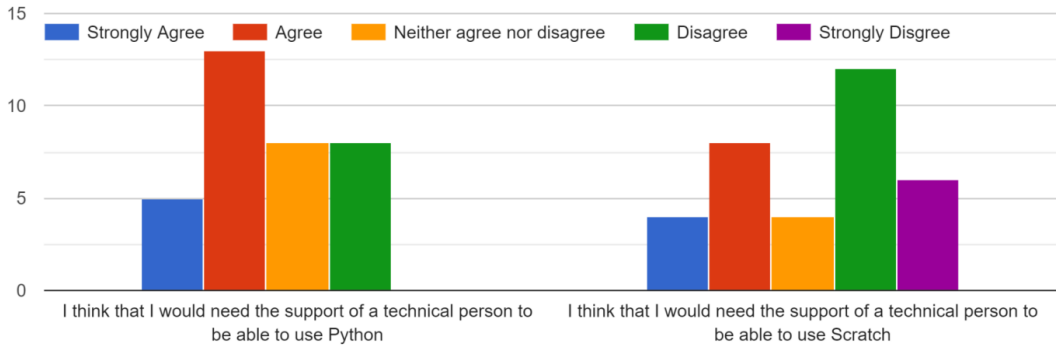
Appendix



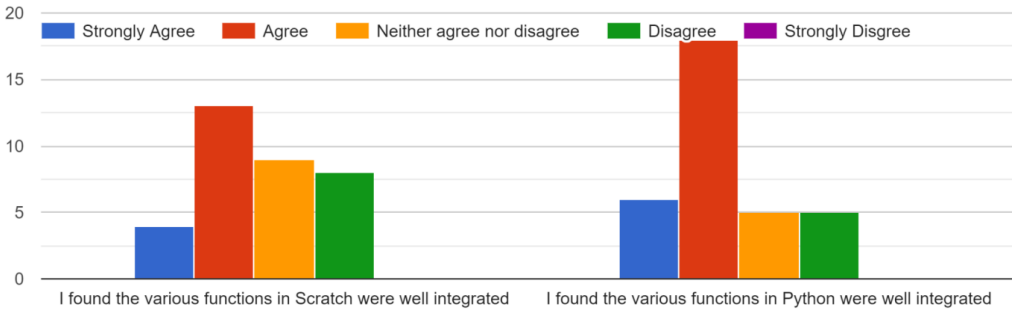
Ease of Use



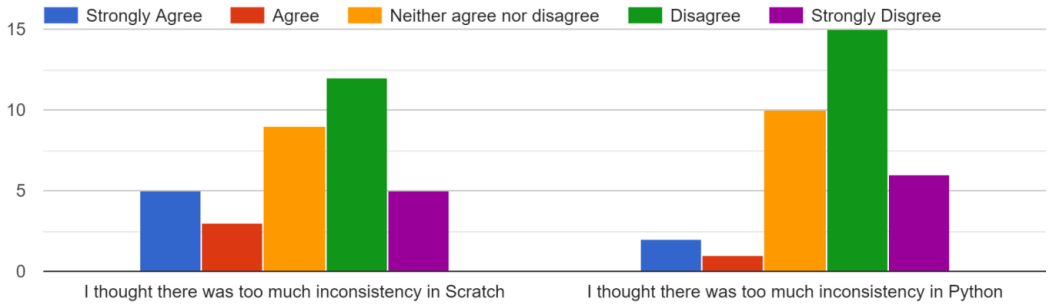
Technical Support



Integration



Consistency



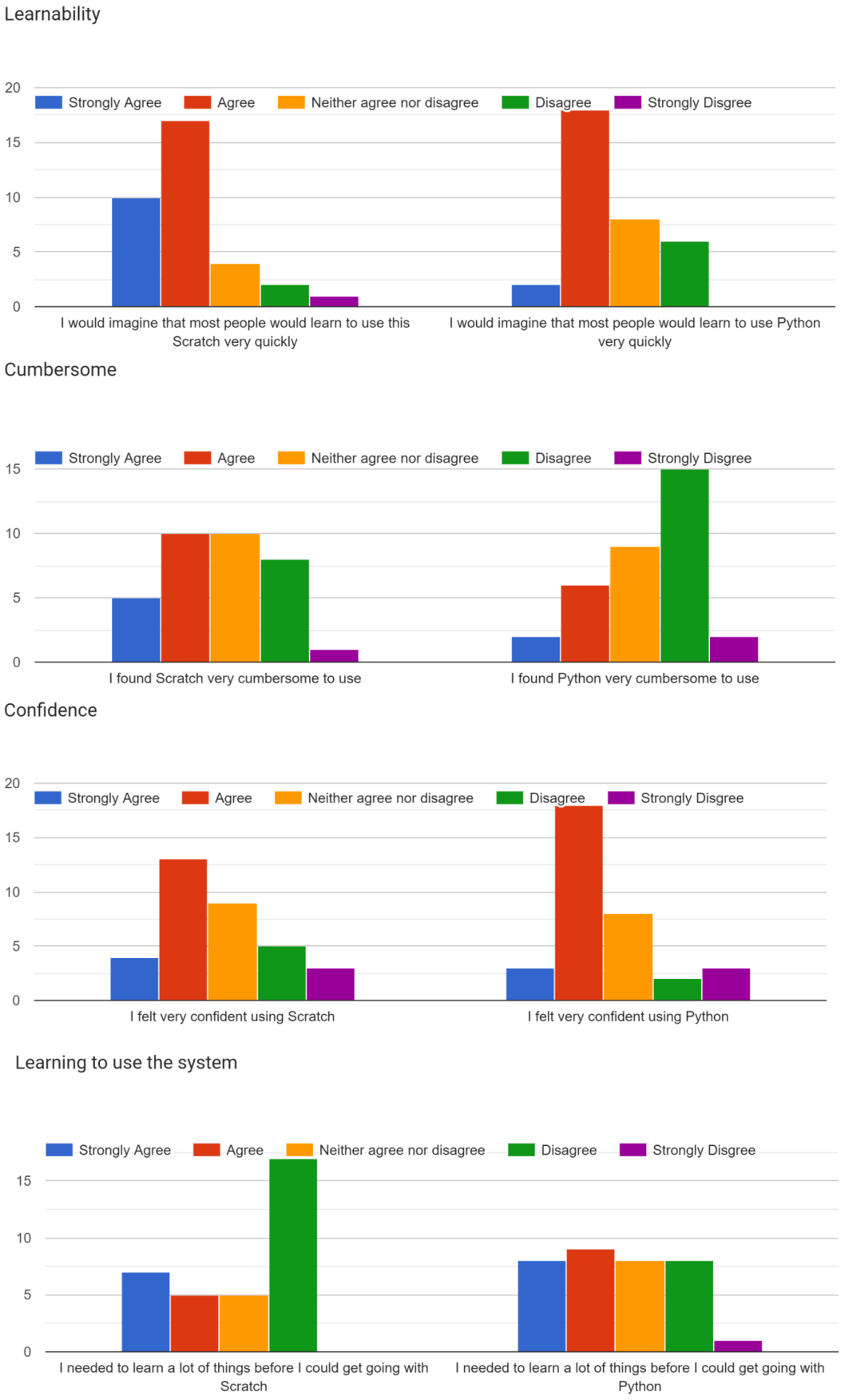


Fig. 10. Evaluation of Scratch and Python Platform using System Usability Scale (SUS)

References

[1] Robins, A. V. (2019). 12 novice programmers and introductory programming. *The Cambridge handbook of computing education research*, 327.

[2] Sakpere, A. B. (2021). Supporting the Learning and Teaching of a Practical-Based Course using Mobile Phone Technology. *University of Ibadan Journal of Science and Logics in ICT Research*, *6*(1 and 2), 145-157.

[3] Eteng, I., Akpotuzor, S., Akinola, S. O., & Agbonlahor, I. (2022). A review on effective approaches to teaching computer programming to undergraduates in developing countries. *Scientific African*, *16*, e01240

[4] Skaarseth, L. K. (2023). *Investigating the transfer from Scratch to Python in Norwegian secondary school* (Master's thesis).

[5] Balreira, D. G., Silveira, T. L. D., & Wickboldt, J. A. (2023). Investigating the impact of adopting Python and C languages for introductory engineering programming courses. *Computer Applications in Engineering Education*, *31*(1), 47-62.

[6] Sakpere, A. B. (2019). Using social platforms to increase engagement in teaching computer programming. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1-6).

[7] Kelly, W., McGrath, B., & Hubbard, D. (2023). Starting from 'scratch': Building young people's digital skills through a coding club collaboration with rural public libraries. *Journal of Librarianship and Information Science*, *55*(2), 487-499.

[8] Elhalid, O. B., Alm Alhelal, Z., & Hassan, S. (2023). Exploring the Fundamentals of Python Programming: A Comprehensive Guide for Beginners. *SAMER, Exploring the Fundamentals of Python Programming: A Comprehensive Guide for Beginners*.

[9] Federici, S., Gola, E., & Sergi, E. (2023). Is the Scratch Programming Environment Ideal for all? Enhancements to the Scratch IDE to Make it Easier to Use and More Useful for Students and Teachers.

[10] Mueller, J. P. (2023). *Beginning programming with Python for dummies*. John Wiley & Sons.

[11] Hetland, M. L. (2017). *Beginning Python: from novice to professional*. Apress.

[12] Khoirom, S., Sonia, M., Laikhuram, B., Laishram, J., & Singh, T. D. (2020). Comparative analysis of Python and Java for beginners. *Int. Res. J. Eng. Technol*, *7*(8), 4384-4407.

[13] Papadakis, Stamatios, et al (2014). "Novice programming environments. Scratch & app inventor: a first comparison." *Proceedings of the 2014 workshop on interaction design in educational environments*.

[14] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

[15] Gökçe, S., & Yenmez, A. A. (2023). Ingenuity of scratch programming on reflective thinking towards problem solving and computational thinking. *Education and Information Technologies*, *28*(5), 5493-5517.

[16] Xie, B. X. Y. (2016). *Progression of computational thinking skills demonstrated by app inventor users* (Doctoral dissertation, Massachusetts Institute of Technology).

[17] Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman and M. Resnick (2004), "Scratch: a sneak preview [education]," Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004., Kyoto, Japan, 2004, pp. 104-109, doi: 10.1109/C5.2004.1314376.

[18] Wolz, U., Leitner, H. H., Malan, D. J., & Maloney, J. (2009). Starting with scratch in CS 1. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 2-3).

[19] Moreno-León J. and Robles G. (2016), "Code to learn with Scratch? A systematic literature review," 2016 IEEE Global Engineering Education Conference (EDUCON), Abu Dhabi, United Arab Emirates, pp. 150-156, doi: 10.1109/EDUCON.2016.7474546.

[20] Kazemitabaar, M., Chyhir, V., Weintrop, D., & Grossman, T. (2022). Codestruct: Design and evaluation of an intermediary programming environment for novices to transition from scratch to python. In *Proceedings of the 21st Annual ACM Interaction Design and Children Conference* (pp. 261-273).

[21] Zdawczyk, C., & Varma, K. (2022). Engaging girls in computer science: Gender differences in attitudes and beliefs about learning scratch and python. *Computer Science Education*, 1-21.

[22] Jalal Nouri, Lechen Zhang, Linda Mannila & Eva Norén (2019): Development of computational thinking, digital competence and 21st century skills when learning programming in K-9, Education Inquiry, DOI: 10.1080/20004508.2019.1627844

[23] Van Toll, W., Egges, A., Fokker, J. D., van Toll, W., Egges, A., & Fokker, J. D. (2019). What Is Programming?. *Learning C# by Programming Games*, 9-23.

[24] Damianos Gavalas, Daphne Economou. Mobile Applications Programming Platforms and Development Tool. Handbook of Research on Mobile Software Engineering, (2012), 250-264. DOI: 10.4018/978-1-61520-655-1.ch015.

[25] Efecan, C. F., Sendag, S., & Gedik, N. (2020). Pioneers on the Case for Promoting Motivation to Teach Text-Based Programming. Journal of Educational Computing Research, 59(3), 453–469. doi:10.1177/0735633120966048

[26] Lavy, S. (2019). A Review of Character Strengths Interventions in Twenty-First-Century Schools: their Importance and How they can be Fostered. Applied Research in Quality of Life. doi:10.1007/s11482-018-9700-6

[27] Su, Y. S., Shao, M., & Zhao, L. (2022). Effect of mind mapping on creative thinking of children in scratch visual programming education. *Journal of Educational Computing Research*, *60*(4), 906-929.

[28] Oladele O. Campbell, Harrison I. Atagana (2022), Impact of a Scratch programming intervention on student engagement in a Nigerian polytechnic first-year class: verdict from the observers,Heliyon, Volume 8, Issue 3, 2022, e09191, ISSN 2405-8440, https://doi.org/10.1016/j.heliyon.2022.e09191.

[29] Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?. *Computers in Human Behavior*, *105*, 105849.

[30] Espinal, A., Vieira, C., & Guerrero-Bequis, V. (2023). Student ability and difficulties with transfer from a block-based programming language into other programming languages: A case study in Colombia. *Computer Science Education*, *33*(4), 567-599.

[31] Malan, D. J., and Leitner, H. H. (2007), "Scratch for Budding Computer Scientists," ACM SIGCSE Bulletin, 39, 223–227.

[32] Stewart, W., & Baek, K. (2023). Analyzing computational thinking studies in Scratch programming: A review of elementary education literature. *International Journal of Computer Science Education in Schools*, *6*(1), 35-58.

[33] Su, J., & Yang, W. (2023). A systematic review of integrating computational thinking in early childhood education. *Computers and Education Open*, *4*, 100122.
[34] Riihiaho, S. (2018). Usability testing. *The Wiley handbook of human computer interaction*, *1*, 255-275.
[35] Sakpere, A. B., Kayem, A. V., & Ndlovu, T. (2015). A usable and secure crime reporting system for technology resource constrained context. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops* (pp. 424-429). IEEE.
[36] Tzortzoglou, F., & Sofos, A. (2023). Evaluating the Usability of Mobile-Based Augmented Reality Applications for Education: A Systematic Review. *Research on E-Learning and ICT in Education*, 105-135.
[37] Adedeji F.M, Adekunle Y.A, Adebayo A.O, Alao O.D, Akande O.A (2022). Systematic Review on Usability Evaluation for University Websites. International Journal of Computer Applications Technology and Research.Volume 11 Issue 02, 22-28, 2022, ISSN:- 2319 8656 DOI:10.7753/IJCATR1102.1003
[38] Brooke, J. (2013). SUS: a retrospective. Journal of Usability Studies archive, 8, 29-40.
[39] Morales, J., Rusu, C., Botella, F., Quiñones, D. (2020). Programmer eXperience: A Set of Heuristics for Programming Environments. In: Meiselwitz, G. (eds) Social Computing and Social Media. Participation, User Experience, Consumer Experience, and Applications of Social Computing. HCII 2020. Lecture Notes in Computer Science(), vol 12195. Springer, Cham. https://doi.org/10.1007/978-3-030-49576-3_15
[40] Rahmatizadeh S, Kirakowski J, Valizadeh-Haghi S, Taheri M, Tavasoli S. A combination of three scales for measuring user perceived usability of a clinical information system: which approach produces the most informative results? Front Health Inform. 2024; 13: 193. DOI: 10.30699/fhi.v13i0.569
[41] Lewis, J. R., & Sauro, J. (2018). Item benchmarks for the system usability scale. Journal of Usability Studies, 13(3).
[42] GitLab, (2023). GitLab Handbook: System Usability Scale. https://handbook.gitlab.com/handbook/product/ux/performance-indicators/system-usability-scale
[43] Resnick, M. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. MIT press.
[44] Amadi, H. N. (2015). Impact of power outages on developing countries: evidence from rural households in Niger Delta, Nigeria. *Journal of Energy technologies and Policy*, *5*(3), 27-38.
[45] Jacob, O. N. (2020). An Investigation into the Challenges Preventing Students of Educational Administration and Planning from Using ICT for Learning in Nigeria Higher Institutions. *International Journal of Advances in Data and Information Systems*, *1*(2), 69-79.
[46] Pérez-Jorge, D., & Martínez-Murciano, M. C. (2022). Gamification with Scratch or App Inventor in Higher Education: A Systematic Review. *Future Internet*, *14*(12), 374.

Authors' Profiles

Aderonke Busayo Sakpere received both BSc and MSc degrees in Computer Science from the University of Ilorin, Nigeria. She holds a PhD in Computer Science from the University of Cape Town, South Africa. She is an Oracle PL/SQL Developer Certified Associate. She is a faculty member at the University of Ibadan with 12 years of experience in lecturing, research and mentoring. As a lecturer, she teaches both undergraduate and postgraduate students, with research interests spanning Human Computer Interaction (HCI), Data Privacy, ICT for Development, and Educational Technology. Dr Sakpere's continued academic and research excellence has earned her various honours and awards. They include Google ExploreCSR Grant, MIT Empowering The Teachers Fellowship, Selected Young Scientist by Lindau Nobel Laureate Meetings, Top 200 young researcher by Heidelberg Laureate Foundation (HLF), Hasso Plattner Institute (HPI) Fellowship, among others. She has been involved in different mentorship programmes and is the founder of Tech Girls Club which was set up to foster networking and interaction among females in technology. She is a co-founder of the Lindau Mentoring Hub hosted by Lindau Nobel Laureate Meetings in Germany. She has previosuly consulted with a digital innovation company named Qhala, on a Wikimedia Foundation Project.

Adedeji Folashade is an experienced UX/HCI researcher, also a lecturer at Lead City University, blending industry and academic success. She specializes in uncovering user behaviors, needs, and motivations to drive effective design choices. Passionate about applying HCI principles to tackle challenges, especially in developing regions. With a robust IT background spanning web development, database administration, and more, she is dedicated to nurturing the next generation of HCI professionals through teaching and mentorship. Active member of ACM SIGCHI, committed to staying current and contributing to the HCI community. Proficient in participatory research, usability studies, design thinking, and both quantitative and qualitative research methodologies.

How to cite this paper: Aderonke Busayo Sakpere, Adedeji Folashade, "Comparative Analysis of Two Programming Platforms for Beginners: Python and Scratch", International Journal of Education and Management Engineering (IJEME), Vol.14, No.5, pp. 46-62, 2024. DOI:10.5815/ijeme.2024.05.05