SACJ 36(1) July 2024
Research Article

Factors that influence computer programming proficiency in higher education: A case study of Information Technology students

Lerushka Ranjeeth, Indira Padayachee

School of Management, Information Technology & Governance, University of KwaZulu-Natal, Durban, South Africa

ABSTRACT
The industrial world has witnessed an increased demand for computing based skills due to the advent of robotics, artificial intelligence, and analytics. However, the learning of computer programming is challenging and requires an intensive cognitive effort to attain a high degree of skill and expertise. Given this phenomenon, this study undertook to ascertain the factors that influence proficiency in computer programming. The study adopted a quantitative approach and a survey research strategy, guided by a conceptual model. A survey questionnaire was designed as the primary source of data collection. The respondents comprised students who were enrolled for Information Systems and Technology (IT) courses at a higher education institution. The main factors that were identified as significant predictors of computer programming performance were Problem-Solving Ability and Self-Efficacy. The findings contribute to enhancing computer programming pedagogy, which could lead to enhanced student performance in assessment, and validation of the conceptual model.

Keywords Computer programming performance, Intrinsic and extrinsic motivation, Learning styles, Self-efficacy, Problem-solving ability

Categories • **Social & programming topics** ~ **Professional topics** • **Computing Education** ~ **Computer science education/Information Technology/Information systems**

Email
Lerushka Ranjeeth – lerushka07@gmail.com
Indira Padayachee – padayacheei@ukzn.ac.za (CORRESPONDING)

Article history
Received: 12 September 2023
Accepted: 13 May 2024
Online: 31 July 2024

1 INTRODUCTION

The industrial world has witnessed an increased demand for computing based skills, particularly in the domain of computer programming. The demand for computer programming expertise has been elevated by the emergence of the 4th Industrial Revolution (4IR), which

Ranjeeth, L., and Padayachee, I. (2024). Factors that influence computer programming proficiency in higher education: A case study of Information Technology students. *South African Computer Journal* 36(1), 40–75. https://doi.org/10.18489/sacj.v36i1.18819

has compelled members of society to become intelligent users of technology. Technological intelligence is embedded in 4IR systems such as robotics, data analytics and artificial intelligence (AI) by virtue of computer programming logic. The specific demand is for skills that translate to job roles such as business analytics, data analytics, project management, computer programming and testing, systems analysis and design, and database and network administrators (Abdunabi et al., 2019). The demand for these skills is expected to escalate by 13% in the period from 2016 to 2026 (Abdunabi et al., 2019). Central to all of these job portfolios is either a deep or conceptual understanding of computer programming. A multitude of factors, including inherent interest, financial stability and the need for acquiring IT expertise have propelled an increasing number of students to register for technology-oriented courses that provide a substantive exposure to computer programming content (Kori et al., 2015). The rise in student enrolments for technology related courses has, unfortunately, been paralleled by an increase in poor performance and failure in these courses. Students have consistently performed poorly in programming assessments at the higher education level and university courses with programming involved in the curricula have also experienced significantly high dropout rates (Luxton-Reilly et al., 2018). A large number of institutions recorded high rates of failure in introductory programming courses (Konecki & Petrlic, 2014; Peng et al., 2017).

It is within this context that the economic sector requires that graduates who are proficient in computer programming, thereby enhancing their employability and ensuring they add value to the sustained imperative to embrace 4IR technologies. Currently, academic studies have not been conclusive or convergent in their attempts to ensure that cognisance is given to a core set of factors that will allow students to acquire proficiency in computer programming. To address this gap, this article seeks to answer the following main research question: "What are the factors that influence proficiency in computer programming at the higher education level?"

2 LITERATURE REVIEW

Quality in the field of teaching and learning, and specifically e-learning, has become of paramount importance for academic staff and students (Pawlowski, 2007). It has been established that academic performance in computer programming requires a significant cognitive effort from students. However, there are many factors that contribute to this cognitive load and an understanding of these factors is pivotal to ensuring that the failure rates in computer programming modules is reduced. The factors that influence the proficiency of computer programming are diverse and range from demographic variables, such as gender and programming experience, to psychological variables, such as intrinsic and extrinsic motivation to learn computer programming. This constellation of factors provides the basis for the discussion that follows.

2.1 Self-Efficacy in Computer Programming

Self-efficacy (SE) is described as a person's evaluation of their inherent abilities and skills and whether or not their competencies can be used to deliver outcomes that bear a positive effect on their community (Bandura, 1977). According to this definition, SE refers to an individual's confidence in their ability to produce a desirable outcome. Attitude towards the subject matter and SE are among the most important factors in determining one's success in a particular field (D. W. Govender & Basak, 2015). A study involving 83 secondary school students conducted by Kallia and Sentance (2019) established that those students who did not understand the functions of some core programming statements rate lower in self-efficacy as compared to students who understood these statements well. Students who faced difficulties earlier on in their experience with programming are more likely to adopt an overall view of computer programming as being inherently difficult. There is a strong link between SE in problem solving and SE in computer programming, suggesting that students who have confidence in their problem-solving ability tend to perform better in computer programming tasks (I. Govender et al., 2014).

A study conducted with 433 programming students reported a strong correlation between students' programming SE and their sense of satisfaction and interest in the module, which was also linked to students' performance in an introductory programming module (Kanaparan et al., 2019). Teachers should, therefore, pay attention to their students' SE rates/levels because the theory states that the more SE a person has, the more resilient they will be with regard to challenges and obstacles faced in computer programming. A study conducted with 214 computer science students at 3 different universities assessed students' self-assessments when encountering different programming practices such as getting a syntax error or planning (Gorson & O'Rourke, 2020). The study also looked at the students' mental imagery of the competence required to be a professional programmer and found that students who believed that they could not acquire this level of competence had low levels of SE, resulting in poor performance in computer programming assessments.

2.2 Programming Experience

Learning programming is very much about learning by doing. Students who have had previous experience with learning programming, through high school classes or their own independent efforts, perform better in programming courses at university (Kori et al., 2016; Lishinski et al., 2016). Students in one study who had previously taken a programming course, perhaps in high school, had a significantly easier time reading and understanding the programming language as compared to first-time programming students (D. W. Govender & Basak, 2015). It has been demonstrated that students with previous knowledge or engagement with computing programming have a higher level of SE towards computing skills (Kolar et al., 2013). This shows that Programming Experience can positively affect SE as it gives students an idea of what to expect. Such knowledge would increase their confidence going into a computer programming course at a higher education institution. The greater the Programming Experience of students, the higher their level of programming SE (Kittur, 2020). These findings suggest that getting students involved with programming from their schooling years will support their SE in programming-related courses (Kittur, 2020).

2.3 Intrinsic and Extrinsic Motivation

The construct of motivation can be categorised into two broad categories, namely, intrinsic motivation (IM) and extrinsic motivation (EM) (Ryan & Deci, 2000). IM refers to a person who is motivated to do something because they get enjoyment and pleasure from doing that task. EM means that a person is motivated to do something because of the outcome they will receive or to avoid a negative consequence. A study by Gottfried (1985) found that intrinsically motivated students had more academic success than extrinsically motivated students. IM and EM are significant factors that determine performance in computer programming because of the inherent nature of programming itself (Tavares et al., 2017). According to Fang (2012), students who experience excess amounts of difficulty in programming may have low levels of self-efficacy, which also reduces their motivation towards the subject; nevertheless, their motivation can increase with aspects of programming that they do find enjoyable. Most students in this study said that, in particular, they enjoyed learning programming through using robotics as it felt like a fun activity rather than learning a skill (Fang, 2012). It has also been found that students who had previous knowledge and engagement with computer programming displayed more EM than students with no programming experience (Aivaloglou & Hermans, 2019).

In another study (Yacob & Saman, 2012), programming students were found to have two main sources of IM, namely, attitude and setting themselves stimulating goals. The aspect of attitude tended to come from the student's prior experience with programming and whether or not the current programming content that they learnt met their expectations. The extrinsic factors that were found to motivate students were "clear direction, reward and recognition, punishment and social pressure and competition" (Yacob & Saman, 2012, p. 426). Each of these factors were found to positively contribute to students' motivation to engage with the programming content. The effect of gamification on students' motivation and performance in programming was the focus of another study (Khaleel et al., 2019), which found that gamification exploits the EM that all humans naturally possess and uses this motivation to make learning less boring and more satisfying and rewarding. Andriotis (2014) reports a majority (80%) of student respondents as saying that they would enjoy their higher education studies more if game-like elements were included in the courses, and 60% reported that their motivation would increase if their university displayed leader boards as this would encourage more competition between their peers and themselves.

2.4 Problem-Solving Ability

Problem-solving abilities required in mathematics are very similar to the cognitive skills required in computer programming. The syntax of a computer programming language coupled with the semantics of the logical rules and data structures provide the theoretical foundation for problem solving. There is a correlation between problem-solving ability, the mental model of the problem domain and computer programming performance (Lishinski et al., 2016) This finding applied to the advanced aspects of computer programming, where the students had to develop a fully-fledged computer programming solution to a real-life problem. The role played by the syntax and semantics of computer programming code has been explored in a phenomenological study (I. Govender, 2021) which focused on the difficulties that novice programmers face when they learn to program. In order to enhance the mental model visualisation of the problem domain, what is important is a "scaffolding" approach to the teaching of computer programming that entails a strong focus on baseline knowledge involving computer programming language syntax and an inculcation of a deep understanding of data types and structures (I. Govender, 2021). Once these fundamentals have been entrenched, students will be cognitively prepared to engage in incremental learning that involves algorithm development and problem solving. These learning challenges have been documented by various studies: one (Robins et al., 2006) highlighted the difficulties faced in learning looping and arrays; one (Goldman et al., 2008) noted the problems students face with learning inheritance; and one (Garner et al., 2005) alluded to the abstractionism inherent in understanding how a constructor instantiates an object of a class.

Some authors have stressed the importance of problem-solving ability as a crucial factor in enhancing algorithmic thinking capacity (Malik et al., 2019; Romero et al., 2017). Algorithmic or computational thinking has been defined as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Wing, 2008, p. 3717). A study conducted with 113 Information Technology and Engineering students at a university found that, when students spent additional time on improving problem-solving skills, this had a positive effect on their perceived ability to learn programming (Lawan et al., 2019). Another study (Kinnunen, 2009), comprising interviews with five computer science lecturers at a university about the reasons for students' poor performance in programming courses, found that all five agreed that students tend to lack the fundamental skills needed for analysing and solving a problem. A preliminary study (Bain & Barnes, 2014), which questioned students on the challenges experienced when learning how to write computer programming code, demonstrated that 50% of the students did not have a strategy for dealing with problems that arose while writing the code. The main method of trying to solve the problem was to turn to Internet searches. It was also found that 53% of students did not understand how different programming concepts and elements of code related to the bigger picture and how small sections of programming topics connected with others to form a whole solution to a problem. The study concluded that the fundamental issue with learning programming was inadequate problem-solving methods and a lack of critical thinking. It has been recommended that, before students begin to engage in computer programming assessment tasks, they should have foundational knowledge of problem-solving strategies and procedures (Loksa et al., 2016).

2.5 Deep and Surface Learning

The approach that students adopt towards learning has been a topic of extensive research pertaining to higher education (Lonka et al., 2004; Trigwell et al., 1999; Vanthournout et al., 2013).

A study (Marton & Säljö, 1976) introducing the concepts of a deep learning and a surface learning approach found that students who tried to obtain a genuine understanding of the academic material had a deep approach to learning. This type of student attaches personal value to the concepts and knowledge gained in class. Students who, on the other hand, use memorisation and rote learning techniques rather than understanding to pass tests and examinations are said to adopt a surface learning approach (Spada & Moneta, 2012). Students who have a surface approach to learning may be able to pass and even excel in a subject; however, their learning style is appropriate only in test and examination situations where they are required simply to reproduce information and, in situations where they are required to apply this information in a practical way, they usually fall short. Students using a deep approach to learning apply critical thinking skills, thereby enabling them to make connections between different concepts more easily (Lindblom-Ylänne et al., 2019). There is a link between the surface learning approach and SE: students with low SE, low motivation to study and negative beliefs about studying tended to use the surface approach to learning (Lindblom-Ylänne et al., 2019). Students who participate more in class activities and adopt a positive attitude towards computer programming will tend to engage in more deep learning techniques (Floyd et al., 2009). In a study that entailed the interviewing of 177 university students who enrolled in a programming module, it was established that students who scored high on deep learning attributes also achieved high marks for their computer programming module (Fincher et al., 2006).

Students can be encouraged to engage in deep learning strategies by being assessed through project work instead of only being assessed through written examinations (Peng et al., 2017) Moreover, through project work, educators can better monitor and aid students to identify their weaker areas more efficiently (Peng et al., 2017). There is agreement that programming needs to encompass both deep and surface learning approaches because of the fact that programming is more of a skill than knowledge (Konecki & Petrlic, 2014). Teaching students problem-solving skills and strategies will encourage students to adopt deep learning techniques because it is the ability to analyse a problem and converge on a solution that comprises a deep learning approach (Malik et al., 2019). One study (Ranjeeth, 2011) found that 50% of computer programming students at higher education institutions tend to adopt a surface learning approach for computer programming in introductory courses. The researcher suggests that students tend to adopt this style of learning to meet the course requirements and to be able to obtain a pass mark for programming assessments. Hence, the adoption of deep and surface learning in computer programming does become a factor that needs to be examined in greater detail in terms of its influence on students' performance in computer programming assessments.

The literature review has been designed to provide a comprehensive coverage of the main topics that prevail in this domain of study. The broad classification of topics, namely self-efficacy, programming experience, intrinsic and extrinsic motivation, problem-solving ability and learning styles, covered in the literature review led to the development of the conceptual model illustrated in Figure 1 to guide the data collection phase of the current study. The next section covers the methodology adopted for the study.

3 METHODOLOGY

A quantitative approach was adopted for this study (Saunders et al., 2009) This decision was based on the observation that many of the correlation-based studies on the factors that influence academic performance in computer programming have been conducted using a quantitative approach and a survey type of methodology.

3.1 Conceptual Model

The study's conceptual model depicted in Figure 1 was constructed on the basis of the factors that have been hypothesised to influence performance in computer programming. The factors identified in Figure 1 were adopted from a range of studies (Bandura, 1977; Fang, 2012; Gottfried, 1985; Kori et al., 2016; Lishinski et al., 2016; Spada & Moneta, 2012).

In Figure 1, the independent variables are Programming Experience, Problem-Solving Ability, Learning Styles (Deep and Surface) and Intrinsic and Extrinsic Motivation. According to the literature reviewed, Programming Experience and Problem-Solving Ability have a direct influence on a student's SE regarding computer programming, and this manifests in a student's ability to learn computer programming. The Learning Style adopted by a student in terms of deep and surface learning also has a direct influence on academic performance in computer programming, as do IM and EM. According to Yacob and Saman (2012) both Intrinsic and Extrinsic Motivation have a positive relationship to the learning of computer programming. While it has been established that Programming Experience and Problem-Solving Ability have an influence on performance in computer programming, these influences are mediated by SE.

The dependent variable in the study is the students' proficiency/academic performance in computer programming. This variable was measured by obtaining a self-assessment-based rating of students' performance in computer programming assessments. The students in the Information Systems and Technology (IS&T) discipline at the University of KwaZulu-Natal (UKZN) undertake formal practical assessments where they are required to use their programming skills to display their proficiency in computer programming and provide a successful solution for the task given to them. It was envisaged that the mark obtained by the students

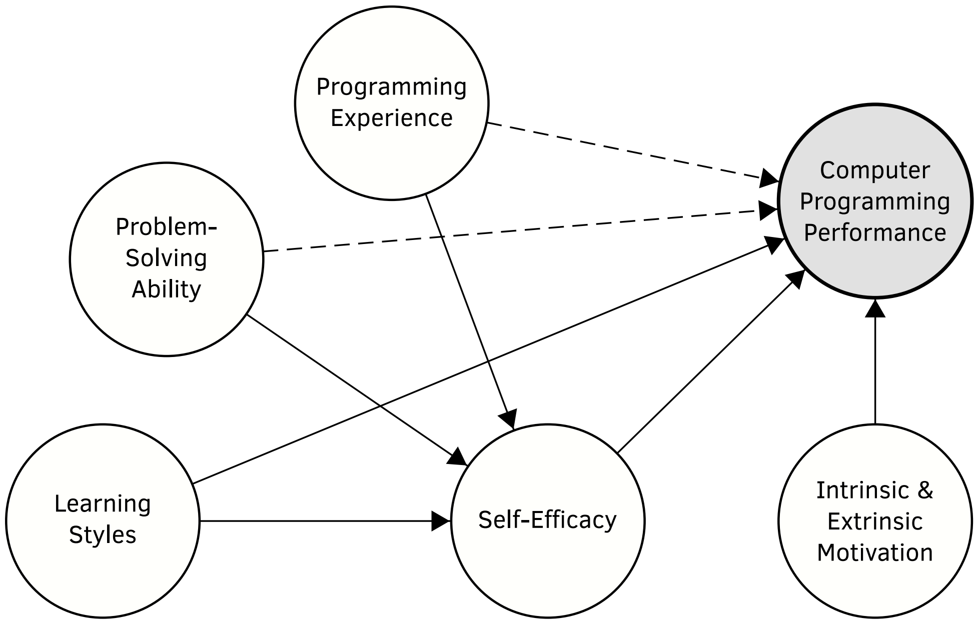


Figure 1: Conceptual model

would provide a guideline to enable the students to rate their individual performance in computer programming assessment. This self-reported rating is used as an indicator of the students' academic performance in computer programming. The strategy of using practical computer programming assessment activity as an indicator of proficiency in computer programming has been used in studies with a similar purpose to that of the current study (Bennedsen et al., 2007; Edwards et al., 2019). SE in computer programming is theoretically linked to a students' background and previous exposure to programming as well as their background in mathematics and problem solving (Abdunabi et al., 2019). A student's level of SE is also related to their learning style because students with higher SE are more likely to adopt a deep learning style as they tend to find the subject inherently interesting. Finally, the overall combination of each of these factors, SE, Learning Styles, Programming Experience, Problem-Solving Ability and Motivation is envisaged to result in a student's achieving higher marks in programming tests and examinations. The results of tests, and examinations then feed back into their SE, if they have performed well in a test or examination, and this will work to increase their belief in themselves and their programming abilities, which then results in them consistently performing well on tests and examinations. Both IM and EM have a positive relationship to the learning of computer programming (Yacob & Saman, 2012). Also, students who find the subject more enjoyable will develop both IM and EM to work on programming tasks, thereby ensuring that they are adequately prepared for examinations and assignments pertaining to computer programming.

3.2 Study Design

The site for the study was the Pietermaritzburg and Westville campuses of UKZN. Due to the adoption by the university of online learning, the launch of the study questionnaire was conducted during online lecture and practical sessions on the Microsoft Teams (3rd year, Honours and Masters programmes) and Zoom (2nd year) video conferencing platforms.

The population for the study comprised all Information System and Technology (IS&T) students that were enrolled for a computer programming module. The population consisted of 2nd and 3rd year IS&T students, as well as Honours and coursework Masters students. The total population of the study was 420 university students from the IS&T discipline. A census approach was adopted where the sample size chosen for the study was also the total population of the study, which was 420 students.

This study employed a structured, survey questionnaire as the primary source of data for the study's empirical analysis (Sekaran & Bougie, 2016). The questionnaire was designed to resonate with the study's conceptual model. Construct validity refers to the alignment between the constructs of the study (which are referred to as unobservable variables specified at a conceptual level) and the questionnaire items that are used to obtain a tangible measure of that construct (Peter, 1981). A viable strategy to ensure construct validity is to align questionnaire items to previous studies where these constructs and items have been validated. The study's main constructs were subjected to theoretical validation by using previous research efforts with a similar objective and also included constructs identified in the study's conceptual model. Table 1 provides a summary of the sources of the questions measuring the constructs in the questionnaire.

Table 1: Summary of the sources of questionnaire items

Construct	**Reference**	**#items**
Self-Efficacy	Askar and Davenport (2009)	12 items
Learning Styles	Mahatanankoon and Wolf (2021)	6 items
Motivation	Amabile et al. (1994)	8 items
Problem-Solving Ability	Tukiainen and Mönkkönen (2002)	10 items

The questionnaire was discussed with academics involved in the teaching of computer programming in the IS&T discipline at UKZN. Comments and suggestions were then incorporated into the questionnaire. The questionnaire was piloted with two IS&T Masters students and two students from the IS&T Honours class.

The questionnaire was launched during formal online lectures by the academic staff members who were lecturing in the 2nd year, 3rd year, Honours and Masters programmes. Students were informed of the requirements regarding the questionnaire and were provided with an opportunity to complete it during the computer programming practical sessions. The questionnaire was made available as an online survey. Students were required to answer programming related tasks to demonstrate their comprehension of conditional, logical and data structure-oriented problems during their computer programming practical sessions at the University. Students were given the latitude of completing these questions without any time restrictions.

Ethical clearance and gatekeeper applications were obtained prior to the data collection phase. In terms of the survey protocol, the study respondents were informed of their voluntary participation in the study and in compliance with the Personal Protection of Information (POPI) Act, no personal information was collected that could be used to directly identify the study's respondents. The anonymous nature of the survey meant that computer programming performance would be a self-assessment rating, which was useful for estimating the construct of academic performance. This self-reported measure of academic performance in computer programming was validated against the respondents' Problem-Solving Ability in the context of computer programming tasks.

The two main statistical methods used were descriptive and inferential statistical analysis. The descriptive statistics used comprised measures of central tendency (mean and median), measures of variability (standard deviation) and frequency distribution. The descriptive results are displayed by stacked bar graphs and histograms. These data visualisation techniques were used to provide an overall view of the empirical evidence with regard to the study's main constructs such as Programming Experience, Problem-Solving Ability, SE and Computer Programming Performance in a formal computer programming assessment. The routine check for data reliability was conducted via the Cronbach alpha test. The inferential statistics used were the one sample t-test, tests of normality, the Pearson Correlation Co-efficient and multiple regression analysis.

4 RESULTS AND DISCUSSION

The primary data collection instrument consisted of a questionnaire that comprised two main sections. The first section (labelled Section A) consisted of demographic questions and questions pertaining to levels of experience in the domain of computer programming and systems analysis and design. The second section of the questionnaire comprised the core aspects that addressed the main objectives of the study (see Table 2). There were 133 valid responses received, constituting a response rate of 32% .

Table 2: Second section of the questionnaire

Section label	**Response type**	**Concept**	**#items**
Part One	Likert scale	Intrinsic and Extrinsic motivation	8
Part Two	Likert scale	Learning styles	6
Part Three	Likert scale	Self-Efficacy	12
Part Four	MCQ	Problem-Solving Ability and Computing Mental Model	10

4.1 Demographic and Background Information of Participants

Section A of the questionnaire was designed predominantly to obtain demographic and background information from the study respondents. The demographic data pertaining to the level of study is presented in Figure 2.

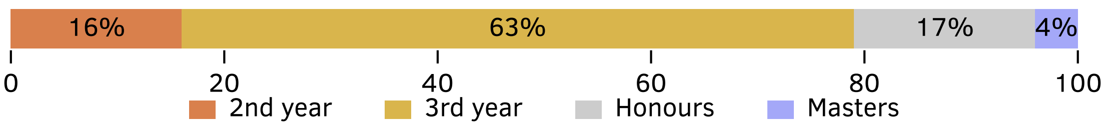


Figure 2: Level of study

Figure 2 illustrates that the majority of the respondents (63.16%) were in the 3rd level of undergraduate study, followed by Honours level (16.54%), 2nd level of undergraduate study (15.79%) and Masters (4.51%) level of study. First year students registered for IS&T modules were not included as part of the target population as students in their 2nd year of undergraduate studies may enroll for the module focusing on Introductory Programming for Information Systems. The largest group of the respondents (79.7%) comprising 3rd year and Honours students were in their respective exit levels, which would give an indication of their preparedness for employment in the computer programming sector.

The academic college of affiliation of the study's respondents is presented in Figure 3 showing that 64% of the respondents were from the College of Law and Management Studies (LMS) and 36% from the College of Agriculture, Engineering and Science (AES).

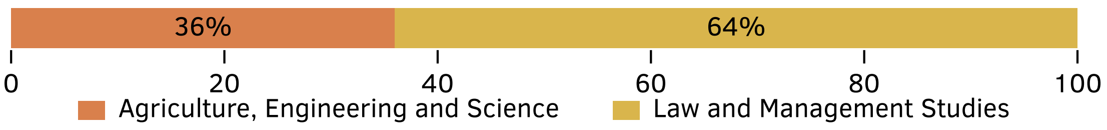


Figure 3: College of Affiliation

A summarised view showing an approximation of the years of computer programming experience acquired by the study's respondents is provided in Figure 4. The majority of the study's respondents were affiliated to the College of Law and Management Studies (LMS), and given the IS&T curriculum specifications, students from the College of Agriculture, Engineering and Science (AES) will in all probability have greater previous experience of computer programming engagement.

The majority of respondents, that is 44.4% had between 0- and 2-years' experience, while 31.6% had between 2- and 3-years' experience, followed by 18% who had between 3- and 4-years' experience and lastly 6% who had more than 4 years of experience in computer programming. These results are consistent with the College of Affiliation results, as AES students undertake computer programming in the first year of their undergraduate curriculum whereas

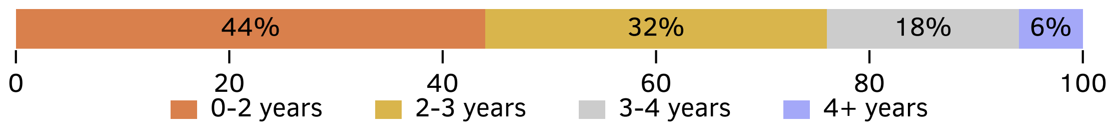


Figure 4: Computer programming experience

BCOM students undertake an introductory programming module in the second year of their undergraduate studies. Respondents that reported more than 4 years of computer programming experience would include students that have computer programming exposure at high school level and students undertaking the Masters qualification.

4.2 Tests of Normality

The study's main constructs were subjected to the Shapiro-Wilk (SW) and Kolmogorov-Smirnov (KS) tests of normality and are presented in Table 3.

Table 3: Tests for normality

	Kolmogorov-Smirnov*			**Shapiro-Wilk**		
	Statistic	df	Sig.	Statistic	df	Sig.
Motivation Composite	0.112	133	0.000	0.952	133	0.000
LS Composite	0.095	133	0.005	0.980	133	0.045
SE Composite	0.069	133	0.200†	0.987	133	0.233

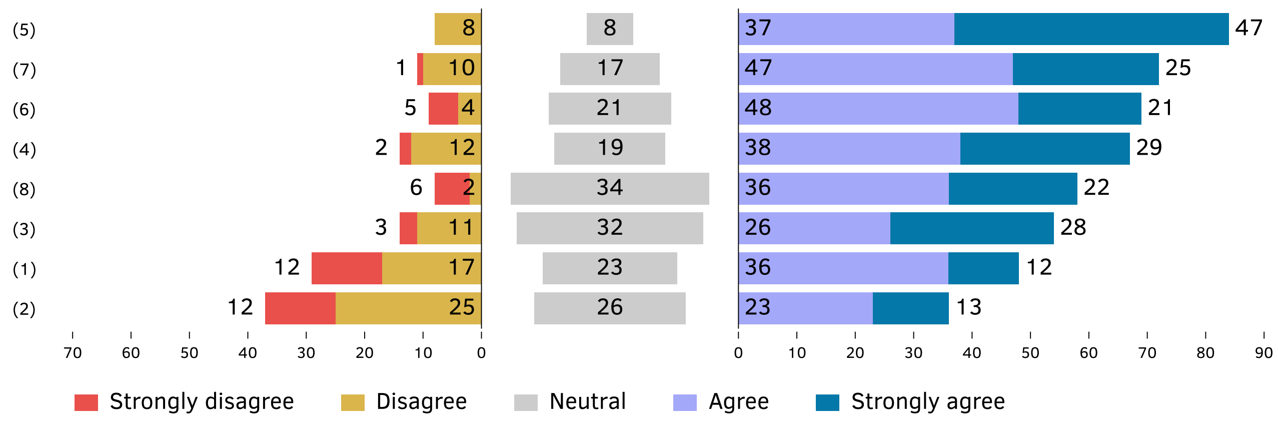
* Lilliefors significance correction
† this is the lower bound of the true significance

When it comes to normality testing, the null hypothesis states that the sampling distributions are *not* normal. As can be observed in Table 3 the constructs of Motivation and Learning Styles (LS) pass the test for normality (null hypothesis rejected, $p < 0.05$). However, the construct of Self-efficacy (SE) fails the test of normality (null hypothesis accepted, ($p > 0.05$) because the probability that the sampling distribution is not normal is quite high.

4.3 Conceptual Model and Empirical Findings

4.3.1 Motivation to Learn Computer Programming

The construct of Motivation (to learn computer programming) has been represented by 8 questionnaire items where 5 represent intrinsic motivation (IM) and 3 represent extrinsic motivation (EM). An overall presentation of the responses is provided in Figure 5.



(1) IM1: I prefer course material that really challenges me so I can learn new things and understand how they work
(2) IM2: When I don't understand something right away I try to figure it out by myself
(3) IM3: I prefer course material that arouses my curiosity even if it is difficult to learn
(4) IM4: Getting good marks for programming brings me a sense of personal satisfaction
(5) IM5: I engage with new technology so that I have a sense of control over the technology
(6) EM1: I want to do well in my programming modules because it is important to show my ability to my family, friends and lecturers
(7) EM2: I engage with new technology because that is what society expects of me
(8) EM3: I make an effort to master computer programming so that I can "fit in" with other students in my group/class

Figure 5: Overall view of responses for the construct of Motivation (sorted by positive responses)

Table 4: Measures of central tendency for Motivation

Motivation	N Valid	N Missing	Mean	Median	Mode	Standard Deviation
IM1: I prefer course material that really challenges me so I can learn new things	133	0	3.67	4	4	1.029
IM2: When I don't understand something right away I try to figure it out by myself	133	0	3.87	4	4	0.900
IM3: I prefer course material that arouses my curiosity even if it is difficult to learn	133	0	3.77	4	4	1.000
IM4: Getting good marks for programming brings me a sense of personal satisfaction	133	0	4.23	4	5	0.900
IM5: I engage with new technology so that I have a sense of control over the technology	133	0	3.84	4	4	1.006
EM1: I want to do well in my programming modules because it is important to show my ability to my family, friends and lecturers	133	0	3.65	4	3	1.095
EM2: I engage with new technology because that is what society expects of me	133	0	3.02	3	3	1.225
EM3: I make an effort to master computer programming so that I can “fit in” with other students in my group/class	133	0	3.20	3	4	1.209

The measures of central tendency for the Motivation construct are displayed in Table 4.

As can be observed in Table 4, the mean response is in excess of 3 ($M > 3$) and the median is greater than or equal to 3 in all cases ($Mdn \geq 3$). To establish whether the mean and median values are significant measures of central tendency for the dataset shown in Table 4 or whether these values occur by chance, the one sample t-test is used. The one sample t-test may be used to determine if the mean of a sample is significantly different from a hypothesised value (DeCoster & Claypool, 2004). In the context of the current data (Table 4), the null hypothesis is that the mean (parametric) is equal to a hypothesised neutral value of 3 (H_0:M=3) and median (non-parametric) is equal to a hypothesised neutral value of 3 (H_0:$Mdn = 3$). In both cases, the alternate hypothesis is that these measures of central tendency are significantly different from 3 ($H_a \neq 3$). The t-test to establish the significance of the measures of central tendency has revealed results that are identical to the non-parametric equivalent test, which is the one-sampled Wilcoxon signed rank test illustrated in Table A1 in Appendix A.

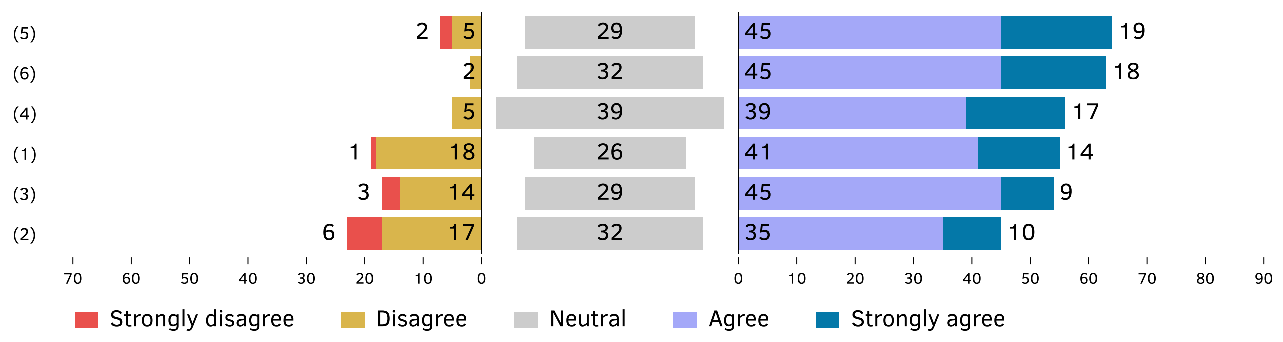
As can be observed in Table A1 in Appendix A, the observed means were significantly greater than the hypothesised means of 3 in 6 of the 8 (75%) questionnaire items. Five of the 6 items were aligned to observable measures of IM. The implication from this analysis is that there is a significant ($p < 0.05$) tendency by the respondents to opt for responses that indicate high levels of IM to learn computer programming. This result indicates that the majority of the study's respondents have IM towards learning and mastering computer programming. A similar conclusion cannot be made when it comes to the EM factors; items 7 and 8 on the questionnaire did not yield a significant ($p > 0.05$) outcome, thereby reducing the prospect of a 95% confidence with conclusions made in terms of EM.

4.3.2 Learning Styles in Computer Programming

The construct of Learning Styles is represented by 6 questionnaire items where 3 of the items were positively worded in favour of a deep learning style and the remaining 3 items were positively worded in favour of a surface learning style. Essentially the learning style component represents students' deep or surface Learning Styles adopted for passing computer programming assessments. An overall presentation of the responses is provided in Figure 6.

The measures of central tendency for the Learning Styles construct are displayed in Table 5. As can be observed in Table 5, the mean response is in excess of 3 ($M > 3$) and the median is greater than or equal to 3 in all cases ($Mdn \geq 3$). To establish whether the mean and median values are significant measures of central tendency for the dataset shown in Table 5 or whether these values occur by chance, the one sample t-test was used. The results of the one sample t-test as well as the Wilcoxon Signed Rank test are shown in Table A2 in Appendix A.

As can be observed in Table A2 in Appendix A. the observed means were significantly greater than the hypothesised mean of 3 for 6 questionnaire items (100%). The implication from this analysis is that there is a significant ($p < 0.05$) tendency by the respondents to opt for responses that indicate high levels of deep learning. Another significant observation is that two of the questionnaire items that were positively worded to indicate surface learning were also significantly ($p < 0.05$) greater than the hypothesised mean of 3. While the responses



(1) LS6: I prefer to ensure that I pass a course even though my understanding of concepts may not be very good
(2) LS5: I find the best way to pass tests is trying to learn the answers to likely questions
(3) LS4: I tend to study best by using memorisation techniques
(4) LS3: I test myself on important topics until I understand them completely
(5) LS2: I find it helpful to study topics in depth rather than trying to remember important facts for tests
(6) LS1: I find most new topics interesting and will often spend extra time trying to understand how they work

Figure 6: Overall view of responses for the construct of Learning Styles (sorted by positive responses)

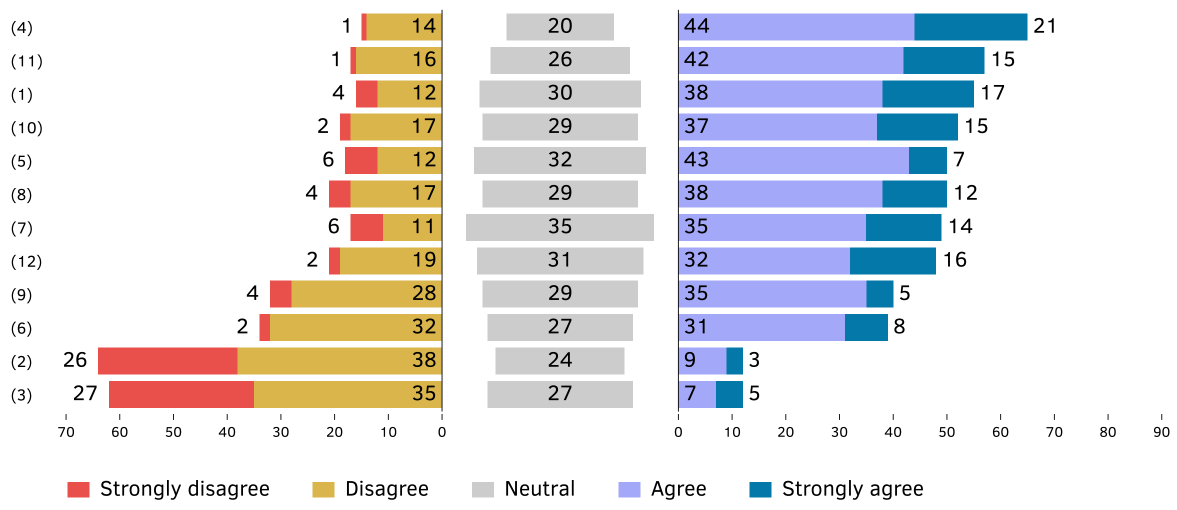
Table 5: Measures of central tendency for Learning Styles

	N					Standard
Learning Styles	Valid	Missing	**Mean**	**Median**	**Mode**	**Deviation**
LS1: I find most new topics interesting and will often spend extra time trying to understand how they work	133	0	3.79	4	4	0.779
LS2: I find it helpful to study topics in depth rather than trying to remember important facts for tests	133	0	3.79	4	4	0.835
LS3: I test myself on important topics until I understand them completely	133	0	3.68	4	4	0.801
LS4: I tend to study best by using memorisation techniques	133	0	3.48	4	4	0.910
LS5: I find the best way to pass tests is trying to learn the answers to likely questions	133	0	3.29	3	4	1.013
LS6: I prefer to ensure that I pass a course even though my understanding of concepts may not be very good	133	0	3.47	4	4	0.989

pertaining to deep learning are indicative of the learning style used to master computer programming, the high scores reported for surface learning are indicative of an intention from the study's respondents to also ensure that they engage in techniques of learning that empower them with a maximum opportunity to pass the computer programming assessment activity.

4.3.3 Self-Efficacy in Computer Programming

The construct of Self-Efficacy (SE) is represented by 12 questionnaire items where 9 of the items were positively worded in favour of high levels of SE and 3 questionnaire items were positively worded in favour of low levels of SE. This construct consisted of questionnaire items that were directed at specific aspects of computer programming. These aspects consisted of: the ability to write procedural and object-oriented code (4 questionnaire items); the ability to debug and recover from errors and the ability to trace through the logic of computer programming code (2 questionnaire items); the ability to compile a logical computer programming solution to a given problem in a specified time range (4 questionnaire items); and the inclination to seek assistance when it comes to the writing of computer programming solutions (2 questionnaire items). An overall presentation of the responses is provided in Figure 7.



(1) SE12: I could manage my time efficiently if I had a pressing deadline on a programming project
(2) SE11: I feel more comfortable to complete a programming problem if someone showed me how to solve the problem first
(3) SE10: I feel that I am better at programming when I get the help of someone else
(4) SE9: I am able to write computer programming code to sort out a given set of numbers into ascending/descending order
(5) SE8: I am confident of my ability to identify the objects in the problem domain and declare, define, and use them
(6) SE7: I could rewrite lengthy and confusing portions of code to be more readable and clearer
(7) SE6: I have a good understanding of the object-oriented paradigm for programming
(8) SE5: I could organize and design my program in a modular/procedural manner
(9) SE4: I am able to mentally trace through the execution of a long, complex program
(10) SE3: I have the capacity to easily identify errors in my programming code
(11) SE2: I am able to construct programming code that is logically correct
(12) SE1: I am confident of my ability to develop suitable strategy for a given programming task in a short time

Figure 7: Overall view of responses for the construct of Self-Efficacy (sorted by positive responses)

The aggregated outcome of the frequency representation for the SE construct is shown in Table 6. As can be observed in Table 6, the mean response is in excess of 3 ($M > 3$) and the median is greater than or equal to 3 in 10 of the 12 cases ($Mdn \geq 3$). In 2 instances, the mean and median response is less than 3. It should be noted that in both of the cases where the mean and median were less than 3, the questionnaire items were phrased positively towards lower levels of SE. The one sample t-test was used to ascertain whether the measures of central tendency were significant or occurred by chance. The results of the one sample t-test as well as the Wilcoxon Signed Rank test are shown in Table A3 (in Appendix A).

From Table A3, it can be observed that there is a significant difference ($p < 0.05$) between the mean and median values for 10 of the 12 items. Questionnaire items 4 and 7 did not yield results that are significant ($p > 0.05$), meaning that there was no significant agreement

Table 6: Measures of central tendency for Self-Efficacy

Self-efficacy	N valid	N missing	Mean	Median	Mode	Standard deviation
SE1: I am confident of my ability to develop suitable strategy for a given programming task in a short time	133	0	3.41	3	4	1.037
SE2: I am able to construct programming code that is logically correct	133	0	3.55	4	4	0.957
SE3: I have the capacity to easily identify errors in my programming code	133	0	3.47	4	4	0.997
SE4: I am able to mentally trace through the execution of a long, complex program	133	0	3.09	3	4	0.981
SE5: I could organize and design my program in a modular/procedural manner	133	0	3.37	3	4	1.026
SE6: I have a good understanding of the object-oriented paradigm for programming	133	0	3.39	3	3	1.043
SE6: I have a good understanding of the object-oriented paradigm for programming	133	0	3.09	3	2	1.011
SE8: I am confident of my ability to identify the objects in the problem domain and declare, define, and use them	133	0	3.32	3	4	0.981
SE9: I am able to write computer programming code to sort out a given set of numbers into ascending/descending order	133	0	3.70	4	4	0.985
SE10: I feel that I am better at programming when I get the help of someone else	133	0	2.27	2	2	1.074
SE11: I feel more comfortable to complete a programming problem if someone showed me how to solve the problem first	133	0	2.25	2	2	1.040
SE12: I could manage my time efficiently if I had a pressing deadline on a programming project	133	0	3.51	4	4	1.027

on the ability to mentally trace through the execution of a long and complex program and to rewrite lengthy and complex portions of code to make them more readable and clearer. This demonstrates that the respondents' self-efficacy did not extend to these two competencies. The questionnaire items that were positively worded in favour of high levels of SE showed a significant positive difference from the hypothesised neutral values for the mean and median. This outcome is indicative of a high level of SE being displayed by the respondents of the study towards the handling of computer programming tasks. The preceding outcome is further corroborated by the negative differences recorded for questionnaire items 10 and 11. These questionnaire items were positively worded to indicate low levels of SE. The low means and medians recorded are an indication that the respondents disagreed with the statements attesting to low levels of SE when it comes to academic performance in computer programming.

4.3.4 Problem-Solving Ability and Performance in Computer Programming

The construct of Problem-Solving Ability was operationalised/measured by adapting the computer programming aptitude test used by Tukiainen and Mönkkönen (2002) in predicting computer programming competence. The test to measure computer programming competence was presented to the study's respondents as a series of problem-solving tasks that tested their cognitive processing ability when faced with computer programming related questions. These tasks were adapted to align with the computer programming content that was delivered to the respondents of the current study during their tenure as students following the IS&T curriculum at UKZN. The classification of questionnaire items used for the Problem-Solving Ability construct is presented in Table 7.

Table 7: Classification of questionnaire items for Problem-Solving Ability

Computer programming concept	**Number of questionnaire items**
Conditional Logic (Logical Operators)	2
Predictive Logic	3
Comparative Logic	1
Iterative Logic	2
Assignment Logic	1
Data Structure Logic	1

Respondents of the study were presented with the set of 10 computer programming related tasks listed in Table 7 and were required to provide a response that was structured as a multiple-choice type of question. Each of the study's respondents were scored on their performance by allocating a point value of 1 for a correct answer and 0 for an incorrect answer.

In this way, each respondent scored a mark out of 10, thereby providing a quantified indicator of the Problem-Solving Ability of the student.

The study's respondents were also required to provide an approximate measure of their academic performance in computer programming assessment. These values were recorded using a scale of 1 to 8. A bivariate correlation analysis was conducted between the respondents' academic performance and their Problem-Solving Ability. The results are presented in Table 8.

As can be seen in Table 8, the Pearson product-moment correlation coefficient (PPMCC) is statistically significant ($r = 0.59$, $N = 133$, $p < 0.01$, two-tailed). The interpretation from this result is that there is a significantly positive relationship between the respondents' academic performance in computer programming assessment and their Problem-Solving Ability in the context of computer programming tasks. This result provides a measure of validity to the construct of academic performance score, which is an estimated value provided by the study's respondents.

4.4 Reliability Testing

For the current study, three constructs were measured using a Likert-scale type of response. The outcome of the Cronbach alpha reliability tests that were conducted on these constructs are presented in Table 9.

Table 8: Academic performance in Computer Programming vs Problem-Solving Ability

		Problem-Solving Ability	**Computer Programming Performance** (numeric)
Problem-Solving Ability	Pearson Correlation	1	0.588*
	Sigma (2-tailed)		0.000
	N	133	133
Computer Programming Performance (numeric)	Pearson Correlation	0.588*	1
	Sigma (2-tailed)	0.000	
	N	133	133

* correlation is significant at the 0.01 level (2-tailed)

Table 9: Cronbach Alpha analysis

Construct	**No of Likert Scale Items**	**Cronbach's alpha**
Motivation (intrinsic and extrinsic)	8 (IM1 to IM5, EM1 to EM3)	0.64
Learning Styles	6 (LS1 to LS6)	0.57
Self-Efficacy	12 (SE1 to SE12)	0.91

Cronbach alpha co-efficient values in the range of 0.7 and above are considered to be good reliability estimates (Sekaran & Bougie, 2016). In Table 9, it can be observed that the Cronbach alpha value for Learning Styles is 0.57 and that of Motivation is 0.64, from which it is inferred that the questionnaire items used to measure these two constructs are not ideally reliable. The constructs were subjected to further validity testing in the form of factor analysis. This is described in the next section.

4.5 Confirmatory Factor Analysis

The process of variable reduction is conducted under the theory that, if the conceptual model does not have an alignment with the study's data, then the conceptual model needs to be rearranged so that it has an optimal alignment with the study's data. This process of fitting the conceptual model to the study's data is referred to as confirmatory factor analysis (CFA), which is a crucial process in ensuring construct validity (Pham et al., 2020).

Three questionnaire items for Motivation and three questionnaire items for Learning Styles identified as significant contributors to the "poor" Cronbach Alpha values in Table 9 were removed from further analysis. In addition, to improve the model fit of the study's conceptual model, the modification indices were examined during CFA and three items that showed high levels of covariance with other items from the SE construct were removed. From an item reliability perspective, the improvement in the internal consistency of the empirical model is confirmed by the reworked Cronbach alpha values shown in Table 10.

Table 10: Re-worked Cronbach Alpha analysis

Construct	**No of Likert Scale Items**	**Cronbach's alpha**
Motivation (intrinsic and extrinsic)	5 (IM1 to IM5)	0.83
Learning Styles	3 (LS1 to LS3)	0.72
Self-Efficacy	9 (SE1 to SE9)	0.93

In Figure 8, the ellipses labelled e4 to e8; e12 to e14 and e15 to e23 refer to the labels of the arrow dumps. The boxes labelled IM1 to IM5, LS1 to LS3 and SE1 to SE9 represent the measured variables. The circles labelled Motivation, Learning Style and Self-Efficacy represent the latent variables, called factors. The item weights in the arrows between the latent variables and the measured variables represents the factor loadings. The double headed arrows between the latent variables Motivation, Learning Style and Self-Efficacy represent correlations

In the context of the current study, the "model fit" indicators arising out of Figure 8 are Comparative Fit Index (CFI) = 0.91, the Tucker Lewis Index (TLI) = 0.9 and the root mean square error of approximation (RMSEA) = 0.082. The measurements for an acceptable model fit are as follows: The CFI index measurement should be closer to 1; the TLI should be in the range of 0,9 to 1; and the RMSEA should be less than 0.08. A RMSEA value that is less than 0.08 and a CFI value of 0.9 or greater are indicators of a good model fit. These results indicate that the empirical model that will be used for the data analysis for the study is not a perfect fit to the study's data but it is closely aligned with the suggested test statistics to ensure a "good fitting" model.

4.6 Correlation Analysis

The main aim of the study was to establish the validity of correlations between the study's main variables as well as the study's conceptual model, illustrated in Figure 1. The course of the correlation analysis is guided by the study by Musil et al. (1998).

4.6.1 Bivariate Correlation Analysis

The data representing the study's main constructs is represented by ordinal scales and the PPMCC may be used to determine the relationship between these constructs (Spada & Moneta, 2012). The Pearson correlation analysis was chosen to establish the significance of relationships between the study's main constructs. It was decided to use the questionnaire items that were positively worded to ascertain levels of deep learning regarding the attainment of good academic performance in computer programming for the generation of the bivariate

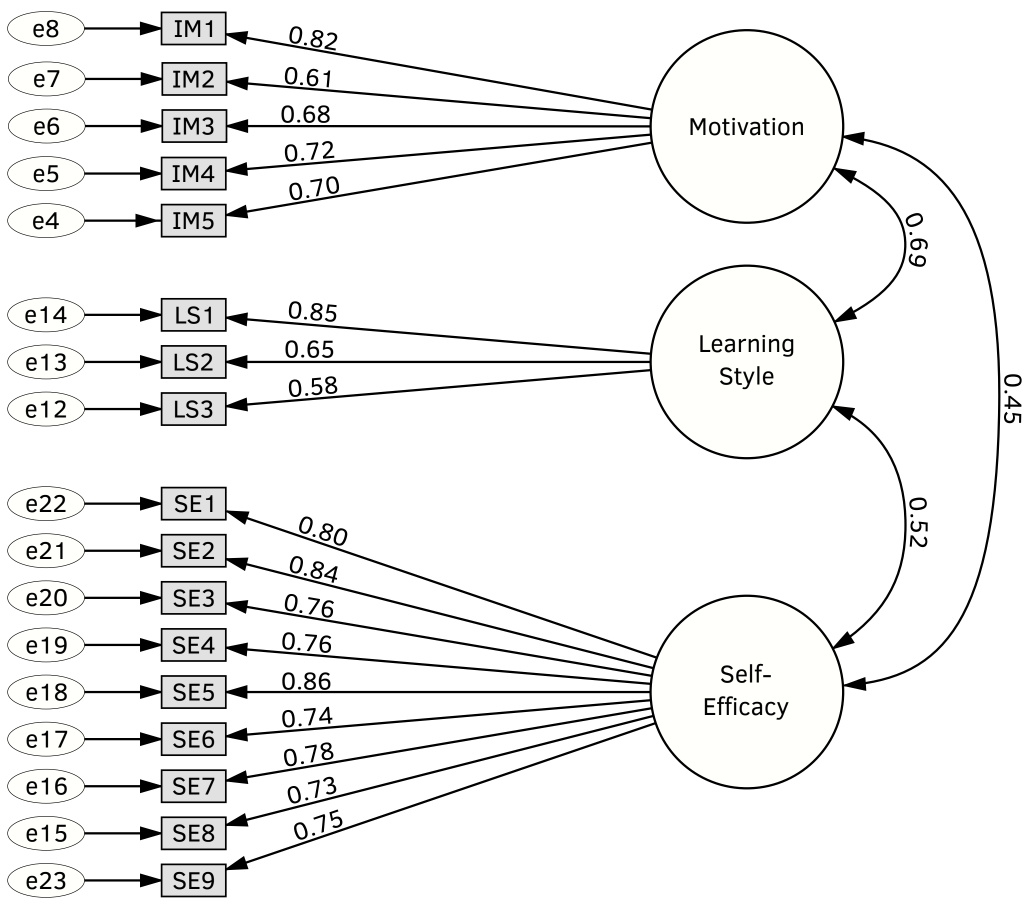


Figure 8: Confirmatory Factor Analysis of the Observed Values

correlation matrix shown in Table 11. From Table 11, the following statistically significant correlations can be observed:

- There exists a moderate positive correlation between Problem-Solving Ability and Computer Programming Performance ($r = 0.59$, $N = 133$, $p < 0.01$, two-tailed)
- There exists a moderate positive correlation between Problem-Solving Ability and Computer Programming Experience ($r = 0.55$ $N = 133$, $p < 0.01$, two-tailed)
- There exists a weak positive correlation between Problem-Solving Ability and Learning Styles (deep learning) ($r = 0.20$ $N = 133$, $p < 0.05$, two-tailed)
- There exists a moderate positive correlation between Problem-Solving Ability and Self-Efficacy ($r = 0.43$, $N = 133$, $p < 0.01$, two-tailed)
- There exists a moderate positive correlation between Programming Experience and Computer Programming Performance ($r = 0.51$, $N = 133$, $p < 0.01$, two-tailed)

- There exists a weak positive correlation between Programming Experience and Learning Styles (deep learning) ($r = 0.23$ $N = 133$, $p < 0.01$, two-tailed)
- There exists a moderate positive correlation between Programming Experience and Self-Efficacy ($r = 0.51$ $N = 133$, $p < 0.01$, two-tailed)
- There exists a weak positive correlation between Learning Styles (deep learning) and Computer Programming Performance ($r = 0.21$ $N = 133$, $p < 0.05$, two-tailed)
- There exists a moderate positive correlation between Learning Styles (deep learning) and Self-Efficacy ($r = 0.42$ $N = 133$, $p < 0.01$, two-tailed)
- There exists a moderate positive correlation between Self-Efficacy and Computer Programming Performance ($r = 0.57$, $N = 133$, $p < 0.01$, two-tailed)
- There exists a moderate positive correlation between Motivation and Learning Styles (deep learning) ($r = 0.51$, $N = 133$, $p < 0.01$, two-tailed)
- There exists a weak positive correlation between Motivation and Self-Efficacy ($r = 0.20$ $N = 133$, $p < 0.05$, two-tailed)

The results in Table 11 indicate that the bivariate correlations between Motivation and Problem-Solving Ability; Motivation and Computer Programming Performance; Motivation and Programming Experience are not significant.

The results from the current study are consistent with those observed in the systematic literature review by Medeiros et al. (2019), who found that in a majority of studies, there is evidence of a positive relationship between programming experience and proficiency in computer programming. The result of the correlation between Self-Efficacy and Computer Programming Performance was supported in a study by I. Govender et al. (2014) where a strong link was established between SE in Problem-Solving Ability and SE in Computer Programming. The correlation between Problem-Solving Ability and Computer Programming Performance is supported by Lishinski et al. (2016), who indicated that Problem-Solving Ability is significantly correlated with good academic performance on programming assignments. The correlation between Problem-Solving Ability and Learning Styles is supported by Malik et al. (2019), who argue that teaching problem-solving skills will inherently promote deep learning techniques among students.

4.6.2 Multiple Regression Analysis (MRA)

The next step after bivariate correlations is to examine the combined effect of multiple independent variables with the dependent variable (Swanson & Holton, 2005). The objective of multiple regression is to provide the researcher with empirical evidence for making decisions regarding the predictive capacity of the study's conceptual model or for enabling an explanation of the relationship between the independent and dependent variables in the study. In

Table 11: Bivariate correlation of the study's main constructs

		Problem-Solving Ability	Computer Programming Performance (numeric)	Programming Experience	Learning Style Deep	Self-Efficacy	Motivation
Problem-Solving Ability	N	133	133	133	133	133	133
	Pearson Correlation	1	0.588*	0.553*	0.202†	0.434*	0.002
	Sigma (2-tailed)		0.000	0.000	0.020	0.000	0.986
Computer Programming Performance (numeric)	Pearson Correlation	0.588*	1	0.506*	0.214†	0.572*	0.130
	Sigma (2-tailed)	0.000		0.000	0.014	0.000	0.137
Programming Experience	Pearson Correlation	0.553*	0.506*	1	0.235*	0.512*	0.131
	Sigma (2-tailed)	0.000	0.000		0.006	0.000	0.134
Learning Style Deep	Pearson Correlation	0.202†	0.214†	0.235*	1	0.421*	0.505*
	Sigma (2-tailed)	0.020	0.014	0.006		0.000	0.000
Self-Efficacy	Pearson Correlation	0.434*	0.572*	0.512*	0.421*	1	0.202†
	Sigma (2-tailed)	0.000	0.000	0.000	0.000		0.020
Motivation	Pearson Correlation	0.002	0.130	0.131	0.505*	0.202†	1
	Sigma (2-tailed)	0.986	0.137	0.134	0.000	0.020	

* correlation is significant at the 0.01 level (2-tailed)
† correlation is significant at the 0.05 level (2-tailed)

the context of the data for the current study, the multiple regression model is guided by Pham et al. (2020). The first output from this analysis is the Model Summary output, shown in Table 12. The analysis of variance (ANOVA) output is shown in Table 13.

Table 12: Model summary for MRA

Model	R	R^2	Adjusted R^2	Std error of estimate
1	0.697*	0.485	0.465	0.817

* Predictors: (Constant), MotivationCompositeMean, Problem-Solving Ability, SECompositeMean, LS_Deep, ProgExperience

By analysing Tables 12 and 13, it can be established that the combined independent variables significantly predict Computer Programming Performance. The statistics that support

Table 13: ANOVA* showing the significance of the model

Model		Sum of squares	df	Mean^2	F	Significance
1	Regression	79.976	5	15.995	23.966	<0.001 †
	Residual	84.761	127	0.667		
	Total	164.737	132			

* Dependent Variable: Computer Programming Performance (numeric)
† Predictors: (Constant), MotivationCompositeMean, Problem-Solving Ability, SECompositeMean, LS_Deep, ProgExperience

this conclusion are listed below:

- The F-statistic in Table 13 ($F(5) = 23.96$ ($p < 0.01$)) is an indicator of the significance of the study's multiple regression model.
- The F-statistic provides a validation indicator for the conclusion that the composite set of independent variables explains 46.5% of the variance in the dependent variable ($R = 0.7$; $R^2 = 0.485$; adjusted $R^2 = 0.465$; $p < 0.01$) as indicated in Table 12.

The final multiple regression output is to examine the coefficient values listed in Table 14, where it can be seen from the Beta values that Problem-Solving Ability and SE are the two main predictors of Computer Programming Performance ($p < 0.01$). The findings on SE concurs with the study conducted by I. Govender et al. (2014), where a strong link was established between SE in problem solving and SE in computer programming. The current study extends the network of influence regarding SE by showing a moderate, positive correlation between SE and a deep learning style. The current study also shows a moderate positive correlation between SE and Programming Experience, which is supported by Kolar et al. (2013). Problem-Solving Ability shows a moderate positive correlation with Computer Programming Performance. This outcome is confirmed in the report compiled by Medeiros et al. (2019), where 26 papers on this topic were reviewed, as well as findings reported by Mahatanankoon and Wolf (2021), and Lonka et al. (2004).

It should also be noted that, according to the data presented in Table 14, Programming Experience, Learning Styles and Motivation are not significant predictors of Computer Programming Performance. This finding was supported by Bennedsen et al. (2007), who found that students with programming experience did not outperform students who did not have programming computer programming experience as they relied heavily on their past knowledge and fell behind with the course material. The finding on deep Learning Styles being a peripheral influence on computer programming achievement may be attributed to the fact that students need to embrace both surface and deep learning styles because of the skill-based nature of programming (Lindblom-Ylänne et al., 2019), as well as the finding that 50% of computer programming students at higher education institutions tend to adopt a surface learning style for introductory computer programming courses (Ranjeeth, 2011).

Table 14: Coefficients of the Multiple Regression Model* showing the significance of the model

		Unstandardised Coefficients		**Standardised Coefficients**		
Model		**B**	**Std. error**	**Beta**	**t**	**Sig.**
1	(constant)	0.974	0.513		1.897	0.060
	ProgExperience	0.132	0.093	0.117	1.418	0.159
	Problem-Solving Ability	0.022	0.004	0.382	4.837	0.000
	SECompositeMean	0.577	0.128	0.366	4.524	0.000
	LS_Deep	-0.152	0.139	-0.088	-1.100	0.273
	MotivationCompositeMean	0.148	0.131	0.084	1.130	0.261

* Dependent Variable: Computer Programming Performance (numeric)

The construct of Motivation played a minimal role in predicting Computer Programming Performance. This outcome is contrary to the results reported by Bergin and Reilly (2005), who found that IM and EM were strongly aligned to Computer Programming Performance. The finding on Motivation not being a significant predictor of Computer Programming Performance can be attributed to the fact that Motivation can be negatively affected by the need to do challenging programming exercises and to expend a great deal of effort in grasping programming concepts (Durak et al., 2019).

5 CONCLUSION AND RECOMMENDATIONS

This study was aimed at addressing the issue of students struggling to obtain proficiency in the domain of computer programming. There have been numerous studies previously that have studied this phenomenon and knowledge around this topic has grown substantially. The problem of poor performance in computer programming does, however, continue to exist. The current study was grounded in the previous efforts to find a solution to this phenomenon. The difference, however, is that this study adopted a multidimensional approach by integrating five significant constructs into a single conceptual model, examining the role of each construct in relation to academic performance in computer programming. This study also showed the relative importance of each factor in contributing towards an improvement in students' performance in computer programming.

The study's findings relating to the main research question, "What are the factors that influence proficiency in computer programming at the higher education level?", are as follows:

Programming Experience and Computer Programming The bivariate correlation shows a moderate but significant, positive correlation with Computer Programming Performance.

This outcome suggests that Programming Experience does influence academic performance in computer programming. The knowledge obtained from the current study regarding Programming Experience is crucial because the implication is that, when Programming Experience is considered as part of a broader understanding of the factors that influence Computer Programming Performance, its significance is minimal as indicated in the Multiple Regression Model in Table 14.

Problem-Solving Ability and Computer Programming The bivariate correlational analysis shows that Problem-Solving Ability has a moderate, significant positive correlation with Computer Programming Performance. The multiple regression analysis indicates that Problem-Solving Ability is a main predictor of Computer Programming Performance. The implication from these results suggest that universities need to invest more time at 1st-year level with a focus on logical reasoning and algorithmic thinking so that students can obtain foundation knowledge on computer programming semantic structures to enhance problem solving. This observation has significant implications for students who have not had prior experience in computer programming because a focus on algorithmic thinking would equip them with the cognitive structures required to obtain a deep understanding of computer programming logic.

Self-Efficacy (SE) and Computer Programming The bivariate correlational analysis shows there is a moderate but significant, positive correlation between SE and Computer Programming Performance. The multiple regression analysis indicates that SE is a main predictor of Computer Programming Performance. The current study also confirms a moderate, significant positive correlation between SE and Programming Experience. The current study extends the network of influence regarding SE by observing that there is a moderate, significant positive correlation between SE and a deep learning style. These observations are significant from a pedagogical perspective because educators should make a concerted effort to enhance and enable high levels of SE amongst students in their programming courses.

Motivation and Computer Programming The construct of Motivation played a minimal role in predicting Computer Programming Performance. While this construct had a weak but positive correlation with SE and a moderate positive correlation with Learning Styles, it did not display a significant relationship with Problem-Solving Ability, Programming Experience or Computer Programming Performance.

Learning styles and Computer Programming The study shows that a weak positive correlation exists between Problem-Solving Ability and Learning Styles (deep learning) but contributes to Computer Programming Performance only peripherally.

The Model Summary output and the analysis of variance (ANOVA) output in Tables 12 and 13 demonstrated that the combined independent variables (Motivation, Problem-Solving

Ability, Self-Efficacy, Programming Experience and Learning Styles) significantly predict Computer Programming Performance.

The main limitation of the study is the threat to external validity because a larger, more expansive sample would have created an opportunity for greater generalisation of the study's results. The delimitation of confining the study to IS&T students was necessitated by the researcher's concerns when it came to data collection because, at the commencement of the study, the COVID-19 pandemic prevented free and open communication with potential respondents. The researcher's field study was confined to IS&T platforms that were made available online. Another limitation of the study was the potential breach of internal validity because the measurement of student programming performance was done through estimates provided by the study respondents. This potential weakness in the study was, however, mitigated by the inclusion of problem-solving tasks in the study's questionnaire. The strong positive correlation between the scores obtained in the problem- solving tasks and the respondents' estimation of their performance in computer programming assessment enhanced the reliability of these variables.

The findings contribute to the body of knowledge in computer programming pedagogy, which could lead to improved student performance in assessment tasks, as well as to validating the adopted conceptual model. The findings emphasise the role played by Self-Efficacy as a significant predictor of Computer Programming Performance as well as its significant, positive correlations with the four major factors, namely Problem-Solving Ability, Programming Experience, Learning Styles (deep learning) and Motivation. The finding on Problem-Solving Ability as a significant predictor of Computer Programming Performance contributes to the wider argument on the effects of problem solving on computer programming performance and vice versa and its potential to improve cognitive skills such as creative thinking, mathematical skills, and reasoning, thereby promoting computational thinking skills in education and society. The findings of the adopted conceptual model demonstrate the importance of the many factors that have direct or indirect effects on student performance in computer programming courses, including Problem-Solving Ability, Self-Efficacy, Programming Experience, Motivation and Learning Styles. The findings of the study have implications for educational practice as the personalized learning instructional approach and multiple learning opportunities can be used to improve individual student performance in computer programming.

A further outcome of the study is the development and validation of a conceptual model to predict Computer Programming Performance. This model has been subjected to validity testing in the form of confirmatory factor analysis and multiple regression analysis. It is recommended that future studies explore the role that Motivation and Learning Styles play when students learn programming in an online distance learning environment as opposed to a face-to-face setting or a hybrid learning environment.

References

Abdunabi, R., Hbaci, I., & Ku, H. (2019). Towards enhancing programming self-efficacy perceptions among undergraduate information systems students. *Journal of Information Technology Education: Research*, *18*, 185–206. https://doi.org/10.28945/4308

Aivaloglou, E., & Hermans, F. (2019). Early programming education and career orientation: The effects of gender, self-efficacy, motivation and stereotypes. *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 679–685. https://doi.org/10.1145/3287324.3287358

Amabile, T. M., Hill, K. G., Hennessey, B. A., & Tighe, E. M. (1994). The work preference inventory: Assessing intrinsic and extrinsic motivational orientations. *Journal of Personality and Social Psychology*, *66*(5), 950–967. https://doi.org/10.1037/0022-3514.66.5.950

Andriotis, N. (2014). 2014 gamification survey result. https://www.talentlms.com/blog/gamification-survey-results-2014/

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java programming among engineering students. *Turkish Online Journal of Educational Technology*, *8*(1), 1303–6521. https://eric.ed.gov/?id=ED503900

Bain, G., & Barnes, I. (2014). Why is programming so hard to learn? *Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference (ITICSE 2014)*, 356. https://doi.org/10.1145/2591708.2602675

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*(2), 191–215. https://doi.org/10.1037/0033-295X.84.2.191

Bennedsen, J., Caspersen, M. E., & Allé, F. (2007). Assessing process and product. *Innovation in Teaching and Learning in Information and Computer Sciences*, *6*(4), 183–202. https://doi.org/10.11120/ital.2007.06040183

Bergin, S., & Reilly, R. (2005). The influence of motivation and comfort-level on learning to program. *17th Workshop of the Psychology of Programming Interest Group*, 293–304. https://www.ppig.org/papers/2005-ppig-17th-bergin/

DeCoster, J., & Claypool, H. (2004). Data analysis in SPSS [Accessed 31 May 2024]. http://www.stat-help.com/notes.html

Durak, H. Y., Yilmaz, F. G. K., & Bartin, R. Y. (2019). Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities. *Contemporary Educational Technology*, *10*(2), 173–197. https://doi.org/10.30935/CET.554493

Edwards, S. H., Murali, K. P., & Kazerouni, A. M. (2019). The relationship between voluntary practice of short programming exercises and exam performance. *Proceedings of the ACM Conference on Global Computing Education (CompEd 2019)*, 113–119. https://doi.org/10.1145/3300115.3309525

Fang, X. (2012). Application of the participatory method to the computer fundamentals course. *Advances in Intelligent and Soft Computing*, *137*, 185–189. https://doi.org/10.1007/978-3-642-27866-2_23

Fincher, S., Robins, A., Baker, B., Cutts, Q., Haden, P., Hamilton, M., Petre, M., Tolhurst, D., Box, I., de Raadt, M., Hamer, J., Lister, R., Sutton, K., & Tutty, J. (2006). Predictors of success in a first programming course. *8th Australasian Computing Education Conference (ACE 2006)*, *52*, 189–196. https://research.usq.edu.au/item/9y1qv/predictors-of-success-in-a-first-programming-course

Floyd, K. S., Harrington, S. J., & Santiago, J. (2009). The effect of engagement and perceived course value on deep and surface learning strategies. *Informing Science: The International Journal of an Emerging Transdiscipline*, *12*, 181–190. https://doi.org/10.28945/435

Garner, S., Haden, P., & Robins, A. (2005). My program is correct but it doesn't run: A preliminary investigation of novice programmers' problems. *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, 173–180. https://doi.org/10.28945/435

Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., Loui, M. C., & Zilles, C. (2008). Identifying important and difficult concepts in introductory computing courses using a delphi process. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, 256–260. https://doi.org/10.1145/1352135.1352226

Gorson, J., & O'Rourke, E. (2020). Why do CS1 students think they're bad at programming? Investigating self-efficacy and self-assessments at three universities. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 170–181. https://doi.org/10.1145/3372782.3406273

Gottfried, A. E. (1985). Academic intrinsic motivation in elementary and junior high school students. *Journal of Educational Psychology*, *77*(6), 631–645. https://doi.org/10.1037/0022-0663.77.6.631

Govender, D. W., & Basak, S. K. (2015). An investigation of factors related to self-efficacy for Java programming among computer science education students. *Journal of Governance and Regulation*, *4*(4), 612–619. https://virtusinterpress.org/IMG/pdf/10-22495_jgr_v4_i4_c5_p6.pdf

Govender, I. (2021). Towards understanding information systems students' experience of learning introductory programming: A phenomenographic approach. *Journal of Information Technology Education: Innovations in Practice*, *20*, 81–92. https://doi.org/10.28945/4782

Govender, I., Govender, D. W., Havenga, M., Mentz, E., Breed, B., Dignum, F., & Dignum, V. (2014). Increasing self-efficacy in learning to program: Exploring the benefits of explicit instruction for problem solving. *The Journal for Transdisciplinary Research in Southern Africa*, *10*(1), 187–200. https://doi.org/10.4102/TD.V10I1.19

Kallia, M., & Sentance, S. (2019). Learning to use functions: The relationship between misconceptions and self-efficacy. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 752–758. https://doi.org/10.1145/3287324.3287377

Kanaparan, G., Cullen, R., & Mason, D. (2019). Effect of self-efficacy and emotional engagement on introductory programming students. *Australasian Journal of Information Systems*, *23*, 1–21. https://doi.org/10.3127/AJIS.V23I0.1825

Khaleel, F. L., Ashaari, N. S., & Wook, T. S. M. T. (2019). An empirical study on gamification for learning programming language website. *Jurnal Teknologi*, *81*(2), 151–162. https://doi.org/10.11113/JT.V81.11133

Kinnunen, P. (2009). *Challenges of teaching and studying programming at a university of technology – Viewpoint of students, teachers and the university* [Doctoral dissertation, Tietotekniikan laitos, Aalto University]. https://aaltodoc.aalto.fi/items/5537e6ee-b600-47e4-91b6-fd0e5d8630b3

Kittur, J. (2020). Measuring the programming self-efficacy of electrical and electronics engineering students. *IEEE Transactions on Education*, *63*(3), 216–223. https://doi.org/10.1109/TE.2020.2975342

Kolar, H., Carberry, A., & Amresh, A. (2013). Measuring computing self-efficacy. *ASEE Annual Conference and Exposition*. https://peer.asee.org/measuring-computing-self-efficacy

Konecki, M., & Petrlic, M. (2014). Main problems of programming novices and the right course of action. *Central European Conference on Information and Intelligent Systems*, 116–123. https://www.proquest.com/docview/1629618268

Kori, K., Pedaste, M., Leijen, Ä., & Tõnisson, E. (2016). The role of programming experience in ICT students' learning motivation and academic achievement. *International Journal of Information and Education Technology*, *6*(5), 331–337. https://doi.org/10.7763/IJIET.2016.V6.709

Kori, K., Pedaste, M., Niitsoo, M., Kuusik, R., Altin, H., Tõnisson, E., Vau, I., Leijen, Ä., Mäeots, M., Siiman, L., Murtazin, K., & Paluoja, R. (2015). Why do students choose to study information and communications technology? *Procedia - Social and Behavioral Sciences*, *191*, 2867–2872. https://doi.org/10.1016/j.sbspro.2015.04.249

Lawan, A. A., Abdi, A. S., Abuhassan, A. A., & Khalid, M. S. (2019). What is difficult in learning programming language based on problem-solving skills? *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 18–22. https://doi.org/10.1109/ICOASE.2019.8723740

Lindblom-Ylänne, S., Parpala, A., & Postareff, L. (2019). What constitutes the surface approach to learning in the light of new empirical evidence? *Studies in Higher Education*, *44*(12), 2183–2195. https://doi.org/10.1080/03075079.2018.1482267

Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016). The influence of problem solving abilities on students' performance on different assessment tasks in CS1. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 329–334. https://doi.org/10.1145/2839509.2844596

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016). Programming, problem solving, and self-awareness: Effects of explicit guidance. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 1449–1461. https://doi.org/10.1145/2858036.2858252

Lonka, K., Olkinuora, E., & Mäkinen, J. (2004). Aspects and prospects of measuring studying and learning in higher education. *Educational Psychology Review*, *16*(4), 301–323. https://doi.org/10.1007/s10648-004-0002-1

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 55–106. https://doi.org/10.1145/3293881.3295779

Mahatanankoon, P., & Wolf, J. (2021). Cognitive learning strategies in an introductory computer programming course. *Information Systems Education Journal*, *19*(3), 11–20. https://eric.ed.gov/?id=EJ1301236

Malik, S. I., Shakir, M., Eldow, A., & Ashfaque, M. W. (2019). Promoting algorithmic thinking in an introductory programming course. *International Journal of Emerging Technologies in Learning (iJET)*, *14*(01), 84–94. https://doi.org/10.3991/IJET.V14I01.9061

Marton, F., & Säljö, R. (1976). On qualitative differences in learning: I – outcome and process. *British Journal of Educational Psychology*, *46*(1), 4–11. https://doi.org/10.1111/J.2044-8279.1976.TB02980.X

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Musil, C. M., Jones, S. L., & Warner, C. D. (1998). Structural equation modeling and its relationship to multiple regression and factor analysis. *Research in Nursing Health*, *21*(3), 271–281. https://pubmed.ncbi.nlm.nih.gov/9609512/

Pawlowski, J. M. (2007, April). The quality adaptation model: Adaptation and adoption of the quality standard ISO/IEC 19796-1 for learning, education, and training. In *Quality research for learning, education, and training* (pp. 3–16, Vol. 10). Taiwan: International Forum of Educational Technology & Society. https://www.jstor.org/stable/jeductechsoci.10.2.3

Peng, J., Wang, M., & Sampson, D. (2017, October). Visualizing the complex process for deep learning with an authentic programming project. In *Educational Technology & Society* (pp. 275–287, Vol. 20). https://www.jstor.org/stable/26229223

Peter, J. P. (1981). Construct validity: A review of basic issues and marketing practices. *Journal of Marketing Research*, *18*(2), 133–145. https://doi.org/10.1177/002224378101800201

Pham, T. B. T., Dang, L. A., Le, T. M. H., & Le, T. H. (2020). Factors affecting teachers' behavioral intention of using information technology in lecturing-economic universities. *Management Science Letters*, *10*(11), 2665–2672. https://growingscience.com/beta/msl/3848-factors-affecting-teachers-behavioral-intention-of-using-information-technology-in-lecturing-economic-universities.html

Ranjeeth, S. (2011). The impact of learning styles on the acquisition of computer programming proficiency. *Alternation*, *18*(1), 336–353. https://typeset.io/pdf/the-impact-of-learning-styles-on-the-acquisition-of-computer-2xrmdwf8qq.pdf

Robins, A., Haden, P., & Garner, S. (2006). Problem distributions in a CS1 course. *Proceedings of the 8th Australasian Conference on Computing Education*, *52*, 165–173. https://doi.org/10.5555/1151869.1151891

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*(1), 42. https://doi.org/10.1186/s41239-017-0080-z

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, *25*(1), 54–67. https://doi.org/https://doi.org/10.1006/ceps.1999.1020

Saunders, M., Lewis, P., & Thornhill, A. (2009). *Research methods for business students* (5th ed.). Pearson Education.

Sekaran, U., & Bougie, R. (2016). *Research methods for business: A skill-building approach* (7th ed.). John Wiley & Sons.

Spada, M. M., & Moneta, G. B. (2012). A metacognitive-motivational model of surface approach to studying. *Educational Psychology*, *32*(1), 45–62. https://doi.org/10.1080/01443410.2011.625610

Swanson, R. A., & Holton, E. F. (2005). *Research in organizations: Foundations and methods in inquiry*. Berrett-Koehler.

Tavares, P. C., Henriques, P. R., & Gomes, E. (2017). A computer platform to increase motivation in programming students – PEP. *Proceedings of the 9th International Conference on Computer Supported Education (CSEDU)*, *1*, 284–291. https://doi.org/10.5220/0006287402840291

Trigwell, K., Prosser, M., & Waterhouse, F. (1999). Relations between teachers' approaches to teaching and students' approaches to learning. *Higher Education*, *37*(1), 57–70. https://doi.org/10.1023/A:1003548313194

Tukiainen, M., & Mönkkönen, E. (2002). Programming aptitude testing as a prediction of learning to program. *14th Workshop of the Psychology of Programming Interest Group*, 45–57. https://www.ppig.org/papers/2002-ppig-14th-tukiainen/

Vanthournout, G., Coertjens, L., Gijbels, D., Donche, V., & Van Petegem, P. (2013). Assessing students' development in learning approaches according to initial learning profiles: A person-oriented perspective. *Studies in Educational Evaluation*, *39*(1), 33–40. https://doi.org/10.1016/j.stueduc.2012.08.002

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*, 3717–3725. https://doi.org/10.1098/RSTA.2008.0118

Yacob, A., & Saman, M. Y. M. (2012). Assessing level of motivation in learning programming among engineering students. *The International Conference on Informatics and Applications*

(ICIA2012), 425–432. https://www.researchgate.net/profile/Yazid-%20Saman/publication/230771467

A STATISTICAL TESTS

Table A1: One-sampled Wilcoxon signed rank test for Motivation

	1-sample t-test Test value = 3					**1-sample Wilcoxon signed rank test**	
			Significance				
Motivation	**t**	**df**	1-sided p	2-sided p	**Mean difference**	**Sig.**[*†]	**Decision** Null hypothesis
IM1: I prefer course material that really challenges me so I can learn new things	7.508	132	0.000	0.000	0.669	0.000	Reject
IM2: When I don’t understand something right away I try to figure it out by myself	11.181	132	0.000	0.000	0.872	0.000	Reject
IM3: I prefer course material that arouses my curiosity even if it is difficult to learn	8.852	132	0.000	0.000	0.767	0.000	Reject
IM4: Getting good marks for programming brings me a sense of personal satisfaction	15.681	132	0.000	0.000	1.226	0.000	Reject
IM5: I engage with new technology so that I have a sense of control over the technology	9.650	132	0.000	0.000	0.842	0.000	Reject
EM1: I want to do well in my programming modules because it is important to show my ability to my family, friends and lecturers	6.808	132	0.000	0.000	0.647	0.000	Reject
EM2: I engage with new technology because that is what society expects of me	0.142	132	0.444	0.888	0.015	0.880	Retain
EM3: I make an effort to master computer programming so that I can “fit in” with other students in my group/class	1.865	132	0.032	0.064	0.195	0.107	Retain

[*] The significance level is .050
[†] Asymptotic significance is displayed

Table A2: One-sampled Wilcoxon signed rank test for Learning Styles

	1-sample t-test Test value = 3					**1-sample Wilcoxon signed rank test**	
			Significance				
Learning Styles	**t**	**df**	1-sided p	2-sided p	**Mean difference**	**Sig.**[*†]	**Decision** Null hypothesis
LS1: I find most new topics interesting and will often spend extra time trying to understand how they work	11.687	132	0.000	0.000	0.789	0.000	Reject
LS2: I find it helpful to study topics in depth rather than trying to remember important facts for tests	10.900	132	0.000	0.000	0.789	0.000	Reject
LS3: I test myself on important topics until I understand them completely	9.848	132	0.000	0.000	0.684	0.000	Reject
LS4: I tend to study best by using memorisation techniques	6.101	132	0.000	0.000	0.481	0.000	Reject
LS5: I find the best way to pass tests is trying to learn the answers to likely questions	3.337	132	0.001	0.001	0.293	0.002	Reject
LS6: I prefer to ensure that I pass a course even though my understanding of concepts may not be very good	5.523	132	0.000	0.000	0.474	0.000	Reject

[*] The significance level is .050
[†] Asymptotic significance is displayed

Table A3: Significance testing for Self-Efficacy

	1-sample t-test Test value = 3					1-sample Wilcoxon signed rank test	
			Significance				
Self-efficacy	**t**	**df**	1-sided p	2-sided p	**Mean difference**	**Sig.**[*†]	**Decision** Null hypothesis
SE1: I am confident of my ability to develop suitable strategy for a given programming task in a short time	4.513	132	0.000	0.000	0.406	0.000	Reject
SE2: I am able to construct programming code that is logically correct	6.613	132	0.000	0.000	0.549	0.000	Reject
SE3: I have the capacity to easily identify errors in my programming code	5.480	132	0.000	0.000	0.474	0.000	Reject
SE4: I am able to mentally trace through the execution of a long, complex program	1.061	132	0.145	0.291	0.090	0.294	Retain
SE5: I could organize and design my program in a modular/procedural manner	4.141	132	0.000	0.000	0.368	0.000	Reject
SE6: I have a good understanding of the object-oriented paradigm for programming	4.322	132	0.000	0.000	0.391	0.000	Reject
SE7: I could rewrite lengthy and confusing portions of code to be more readable and clearer	1.029	132	0.153	0.305	0.090	0.278	Retain
SE8: I am confident of my ability to identify the objects in the problem domain and declare, define, and use them	3.800	132	0.000	0.000	0.323	0.001	Reject
SE9: I am able to write computer programming code to sort out a given set of numbers into ascending/descending order	8.190	132	0.000	0.000	0.699	0.000	Reject
SE10: I feel that I am better at programming when I get the help of someone else	-7.832	132	0.000	0.000	-0.729	0.000	Reject
SE11: I feel more comfortable to complete a programming problem if someone showed me how to solve the problem first	-8.336	132	0.000	0.000	-0.752	0.000	Reject
SE12: I could manage my time efficiently if I had a pressing deadline on a programming project	5.741	132	0.000	0.000	0.511	0.000	Reject

[*] The significance level is .050
[†] Asymptotic significance is displayed