education sciences

Article

Using Robotics in the Learning of Computer Programming: Student Experiences Based on Experiential Learning Cycles

Reginald Gerald Govender * and Desmond Wesley Govender

Computer Science, School of Education, University of KwaZulu-Natal, Durban 3605, South Africa
* Correspondence: govenderr4@ukzn.ac.za

Abstract: The understanding of basic constructs in computer programming has always been seen by students as challenging, especially for novice programmers with no prior exposure at the school level. This paper emanates from a larger study and sets out to explore the use of robotics to enhance the learning of computer programming by asking: what are students' experiences of using robotics when learning to program? Guided by Kolb's Experiential Learning Cycle, a series of workshops were conducted that promotes hands-on learning and encourages the use of tools in the learning process for knowledge development. The site for this study was a university campus in KwaZulu-Natal, South Africa. The sample was composed of 75 students, most of whom were first-year students who had just started a computer course and had no prior exposure to Computer Programming. The findings showed that the meaningful adoption of Kolb's experiential learning has proven to be successful in the progressional development of computer programming constructs when using a physical component such as a robot. The use of microcontrollers that provide a robotic element offering a physical attribute during the learning of code proves to be affective. It is hoped that the findings of this study will contribute to the development of innovative methods to introduce computer programming through the use of robotics.

Keywords: computer programming; educational robotics; Kolb's Experiential Learning Cycle

check for updates

Citation: Govender, R.G.; Govender, D.W. Using Robotics in the Learning of Computer Programming: Student Experiences Based on Experiential Learning Cycles. *Educ. Sci.* **2023**, *13*, 322. https://doi.org/10.3390/educsci13030322

Academic Editors: Magdalena Ramos Navas-Parejo, Pedro Antonio García-Tudela, Lucía Lomba Portela and Marco Antonio Zamora Antuñano

Received: 2 March 2023
Revised: 20 March 2023
Accepted: 20 March 2023
Published: 21 March 2023

1. Introduction

South Africa continues to endure a critical shortage of digital skills, including those needed for computer programming [1–3]. It would seem that the greatest challenge faced by most students is the understanding of programming basics. This is especially true for novice programmers with no formal training in programming or for those in their first year of study at the tertiary level with no prior exposure at the school level [4,5]. Furthermore, the teaching of programming is classified as difficult, since computer programs and algorithms are complex constructs that require abstract thinking. Consequently, these concepts and processes are often regarded as difficult to teach and learn [6–9].

Numerous strategies have been employed to introduce students to programming to simplify the process and render understanding easy. Such strategies include block-based coding, visualisation tools and GUI environments to enhance understanding. However, such methods have not always been effective, and there is limited research on the best practices for learning to program [8,10–13].

The programming language used in this study was Python, as it is one of the most widely used general-purpose programming languages that contains high-level functionality [14,15]. Python was used in a procedural paradigm to introduce students to the basic principles of programming and was combined with a robotic element for prototype building as an educational tool. It was hoped that the proposed strategy would achieve better results than what has been achieved using various strategies, such as object-oriented approaches, which may not be ideal for introducing programming basics [16]. In addition, factors such as motivation, interest and belief in the subject content are essential for student success, as these factors render the content interesting and appealing [17–19]. These factors were also investigated in the study.

2. Research Problem

In South Africa, there is a lack of early exposure to the fundamentals of coding, let alone robotics. This is in contrast to other specialisation subjects in school, such as, geography, physical science, biology, history, etc., in which subject content in the prior phases lays a foundation for these subjects in the Further Education Training (FET) phase. Coding as a subject (Information Technology) can be selected only in the later years of schooling, at the FET phase (the final 3 high-school grades), making the Zone of Proximal Development (ZPD) difficult to scaffold.

The ZPD can be described as knowledge that a student can attain (by building on prior knowledge) with the assistance of facilitation to scaffold student development across the "gap" between what is known and the construction of new knowledge [20]. Students who lack prior exposure to programming cannot build new knowledge from any foundation, thereby adding to the abstract nature of the subject. As a result, many students may avoid fields at tertiary level that involve computer programming, due to their lack of prior exposure and, thus, foundation in the subject. Additionally, computer programming courses are sometimes perceived as uninteresting and demotivating, leading to high dropout rates [21]. Hence, this study sets out to explore the use of robotics to enhance the learning of computer programming in an attempt to answer the following question. What are students' experiences of using robotics when learning to program? It is hoped that the findings of this study will contribute to the development of innovative methods to introduce computer programming through the use of robotics.

3. Theoretical Framework

The study was guided by Kolb's Experiential Learning Cycle (KELC), as each workshop in the series represented a cycle. KELC promotes hands-on learning and encourages the use of tools in the learning process for knowledge development [22,23]. Hence, KELC was best suited for using the robotic element through prototype building as an educational tool.

To achieve effective learning, the learner would need to progress through a cycle [24]. The four stages that govern KELC formed an iterative process in which each workshop session represented a micro-cycle following the principles of KELC [25,26]:

1. Concrete experience—active engagement or experience;
2. Reflective observation—reflection on the activity or experience;
3. Abstract conceptualization—building knowledge from the experience; and
4. Active experimentation—testing the acquired skills or abilities.

Each of the six workshops involved coding, designing and testing prototypes by using the Arduino robot kit. Each workshop was designed to continuously build on the previous, thus targeting the Zone of Proximal Development [20] and building on existing knowledge. Each workshop consisted of three activities addressing the four KELC principles.

Table 1 shows how Activities 1, 2 and 3 are linked to KELC. Activity 1 provided the participant with an encounter with a concrete experience, while reflective observation and abstract conceptualisation are encountered in Activity 2. Lastly, Activity 3 allowed for active experimentation.

Activity 1 was a guided, step-by-step activity that introduced and explored a programming concept. Activity 2 was a semi-guided self-discovery programming mission that formed the foundation of the next activity. Activity 3 was a task based on Activity 1 and Activity 2. This final activity was completed without any help. Each of the six workshops was designed to continuously build on the previous, thus targeting the Zone of Proximal Development and building on existing knowledge. It was important to facilitate the learning online by using a set learning path that was created in LMS and with criteria in place to prevent the skipping of workshops or activities.

Table 1. Kolb's Experiential Learning Cycle vs the DBR prototyping phase.

Kolb's Experiential Learning Cycle	**Prototyping—Micro-Cycle (Workshop)**
Concrete experience	Activity 1
Reflective observation	Activity 2
Abstract conceptualisation	
Active experimentation	Activity 3

Note. Kolb's Experiential Learning Cycle imposed on the three activities per workshop.

4. Study Design

The methodology for the study is an intervention study which is a design study. Design-Based Research (DBR) usually occurs in three phases: planning, experimental and analysis [27]. The planning phase involves the design of the activities, while the experiential phase is the completion of the set activities by the students. Analysis is the review of the results. The intervention aimed to present first-time programming students to build–code–test (prototype coding) in learning how to code in a text-based language, thus offering a more learner-centred and scenario-oriented approach. The study adopted a mixed method approach which is favoured in DBR by using Likert scale surveys, workshop activities and a focus group interview in the generation and collection of data.

4.1. Participants and Procedures

The site for this study was a university campus in KwaZulu-Natal, South Africa. The necessary permissions were obtained from the registrar's office and ethics board before making any contact with students. Thereafter, an invitation was sent out to students outlining the study, criteria to participate, their anonymity and the voluntariness of the project. A combination of non-probability sampling techniques was adopted, namely, quota and convenience sampling. Participants needed to meet the following criteria to be selected:

1. Registered in a computer course/module; and
2. No exposure to computer programming during the duration of the degree.

After a shortlisting process, the sample was composed of 75 students, most of whom were first-year students who had just started a computer course and had no prior exposure to computer programming at the University. Due to the COVID-19 pandemic, participants were required to pick up a robotic kit from the University prior to the start of the online series of workshops. There were six online workshops that covered programming using robotics and coding by using the Arduino UNO R3 kit. The study utilised this kit due to it being cost-effective and easily sourced. The students did not need to return the kits, allowing for continued exploration.

In addition to the six workshops, an introductory workshop served to familiarise participants with the components and the software environment. The workshops were hosted online, allowing for queries and communication among peers to be posted on the forum.

A pilot study was conducted, which allowed for the testing and refining of the workshops and data-collection instruments before the main study [28,29]. Additionally, the pilot study helped to ensure the reliability, validity and practicability of using the data-collection tools [28]. The necessary changes and modifications were made.

4.2. Before the Workshop Sessions

A pre-workshop session was excluded from the six micro-cycles (iteration), as its purpose was to familiarise participants with the IDLE (Integrated Development and Learning Environment), Python language and the setup. In addition, the background to computer programming and an introduction to coding was covered. Most importantly, the session familiarised participants with the Arduino components that were given to them.

4.3. Data Collection Tools

A 5-point, 10-item Likert scale survey that consisted of 10 statements served to capture each participant's self-efficacy, perceptions and attitude towards programming and robotics. This data-collection instrument was administered first to prevent any prior influence from other instruments that might have affected an individual response to programming.

Each online workshop session started with a guided step-by-step activity taking participants through an activity: Activity 1, followed by a semi guided self-explorative activity: Activity 2 and a no-help activity: Activity 3 as per Kolb's experiential learning cycle. Workshop sessions 1, 2 and 3 introduced the basics of programming: syntax, input, output, if statements, for loops and while loops. Workshop sessions 4, 5 and 6 provided consolidation and further exploration of the robotic components. At the end of each activity (micro-cycle), participants were prompted to complete a self-formative evaluation in the form of a 5-point Likert scale: 1—Very difficult, 2—Difficult, 3—Neutral, 4—Easy and 5—Very easy.

4.4. After the Workshop Sessions

A focus-group interview was planned in which the researcher used a set sequence of questions to probe for participants' insights [30]. The interview was conducted after the six workshops, as this prompted in-depth explanations that provided more information on participants' experiences of learning code through the use of the Arduino robot. To prevent the risk of some participants dominating the discussion, the chair (researcher) encouraged every participant to share their views, thus contributing equally to the discussion. The interview consisted of 8 leading questions, which lasted approximately 45 minutes. The focus-group interview was held on Zoom with participants who volunteered to be part of the discussion.

5. Findings and Discussion

For reporting purposes, to convey a coherent discussion, the findings from each of the workshops are presented in Table 2; thereafter, the analysis of the Likert and focus-group interview follows.

Table 2. Summary of all micro-cycles.

	Activities	**1—Very Difficult**	**2—Difficult**	**3—Neutral**	**4—Easy**	**5—Very Easy**
Workshop One	Activity 1: Getting to know the hardware	1%	5%	23%	37%	33%
	Activity 2: Hello world	1%	11%	21%	44%	23%
	Activity 3: Blinking LED	5%	16%	36%	29%	13%
Workshop Two	Activity 1: User control	1%	4%	28%	31%	36%
	Activity 2: Blinking LED	0%	8%	25%	45%	21%
	Activity 3: Traffic lights	7%	17%	33%	21%	21%
Workshop Three	Activity 1: Reading the LDR value	0%	1%	36%	29%	33%
	Activity 2: Continuous reading of LDR values	0%	9%	28%	28%	35%
	Activity 3: Night sensor	7%	21%	37%	27%	8%

Table 2. *Cont.*

	Activities	1—Very Difficult	2—Difficult	3—Neutral	4—Easy	5—Very Easy
Workshop Four	Activity 1: Buzzer tone	1%	4%	23%	21%	51%
	Activity 2: Flame Sensor	0%	13%	21%	23%	43%
	Activity 3: Fire alarm	4%	25%	27%	39%	5%
Workshop Five	Activity 1: Reading from a tilt sensor	3%	5%	23%	24%	45%
	Activity 2: Tilt lights ON	1%	12%	20%	24%	43%
	Activity 3: Earthquake	8%	20%	32%	19%	21%
Workshop Six	Activity 1: Turn to a degree	0%	7%	23%	23%	48%
	Activity 2: Button open and close	0%	13%	23%	27%	37%
	Activity 3: Fibonacci in motion	9%	21%	35%	21%	13%

Note. Feedback based on self-evaluation for each workshop session. Based on n = 75.

5.1. Micro-Cycles

5.1.1. Workshop Session One

Participants were acquainted with the components of the Arduino kit. This included the exploration of the microcontroller, breadboard and jumper cables. Participants started coding their first program called *Hello world* while familiarising themselves with the Python language. The highlight of this introduction workshop was the use of the iteration structure (for-loop) in controlling a LED switching ON/OFF for a number of times following the display of *Hello world* on the PC screen. In total, 44% of the participants found Activity 1 to be relatively Easy, which was followed closely by 37% of participants who found Activity 2 to be Easy, and 36% of participants responded Neutral for Activity 3.

5.1.2. Workshop Session Two

On the basis that participants were familiar with the Arduino UNO R3, IDLE and basic Python code setup, this session introduced user input and conditional statements (if statement). The highlight of this session was coding a traffic robot system prototype that used red, yellow and green LEDs. The majority of participants found Activity 2 to be Easy (45%), followed by Activity 1 being Very easy (36%) and, lastly, Activity 3 being Neutral (33%).

5.1.3. Workshop Session Three

The use of iteration structures was furthered by emphasising and introducing the while-loop structure. In addition, participants were required to code nested structures. The highlight of this session was creating a night sensor prototype by using an LDR (Light-Dependent Resistor). The majority of participants found Activity 3 to be Neutral (37%), followed by Activity 1 being Neutral (36%) and, lastly, Activity 2 being Very easy (35%).

5.1.4. Workshop Session Four

The need for iteration structures was reinforced while different Arduino parts were used, especially the use of sensors that constantly need to read in values around its surrounding. The highlight of this session was creating a fire alarm by building a prototype by using a flame sensor and a buzzer. The majority of participants found Activity 1 to be

Very easy (51%), followed by Activity 2 being Very easy (43%) and, lastly, Activity 3 being Easy (39%).

5.1.5. Workshop Session Five

This session used the buzzer that was introduced in the previous session but with the tilt sensor. The highlight of this consolidation session was creating an earthquake detector. The majority of the participants found Activity 1 to be Very easy (45%), followed by Activity 2 being Very easy (43%) and, lastly, Activity 3 being Neutral (32%).

5.1.6. Workshop Session Six

Previous iteration and conditional coding structures were incorporated together with nested structures. The latter added complexity and unified development of coding structures. The highlight during this session was coding the movement of an actuator (servo motor) according to the Fibonacci sequence. The majority of participants found Activity 1 to be Very easy (48%), followed by Activity 2 being Very easy (37%) and, lastly, Activity 3 being Neutral (35%).

5.2. Composite Accomplishment Ratings

Based on modal responses per activity, per workshop, all accomplishments ranged between Neutral, Easy and Very easy (Table 3).

Table 3. Composite accomplishment ratings based on the majority responses.

Workshops	Activity 1	Activity 2	Activity 3
Workshop 1	Easy	Easy	Neutral
Workshop 2	Very easy	Easy	Neutral
Workshop 3	Neutral	Very easy	Neutral
Workshop 4	Very easy	Very easy	Easy
Workshop 5	Very easy	Very easy	Neutral
Workshop 6	Very easy	Very easy	Neutral

Note. There are no combinations of responses. The table represents the modal response.

It is not surprising that most of the last activities (Activity 3) had a Neutral response, since this activity was unguided, requiring participants to use their understanding and knowledge development from Activities 1 and 2. Similarly, most Activities 1 and 2 elicited a response of Easy and Very easy, as these activities were guided (Activity 1) or partially guided (Activity 2). Workshops 3 and 4 stand out from the other responses.

In workshop 3, Activities 1 and 3 had Neutral responses. Activity 1 involved reading values from the Light-Dependent Resistor (LDR) that was introduced in this workshop, while Activity 3 involved the use of the LDR and LED from the previous workshops. The Neutral accomplishment responses, especially for Activity 1, were likely due to the high sensitivity of the LDR sensor. The values are quick to fluctuate due to light intensity and light exposure from the surroundings of the LDR sensor, hence resulting in volatile sensor readings. All workshop 4 activities were rated Easy-Very easy. This workshop involved prototypes that used the flame sensor and buzzer that were introduced during the workshop. This is likely due to the flame sensor being very accurate to the temperature (a smartphone flashlight can be used to imitate a fire flame) of light such that by default, it registers 1.0 when there is no flame detected, and the value immediately starts to decrease when a flame is detected.

5.3. Likert Scale and Survey

A 5-point, 10-item Likert scale survey was given first to students to complete. The survey consisted of 10 items that were designed to gather initial perceptions from students regarding coding and robotics.

Most responses fell in the extremes of the Likert scale: Strongly agree and Strongly disagree, except items 3, 8 and 9 (Figure 1). These exceptions indicate a more even distribution, resulting in mixed reactions to items 3: I think programming is too technical; 8. My perception of programming is that it is difficult to learn; and 9. I think programming is hard. Five items consisted of negatively worded statements. In determining the Cronbach's alpha (α), the five positively worded statements remained as is (Table 4), while the five negatively worded statements were reverse-coded before calculating α (Table 5).

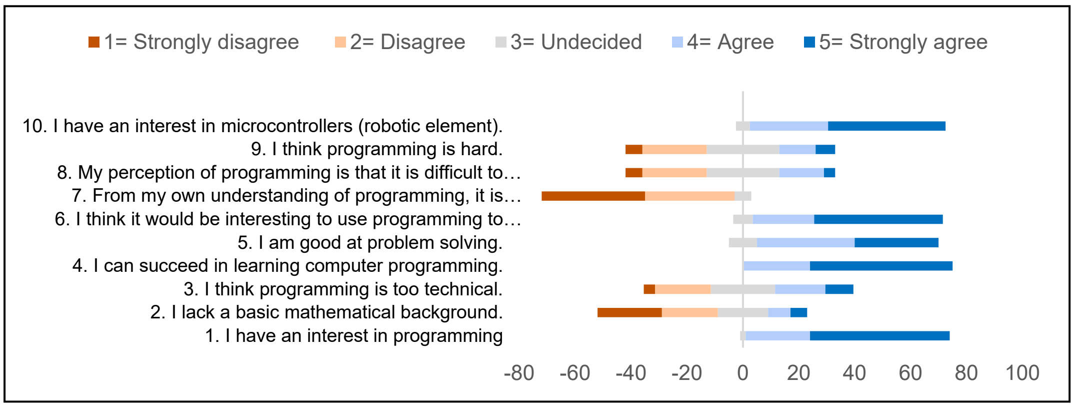

Figure 1. Distribution of responses from pre-survey. A visual representation of the responses from the pre-survey.

Table 4. Reliability statistics of non-reverse coded items.

Item	**α if Item Deleted**
1. I have an interest in programming.	0.684
4. I can succeed in learning computer programming.	0.669
5. I am good at problem solving.	0.712
6. I think it would be interesting to use programming to solve problems.	0.623
10. I have an interest in microcontrollers (robotic element).	0.637
Reliability	0.714

Note. Positively worded statements.

Table 5. Reliability statistics of reverse-coded items.

Item	**α if Item Deleted**
2. I lack a basic mathematical background.	0.619
3. I think programming is too technical.	0.635
7. From my own understanding of programming, it is boring.	0.751
8. My perception of programming is that it is difficult to learn.	0.621
9. I think programming is hard.	0.502
Reliability	0.690

Note. Negatively worded statements.

A Cronbach's alpha above 0.70 is generally an acceptable value [31]. As remarked by [32], an α of 0.60–0.70 indicates an acceptable reliability level. As shown in Table 4, the non-reverse-coded items are therefore acceptable, with $\alpha \geq 0.7$. It is important to note that all the non-reverse-coded items were framed from a personal perspective, with all statements starting with "I . . . " prompting a response.

Table 5 shows all reverse-coded items are acceptable, as $\alpha = 0.690$ [32]. As supported by [33], one can accept values near 0.60. To probe further, each item from the survey questionnaire is depicted in Table 6, offering an understanding of student self-efficacy, perceptions and attitudes towards programming.

Table 6. Response per item from survey.

Statements	**1 = Strongly Disagree**	**2 = Disagree**	**3 = Undecided**	**4 = Agree**	**5 = Strongly Agree**
1. I have an interest in programming	0.00%	0.00%	2.67%	30.67%	66.67%
2. I lack a basic mathematical background.	30.67%	26.67%	24.00%	10.67%	8.00%
3. I think programming is too technical.	5.33%	26.67%	30.67%	24.00%	13.33%
4. I can succeed in learning computer programming.	0.00%	0.00%	0.00%	32.00%	68.00%
5. I am good at problem solving.	0.00%	0.00%	13.33%	46.67%	40.00%
6. I think it would be interesting to use programming to solve problems.	0.00%	0.00%	9.33%	29.33%	61.33%
7. From my own understanding of programming, it is boring.	49.33%	42.67%	8.00%	0.00%	0.00%
8. My perception of programming is that it is difficult to learn.	8.00%	30.67%	34.67%	21.33%	5.33%
9. I think programming is hard.	8.00%	30.67%	34.67%	17.33%	9.33%
10. I have an interest in microcontrollers (robotic element).	0.00%	0.00%	6.67%	37.33%	56.00%

For reporting purposes, the percentages for Strongly disagree and disagree are combined for some values, as is the case with Agree and Strongly agree in the excerpts that follow.

5.3.1. Response to Item 1: I Have an Interest in Programming

The majority of students had a keen interest in programming (97.34%—Agree + Strongly agree), while a small percentage are undecided (2.67%). This is supported by the following excerpts:

> "It was interesting to learn how we can use basic programming to do tasks" (P17, Focus Group Interview).

> "Coding was interesting for me; it is something that I have wanted to learn for a long time and I think it is a very valuable skill" (P15, Focus Group Interview).

Undecided responses may arise from students who are unsure of whether they find the robotic element or the coding aspect motivating, since the robotic element introduces a tangible component to the learning process and acts as a manipulative. The use of manipulatives in learning how to code has been found to be a key motivation for students [34,35], as they are actively involved in the learning process. The latter is affirmed by the following excerpts:

> "It was an interesting journey, learning to work with a microcontroller" (P60, Focus Group Interview).

> "The robot just makes the learning and understanding quicker" (P62, Focus Group Interview).

> "Coding of the buzzer was very interesting. I feel like I can use that information in developing solutions in the future (knowledge development of coding the robotic element) (P17, Focus Group Interview).

> "The strategy I used was to build my prototype and then; I will code because after building, I would have an idea of what the code must do and what must take place" (P18, Focus Group Interview).

Similar research [36] found that a student's interest in learning how to code is stimulated through the manipulative. Another similar study [37] found that students displayed high levels of engagement due to there being a physical object.

5.3.2. Response to Item 2: I Lack a Basic Mathematical Background

Most students (57.34%—Disagree + Strongly disagree) are of the view of having acquired the basics in mathematical skills. Interestingly, some students (18.67%—Agree + Strongly agree) say they do not have an essential mathematical background, while 24.00%

are unsure. Ref. [38] reason that confidence in mathematical skill is associated with self-awareness and self-efficacy; thus, having a positive outlook on one's mathematical ability may improve performance.

5.3.3. Response to Item 3: I Think Programming Is Too Technical

Many students are unsure (30.67%) if programming is technical, while the majority agree (37.33%—Agree + Strongly agree) that programming is technical. The findings of this study are similar to the findings by [39] that, even with interventions, students find programming difficult, leading to a lack in motivation and high dropout rates. Interestingly, some students do not think programming is too technical (32.00%—Disagree + Strongly disagree). Thus, the responses represent a relatively mixed view on the technicality of programming. It is important to note that all students met the criteria set out for the study, i.e., not having any exposure to computer programming during the duration of their degree but being currently registered in a computer course/module. The following excerpts support these mixed views:

> "The coding started out to be really motivating at the beginning and then eventually I will just get bored of it … although it was amazing to code" (P42, Focus Group Interview).
>
> "Coding somethings is tedious and then it's not like you can really see your progress when you stuck" (P1, Focus Group Interview).
>
> "The thing that I liked the most was you could read the programming language and easily understand what it does" (P18, Focus Group Interview).

5.3.4. Response to Item 4: I Can Succeed in Learning Computer Programming

All students perceived that they could succeed in learning programming. In other words, none of the students doubted their capabilities, with the majority (68.00%) strongly agreeing that they could succeed in learning computer programming. Similar to the findings from the response to item 2, the cohort of students exhibited high confidence related to self-awareness and self-efficacy [38]. In total, 68.00% of the students displayed intrinsic motivation and perceived programming as enjoyable and interesting [40], as confirmed by the following:

> "I really enjoyed this workshop it was an interesting journey, learning to work with a microcontroller. It was a bit difficult and it did require a bit of thinking, but nothing was impossible" (P60, Focus Group Interview).
>
> "In high school, I did Java as a programming language. But now doing Python, it's less complicated and simple … everything is straightforward" (P15, Focus Group Interview).

5.3.5. Response to Item 5: I Am Good at Problem Solving

Like the results in the response to item 4, most students held the view that they could excel in learning to program; the majority responded that they were good at problem solving (86.67%, Agree + Strongly agree). On the other hand, some were undecided (13.33%). [41] found that some students doubt their problem-solving abilities only to reflect their potential later.

5.3.6. Response to Item 6: I Think It Would Be Interesting to Use Programming to Solve Problems

Interestingly, no student refuted the use of programming to solve problems, with a few being unsure (9.33%, undecided). Based on the sample selection criterion, it was expected for some students to be undecided as they had no previous exposure to computer programming and robotics, as supported by the following excerpts:

> "I got an idea of how they (robotic element) are used in real-world situations … so you get to know how things work" (P15, Focus Group Interview).

> "They (robotic element) are very applicable in real-life problems" (P9, Focus Group Interview).

> "I enjoyed the problem solving; I enjoy combining things and then putting them together to make up one thing" (P26, Focus Group Interview).

5.3.7. Response to Item 7: From My Own Understanding of Programming, It Is Boring

A minority of students (8.00%) were unsure if programming is boring or not. On the other hand, most students reported that computer programming is not mundane (92.00%, Disagree + Strongly disagree). As noted, earlier programming can be complex and difficult [42], which leads to boredom [43]. However, through manipulatives, such as robots [44] and game design [16], negative attitudes towards computer programming can be changed.

5.3.8. Response to Item 8: My Perception of Programming Is That It Is Difficult to Learn

Some students were unsure if programming is difficult (34.67%, Undecided), as this was their first encounter with coding. A minority viewed programming as difficult (26.66%, Agree + Strongly agree). The majority of students responded that programming is not difficult to learn (38.67%, Disagree + Strongly disagree).

5.3.9. Response to Item 9: I Think Programming Is Hard

Strikingly, the results in the response to item 9: I think programming is hard. are similar to the response to item 8: My perception of programming is that it is difficult to learn. regarding the idea of programming as challenging to learn for Strongly disagree, Disagree and Undecided. The minority of students thought programming is complex (26.66%, Agree + Strongly agree), similar to the results in the response to item 8: My perception of programming is that it is difficult to learn. The majority believed that programming is not hard (38.67%, Disagree + Strongly disagree), again similar to the results in Table 6. The uncertainty (34.67%) of whether programming is hard or not is, once again, similar to the results in the response to item 8: My perception of programming is that it is difficult to learn.

5.3.10. Response to Item 10: I Have an Interest in Microcontrollers (Robotic Element)

A large number of students purported to have an interest in microcontrollers (93.33%, Agree + Strongly agree), while few are undecided (6.67%). This finding suggests and confirms the results in Table 2 that the robotic element had definitely sparked curiosity and interest towards the learning of programming. This indication is supported by the findings by [35] that manipulatives such as the microcontroller used in this study stimulated students' interest and kept students engaged while learning how to code.

> "I struggled with some programming; it can get quite abstract, but the microcontroller makes it most fun and I feel like it solidifies understanding in some way" (P62, Focus Group Interview).

> "With this workshop having to hold and work the microcontroller made it (coding) interesting" (P42, Focus Group Interview).

6. Conclusions

The present study set out to understand students' experiences of using robotics when learning to program. It was found that the meaningful adoption of Kolb's Experiential Learning Cycle has proven to be successful in the progressional development of computer programming concepts/constructs when using a physical component such as a robot. Each workshop that formed a cycle offered an iterative experience with a gradual cognitive demand level through the transition of the three prototyping activities. Starting with a concrete experience, followed by a reflective observation and, thereafter, an abstract conceptualisation and finally, an active experimentation of learning to code the robot while consolidating the computer programming concept. All of this proved to offer an innovative method to introduce computer programming through the use of robotics. This finding is affirmed by the large percentage of students indicating the workshop activities to be easy to very easy, but the activities still provided a challenge to some students given the neutral responses.

Upon examining the Likert-scale data which was complemented by the interviews, it was found that the majority of students held a positive attitude towards the learning of computer programming through the use of robotics. The use of microcontrollers, such as the Arduino, which provide a robotic element that offers a physical attribute during the learning of code proves to be affective. It is strongly encouraged that there needs to be an increase in such activities, hence bringing ease to computer programming for the novice. It is hoped that this study provides a gateway for further innovative methods which use robotics in the learning of computer programming. Future research could include a larger dataset and adopt an experimental design using a pre-test–post-test.

Author Contributions: Conceptualization, R.G.G.; methodology, R.G.G.; software, R.G.G.; validation, R.G.G.; formal analysis, R.G.G.; investigation, R.G.G.; resources, R.G.G.; data curation, R.G.G.; writing—original draft preparation, R.G.G.; writing—review and editing, R.G.G. and D.W.G.; visualization, R.G.G.; supervision, D.W.G.; project administration, R.G.G.; funding acquisition, R.G.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Research Foundation, South Africa grant number and 122017. The APC was fully waived based on special invite to R.G.G.

Institutional Review Board Statement: The study was granted full approval by the Humanities and Social Sciences Research Ethics Committee (HSSREC) of the University of KwaZulu-Natal (Protocol reference number: HSS/00000574/019M, 16 August 2019).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Limited data is available from the authors upon reasonable request. Note some data is unavailable due to privacy or ethical restrictions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Business Chief. Available online: https://businesschief.eu/leadership-and-strategy/digital-skills-south-african-graduates-will-need-2020 (accessed on 12 June 2022).
2. Maisiri, W.; Van Dyk, L. Industry 4.0 skills: A perspective of the South African manufacturing industry. *SA J. Hum. Resour. Manag.* **2021**, *19*, a1416. [CrossRef]
3. ITWEB. Available online: https://www.itweb.co.za/content/4r1ly7RblGE7pmda (accessed on 21 June 2022).
4. Lo, C.A.; Lin, Y.T.; Wu, C.C. Which programming language should students learn first? A comparison of Java and Python. In Proceedings of the 2015 International Conference on Learning and Teaching in Computing and Engineering LaTiCE, Taipei, Taiwan, 9–12 April 2015.
5. Saeli, M.; Perrenet, J.; Jochems, W.M.G.; Zwaneveld, B. Teaching programming in secondary school: A pedagogical content knowledge perspective. *Inform. Educ.* **2011**, *10*, 73–88. [CrossRef]
6. Bati, T.B.; Gelderblom, H.; Van Biljon, J. A blended learning approach for teaching computer programming: Design for large classes in Sub-Saharan Africa. *Comput. Sci. Educ.* **2014**, *24*, 71–99. [CrossRef]
7. Mendes, A.J.; Paquete, L.; Cardoso, A.; Gomes, A. Increasing student commitment in introductory programming learning. In Proceedings of the Soaring to New Heights in Engineering Education: 2012 Frontiers in Education Conference, Seattle, WA, USA, 3–6 October 2012. [CrossRef]
8. Olsson, M.; Mozelius, P.; Collin, J. Visualisation and gamification of e-learning and programming education. *Electron. J. E-Learn.* **2015**, *13*, 441–454.
9. Chen, Q.; Tang, Y.; Li, L.; Yang, G.; Yang, M.; Xie, Z.; Huang, R. A practice on Lego Mindstorms for computer science freshman experimental education. *Destech Trans. Soc. Sci. Educ. Hum. Sci.* **2017**, *1*, 17–21. [CrossRef] [PubMed]
10. Kalelioğlu, F.; Gülbahar, Y. The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Inform. Educ.* **2014**, *13*, 33–50. [CrossRef]
11. Tanrikulu, E.; Schaefer, B.C. The users who touched the ceiling of scratch. *Procedia Soc. Behav. Sci.* **2011**, *28*, 764–769. [CrossRef]

12. Techapalokul, P.; Tilevich, E. Understanding recurring quality problems and their impact on code sharing in block-based software. In Proceedings of the 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Raleigh, NC, USA, 11–14 October 2017. [CrossRef]
13. Erol, O.; Kurt, A.A. The effects of teaching programming with scratch on pre-service information technology teachers' motivation and achievement. *Comput. Hum. Behav.* **2017**, *77*, 11–18. [CrossRef]
14. Shein, E. Python for beginners. *Commun. ACM* **2015**, *58*, 19–21. [CrossRef]
15. Summerfield, M. *Programming in Python 3: A Complete Introduction to the Python Language*, 2nd ed.; Addison-Wesley Professional: Indianapolis, IN, USA, 2010; pp. 1–525.
16. Govender, R.G.; Govender, D.W. ROBOPROG: Learning of flowcharts through a gamified experience. *Int. J. Bus. Manag. Stud.* **2020**, *12*, 612–624.
17. Douglas, M.E.; Peecksen, S.; Rogers, J.; Simmons, M. College students' motivation and donfidence for ePortfolio use. *Int. J. Eportfolio* **2019**, *9*, 1–16.
18. Blotnicky, K.A.; Franz-Odendaal, T.; French, F.; Joy, P. A study of the correlation between STEM career knowledge, mathematics self-efficacy, career interests, and career activities on the likelihood of pursuing a STEM career among middle school students. *Int. J. STEM Educ.* **2018**, *5*, 1–15. [CrossRef]
19. Fırat, M.; Kılınç, H.; Yüzer, T.V. Level of intrinsic motivation of distance education students in e-learning environments. *J. Comput. Assist. Learn.* **2018**, *34*, 63–70. [CrossRef]
20. Vygotsky, L.S. *Mind and Society: The Development of Higher Mental Processes*; Harvard University Press: Cambridge, MA, USA, 1978; pp. 1–174. [CrossRef]
21. Lin, H.T.; Kuo, T.H. Teaching programming technique with edutainment robot construction. In Proceedings of the 2nd International Conference on Education Technology and Computer, Shanghai, China, 22–24 June 2010. [CrossRef]
22. Bergsteiner, H.; Avery, G.C.; Neumann, R. Kolb's experiential learning model: Critique from a modelling perspective. *Stud. Contin. Educ.* **2010**, *32*, 29–46. [CrossRef]
23. Morris, T.M. Experiential learning A systematic review and revision of Kolb's model. *Interact. Learn. Environ.* **2020**, *28*, 1064–1077. [CrossRef]
24. Kolb, D.A. Learning Styles and Disciplinary Differences. In *The Modern American College*; Chickering, A.W., Ed.; Jossey-Bass: San Francisco, CA, USA, 1981; pp. 232–255.
25. Kolb, A.; Kolb, D. Eight important things to know about the experiential learning cycle. *Aust. Educ. Lead.* **2018**, *40*, 8–14.
26. Kolb, D. *Experiential Learning: Experience as the Source of Learning and Development*, 2nd ed.; Pearson: Hoboken, NJ, USA, 2015; pp. 1–377.
27. Gravemeijer, K.; Cobb, P. Design research from a learning design perspective. In *Educational Design Research*, 1st ed.; Van den Akker, J., Gravemeijer, K., McKenney, S., Nieveen, N., Eds.; Routledge: London, UK, 2006; pp. 17–51.
28. Cohen, L.; Manion, L.; Morrison, K. *Research Methods in Education*, 8th ed.; Routledge: London, UK, 2017; pp. 469–504.
29. Denscombe, M. *The Good Research Guide*, 5th ed.; McGraw-Hill: London, UK, 2014; pp. 163–225.
30. Krueger, R.A.; Casey, M.A. Focus group interviewing. In *Handbook of Practical Program Evaluation*, 4th ed.; Newcomer, K.E., Hatry, H.P., Wholey, J.S., Eds.; Jossey-Bass: San Francisco, CA, USA, 2015; pp. 506–534.
31. Nunnally, J.C. *Psychometric Theory*, 2nd ed.; McGraw-Hill: New York, NY, USA, 1978; pp. 1–701.
32. Hulin, C.; Netemeyer, R.; Cudeck, R. Can a reliability coefficient be too high? *J. Consum. Psychol.* **2001**, *10*, 55–58. [CrossRef] [PubMed]
33. Hair, J.F.; Hult, G.T.M.; Ringle, C.; Sarstedt, M. *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM)*, 2nd ed.; Sage: Newbury Park, CA, USA, 2016; pp. 1–384.
34. Nugent, G.; Barker, B.; Grandgenett, N.; Adamchuk, V. The use of digital manipulatives in K-12: Robotics, GPS/GIS and programming. In Proceedings of the 39th IEEE Frontiers in Education Conference, San Antonio, TX, USA, 18–21 October 2009. [CrossRef]
35. Merkouris, A.; Chorianopoulos, K.; Kameas, A. Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Trans. Comput. Educ. TOCE* **2017**, *17*, 1–22. [CrossRef]
36. Yilmaz, I.; Koc, M. The consequences of robotics programming education on computational thinking skills: An intervention of the young engineer's workshop (yew). *Comput. Appl. Eng. Educ.* **2021**, *29*, 191–208. [CrossRef]
37. Ioannou, A.; Makridou, E. Exploring the potentials of educational robotics in the development of computational thinking: A summary of current research and practical proposal for future work. *Educ. Inf. Technol.* **2018**, *23*, 2531–2544. [CrossRef]
38. Khasawneh, E.; Gosling, C.; Williams, B. What impact does maths anxiety have on university students? *BMC Psychol.* **2021**, *9*, 1–9. [CrossRef] [PubMed]
39. Sobral, S.R. Flipped Classrooms for Introductory Computer Programming Courses. *Int. J. Inf. Educ. Technol.* **2021**, *11*, 178–183. [CrossRef]
40. Zainal, N.F.A.; Shahrani, S.; Yatim, N.F.M.; Abd Rahman, R.; Rahmat, M.; Latih, R. Students' perception and motivation towards programming. *Procedia-Soc. Behav. Sci.* **2012**, *59*, 277–286. [CrossRef]
41. Belski, I. Teaching thinking and problem solving at university: A course on TRIZ. *Creat. Innov. Manag.* **2009**, *18*, 101–108. [CrossRef]

42. Rubio Escudero, M.A.; Mañoso Hierro, C.M.; Pérez de Madrid Pablo, A. Using arduino to enhance computer programming courses in science and engineering. In Proceedings of the 5th International Conference on Education and New Learning Technologies, Barcelona, Spain, 1–3 July 2013.
43. Khaleel, F.L.; Ashaari, N.S.; Tengku, T.S.M.; Ismail, A. Programming Learning Requirements Based on Multi Perspectives. *Int. J. Electr. Comput. Eng.* **2017**, *7*, 1299–1307. [CrossRef]
44. Kurebayashi, S.; Kamada, T.; Kanemune, S. Learning computer programming with autonomous robots. In Proceedings of the International Conference in Informatics in Secondary Schools—Evolution and Perspectives, ISSEP, Vilnius, Lithuania, 7–11 November 2006. [CrossRef]