The Inclusion of Python as Introductory Computer Programming in the Preparatory Year of Higher Education: Modeling for Students' Perceptions

Wafaa Alsaggaf[1]
Department of Information Technology
King Abdulaziz University
Jeddah, Saudi Arabia
waalsaggaf@kau.edu.sa

Samar Alkhuraiji[2]
Department of Computer Science
King Abdulaziz University
Jeddah, Saudi Arabia
salkhuraiji@kau.edu.sa

Hanan Baaqeel[3]
Department of Statistics
King Abdulaziz University
Jeddah, Saudi Arabia
hbaageel@kau.edu.sa

Hani Brdesee[4]
Department of Computer Information Technology
King Abdulaziz University
Jeddah, Saudi Arabia
hbrdesee@kau.edu.sa

Abstract:
Programming is one of the most important subjects that can logically assist with analytical thinking and finding solutions to problems. This, in turn, allows students at the undergraduate level to im-prove their critical thinking in line with the level of study at that stage. It is well known that ac-quiring programming skills is not easy, especially for novice programmers. This study aimed to study, for the first time, the inclusion of the Python programming language in the preparatory year for one of the largest univers0ities in Saudi Arabia. The main objective of this study was to investigate the factors that may affect the process of learning computer programming, and the factors that may help predict the level of students' knowledge of programming skills. For the purpose of this research, an exploratory questionnaire was designed and distributed to the first batch of students for this course, and was analyzed using several statistical methods. This study presents the results by discussing the factors that may affect the process of learning programming, the most important of which were the difference in the prior knowledge of programming at the general education stage, the looping concept, and problem-solving skills. Laboratory assignments and recorded lectures were found to be among the most important factors that may help to predict a high level of performance for students. We intend to implement some of the proposed solutions from this study and compare them with subsequent studies at a future stage.

Keywords: *Python; data analysis; introductory programming; computer programming education; higher education; preparatory year; novice programmers*

I. INTRODUCTION

Learning computer programming is one of the essential requirements for higher education. Many studies have proven that learning programming is difficult and com-plex, and that students, especially beginners, face many challenges in solving programming problems [1], [2]. Despite the existence of many studies finding solutions to these problems, they still exist, as can be seen in the research by Medeiros [3].

Due to this prior knowledge of the challenges of learning programming, including a programming subject for the first time in any study program is important, and worthy of study and investigation, especially in a context in which there is little research aimed at studying the background knowledge and the relationship between public education and higher education. Accordingly, the main objective of this research is to study the inclusion of a programming subject in the preparatory year for one of the largest universities in the Kingdom of Saudi Arabia. Moreover, as this preparatory year has a significant impact on the academic and professional future of students, as described in detail in Brdesee and Alsaggaf [4], it reinforces the importance of this study.

King Abdulaziz University has integrated many educational improvements and innovative systems that were designed using the latest programming languages, data-bases, and software for their excellence [5]–[9]. Furthermore, providing quality innovative, interactive education services to university students provides a gateway to the university's digital educational offerings through their portals [10]–[14]. The university has been ranked first in the Middle East and within the top 200 universities globally, which is a good reason to select such a university in which to conduct this research. Therefore, this research aims to investigate the factors that may affect the process of learning computer programming for the target group in this study, and looks at the aspects that help predict the level of student knowledge of programming skills in a leading university.

The rest of this paper is organized as follows. Section 2 presents some background information for the context of this study. Section 3 describes the literature review, and Section 4 details the materials and methods, including course structure, course preparation, teaching and learning strategies, and the survey used to obtain the data. Section 5 presents and analyzes the results of the research objectives. A discussion

Manuscript received March 5, 2022
Manuscript revised March 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.3.73

of the results obtained by the survey is provided in Section 6. Conclusions and future work are outlined in Section 7 (see Fig. 1).

II. BACKGROUND

At the beginning of each academic year, universities face a huge demand from students for admission and social and administrative pressures in organizing the ad-mission process, which sometimes results in students not being accepted into the college that is appropriate for their abilities and skills. This may be because the students do not have enough information about the college and its subjects, and possible future careers. There is a difference between secondary school and university academic styles. Thus, the preparatory year paves the way for the transition between the stages of secondary school and university. In the preparatory year, students study full-time in the morning hours and must attend daily according to the academic schedule that is registered to them. The duration of the preparatory year is one academic year divided into two semesters. Throughout this year, students study all subjects in the first and second academic semesters.

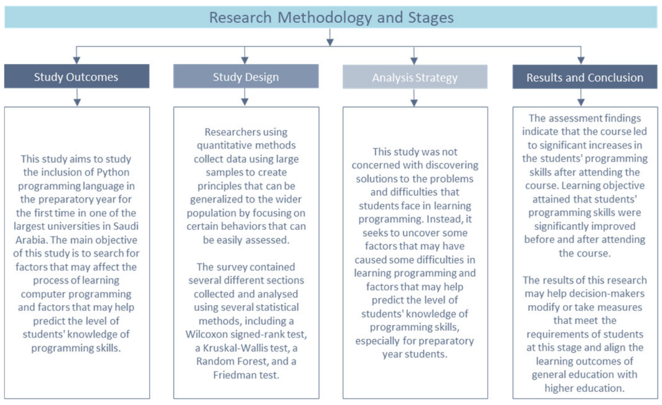
A. Preparatory Year Objectives

Creating a preparatory year at university has many advantages, which can be summarized as follows:

- Rationalizing admission by directing students to the appropriate college based on their abilities and skills, then based on their desires and choices.
- Standardization of admission to university.
- Introducing students to the subjects available at the university and the nature of study there.
- Introducing students to university bylaws and regulations.
- Providing students with the necessary skills and knowledge in English and computer usage, and developing learning, research, and communication skills.
- Allowing students to discover their scientific capabilities in a university environment

B. General Framework for the Preparatory Year

According to the Deanship of Admission and Registration at King Abdulaziz University, the preparatory year consists of: (1) the health colleges track; (2) the science colleges track; and (3) the administrative and human sciences colleges track. After completing all the preparatory year subjects, students are enrolled in colleges ap-propriate to their desires, paths, and abilities compared with each other, according to the vacancies and achievements [15].



This study aims to study the inclusion of Python programming language in the preparatory year for the first time in one of the largest universities in Saudi Arabia. The main objective of this study is to search for factors that may affect the process of learning computer programming and factors that may help predict the level of students' knowledge of programming skills.

Researchers using quantitative methods collect data using large samples to create principles that can be generalized to the wider population by focusing on certain behaviors that can be easily assessed.

The survey contained several different sections collected and analysed using several statistical methods, including a Wilcoxon signed-rank test, a Kruskal-Wallis test, a Random Forest, and a Friedman test.

This study was not concerned with discovering solutions to the problems and difficulties that students face in learning programming. Instead, it seeks to uncover some factors that may have caused some difficulties in learning programming and factors that may help predict the level of students' knowledge of programming skills, especially for preparatory year students.

The assessment findings indicate that the course led to significant increases in the students' programming skills after attending the course. Learning objective attained that students' programming skills were significantly improved before and after attending the course.

The results of this research may help decision-makers modify or take measures that meet the requirements of students at this stage and align the learning outcomes of general education with higher education.

Fig. 1. Research methodology and stages.

A committee for the preparatory year submits its recommendations to the University Council and approves the tracks and number of students that can be accepted yearly. It also monitors and supervises the preparatory year in general. After that, the Deanship of Admission and Registration makes admission available on the electronic portal. Students are accepted according to the conditions prepared and announced in the admission guide, which is published annually. The admission guide explains all the programs available in the colleges after the successful completion of the preparatory year.

C. The Programming Subject of Python, the Beginning and Importance

Prior to 2020, students studied an introductory computer skills course that included the basics of information technology and operating systems, and basic office programs. However, the university found that the course was too irrelevant to the scientific tracks and did not adequately establish students' knowledge. Therefore, the university's specialized committees included a new subject with programming skills and Python programming language problem-solving.

The main objective of this course is to teach the students the basics of constructing algorithms and programming languages. According to Computer Skills Unit, at the end of this course, students are expected to have learned the basic skills of algorithmic problem-solving, a systematic approach to defining problems and creating a number of solutions, and basic programming skills, which include syntax, commands, variables, selection statements, loops, functions, etc. [16].

On its webpage, the college of Computer Science and IT, the subject provider, argues that "by completion of the course the students should be able to:

1. Construct algorithms for solving simple problems.
2. Write a programming code that implements algorithms for solving simple problems.
3. Analyze and explain the behavior of simple programs involving the fundamental programming constructs.

4. Identify and describe uses of Python built-in data types and functions.
5. Write programs that use Python built-in data types and functions.
6. Apply appropriate conditional and iteration constructs for a given programming task.
7. Write and modify short programs that use standard conditional structures.
8. Write programs that use standard iterative control structure.
9. Write programs that use functions.
10. Trace the execution of a variety of code segments and write summaries of their computations.
11. Identify common coding errors and apply strategies for avoiding such errors.
12. Apply a variety of strategies to the testing and debugging of simple programs.
13. Use an appropriate IDE (Integrated Development Environment) to create, compile and run a program developed by the selected programming language."

III. Literature Review

Regarding the significance of acquiring programming and computational thinking in higher education, Agbo et al. [17] investigated the literature on programming education, specifically computational thinking in higher education. While this paper investigates the perception of students and instructors on developing an introductory programming course for the preparatory year in King Abdulaziz University, Saudi Arabia, this section tackles a review of the literature on the topic, covering several points. Agbo et al. claim that computational thinking has boomed in several contexts, especially in the developed world. The researchers' view is that, although most educators use the course design approach for this type of education in higher education institutions, the computational thinking approach is recommended for computer programming learning for beginners in higher institutions. It boosts the cognitive abilities of students and bridges the gap between those students who have programming back-grounds and those who do not.

Meanwhile, Medeiros et al. [3] performed a systematic literature review to better understand the issues and problems relevant to introductory programming. The study concludes that students need previous problem-solving skills and mathematical knowledge to learn introductory programming. In contrast, the study adds that novice students need motivation and engagement, and teachers need the appropriate tools and methods to succeed in this task. The study shows that a clarification of problem-solving is required and recommends more communication between primary and higher educational institutions.

In this context, Shein [18] argues that learning how to code in Python is essential for any learner since it is one of the main introductory programming languages. Mastering Python facilitates the transfer of programming concepts to other languages. Koulouri et al. [19] tackle how programming language selection, problem-solving training, and formative assessment may affect the learning process. The study finds that using Python as a simple programming language facilitates the learning process. Furthermore, Babbitt et al. [20] argue that, since computing is essential to other disciplines, all students need to have an introductory course in computer science to attain the required basic understanding of it. The study concludes that an introductory computer science course is a component that is deemed valuable and necessary for every student, recommending that such a course should be compulsory in any higher education program.

The literature on the topic has also focused on the skills needed to learn programming. Figueiredo and García-Peñalvo [21] investigated the strategies for teaching and learning programming in university education. The researchers maintain that teachers and students need dedication and motivation, as the study seeks to improve students' achievements in programming courses. The study suggests building a dynamic learning model that performs constant analysis of students' work throughout the course, therefore offering students more training and insight. Feaster et al. [22] investigated adapting CS (Computer Science) Unplugged materials, a set of active learning activities, to teach CS principles without having computers in class. The CS Unplugged pro-gram experiment resulted in failure. This experiment used a quasi-experimental control group for one semester, repeated for another two semesters. The program failed to change students' attitudes towards CS or learning the content as expected from a statistical perspective.

Novice CS students in higher education encounter several difficulties and mis-conceptions. Qian and Lehman [23] maintain that one of the teacher competencies in introductory programming courses is identifying and addressing students' misconceptions. The study investigates the difficulties that students encounter and the tools needed to address them, and finds that students reveal several misconceptions and other issues regarding syntax, concepts, and strategies due to their unfamiliarity with syntax, natural language, math, and strategies, among others. The study recommends conducting further research on CS education to address students' conceptions and misconceptions by integrating theories of conceptual change. The study also suggests the development of instructor pedagogical content knowledge (PCK). Pattanaphanchai [24] investigated using the flipped-classroom approach in introductory courses for CS programming in higher education in Thailand. The study aimed at measuring novice students' achievements based on their performance in a coding test and an examination. The study compared the scores of students who were taught using a flipped-classroom approach with the scores of students in a traditional classroom, and found that the students had a positive perception of the flipped-classroom approach, with the in-class activities boosting the students' understanding. The study also found that the students in the flipped classroom achieved better in their exams than the traditional students. Troya et al. [25] applied the flipped-classroom approach for under-graduates in computer laboratory sessions. The study argues that one of the issues encountered in the laboratory sessions is that 14–50% of the laboratory time is wasted on giving technical instructions, and if a student misses one laboratory session, they may face difficulties catching up again. The study found that using the flipped-laboratory approach addresses these problems and improves performance and motivation. Teachers also face several challenges in teaching introductory CS courses. A study performed by Qiyan et al. [26] found that a good CS teacher needs to know the common student misconceptions about CS. The study investigated CS teachers' understanding of student misconceptions. The study conducted a survey to assess the perceptions of teachers regarding students' misconceptions. The study found that teaching degrees and extra training give teachers more confidence in addressing student misconceptions.

One of the methods for teaching and/or learning introductory programming is watching recorded videos. Nørmark [27] claims that most students perceive depending on video resources in teaching programming as attractive, a finding based on two questionnaires. The study argues that the very positive student response to short video lectures is the study's most important finding. The researcher maintains that such videos must be considered carefully relative to the course content and workload, while some theoretical topics may be best taught using traditional methods. The study also recommends short videos rather than long ones. More recently, Picardo et al. [28] maintain that lecture recording is a valuable resource for students, especially after the shift to online modes, for several reasons, including the spread of the COVID-19 pan-demic, enabling students to access course content. This shines a light on the need to understand how students deal with course content in recording formats. The study probed how students cope with lecture recording and how far this affects their aca-demic performance. The researchers found a positive correlation between recording views and final scores, although a small percentage of students engaged in binge-watching, which is an unproductive activity that must be discouraged.

The literature also tackles the curriculum of introductory programming from different perspectives. Malik [29] argues that programming needs special skills, which is a challenge to students, as they have to learn problem-solving strategies and programming language semantics and syntax, among other skills. The study compared the used learning approach for introductory programming courses with the six categories of Bloom's taxonomy. The study found that the practical teaching/learning approach addresses the six categories, but half of the students' learning outcomes were still un-der the Not Good Enough category. Figueiredo and García-Peñalvo [30] discuss skills building in introductory programming courses, claiming that building skills in introductory programming is a universal problem. The study suggests building a profile for student competencies so that each student has the chance to improve specific skills through training. The study describes a system that suggests exercises and automatic assessments to construct a profile for each student. Allan et al. [31] note that the number of students majoring in CS has decreased, despite the increasing job opportunities for this industry, due to several factors, including curriculum revisions and introducing computational thinking into disciplines other than CS. The study discusses integrating computing into disciplines in the high school curriculum and the importance of raising students' awareness of CS as a rewarding field of study. Luxton-Reilly et al. [32] state that the literature on introductory programming is growing, and therefore the researchers are exploring the CS trends from the past and the possible future approaches.

IV. MATERIALS AND METHODS

The computer programming and problem-solving course provides an introductory structured programming course for novice students in their preparatory year at King Abdulaziz University. It is provided by the Computer Skill Unit/Faculty of Computing Science and Information Technology (FCIT). The main objectives are programming knowledge concepts followed by program writing skills, and the course is pre-pared by a team specializing in program teaching from the FCIT. The team focuses on choosing topics that do not have the technical difficulty that will be encountered in programming computer science courses, and they assume that all students do not have proficiency in these areas. The course team introduces students to the core concepts of programming, which may be very new to many students. They ensure that students can understand all the concepts of algorithms and the characteristics of programming using Python, and are able complete all the assignments without any difficulty. The course team looked at different textbooks until they decided on one book that covered all the desired concepts, "Introduction to Programming Using Python.". The course curriculum focuses on mastering the basic problem-solving skills and programming concepts.

A. Course Structure

- Introduction and fundamental concepts of computers and programming.
- The conceptual model: flowcharts and pseudo code.
- Variables, expressions, and statements, including input, processing, and output.
- Data and data structures.
- Conditional decision structures and Boolean logic.
- Repetition structures.
- Functions and return values.

B. Course Preparation

The course was prepared by a professional team of different instructors from the FCIT college. The course content materials are explained in PowerPoint slides covering all the course learning objectives (CLOs), defined in Section 2.3. The course curriculum for all lectures is taught over 16 weeks. In addition, 84 recorded videos were prepared on YouTube hosted by a professional lecturer from the FCIT college. The instructor ex-plains all the PowerPoint slide lectures. Furthermore, 18 videos were recorded ex-plaining the laboratory materials. All the prepared course materials were uploaded onto the Blackboard Learning Management System, which is available to all students. The course team also designed an instructor manual for each lecture and an instructor manual for the laboratory. The instructor manual provides detailed information about all the materials and concepts to be covered for each lecture during each week. It explains to the instructor the exercises given as examples and the exercises that should be discussed with the students during the classes. The laboratory instructors provide in-formation about the objectives of the laboratory sessions and what current laboratory learning outcomes should be covered during the sessions. Moreover, the laboratory instructors explain all the activities that should be carried out during the session, and the activities that can be given to the students as practice activities. All these preparations are made to ensure all the course sessions are provided with the same materials and concepts to cover the course learning outcomes.

C. *Teaching and Learning Strategies*

The course currently includes 3 lectures of 150-min duration and a 90-min laboratory session per week. All the lectures and laboratory sessions are given in computer laboratories, and the entire number of classes during the semester is 42 with 13 laboratory sessions. The course is offered in 51 sections. All the instructors and laboratory instructors follow the course manuals for the lectures and laboratory sessions.

D. *Methods*

To answer the research question and address the five objectives of this research, a quantitative method was used. Researchers using quantitative methods collect data using large samples to create principles that can be generalized to the wider population by focusing on certain behaviors that can be easily assessed [33]. Our research used a survey as a quantitative instrument to produce quantitative data that was capable of statistical analysis. The survey questions aimed to investigate students' perceptions of the inclusion of the Python course in the preparatory year. At the end of the semester, the survey was widely distributed to all students enrolled in the Python course (1485 students), of which 1361 were returned, making it a response rate of 92%, which is relatively high and can be considered to be representative of the population [34]. We designed our survey in five sections. The first section required information about the type of school the students came from (public school, private school, or international school) and the weighted ratio of their high school result. The second section considered the level of knowledge the students had before and after the course. In the third section, students were asked to rank the difficulty of the main topics included in the course, which were: programming basic concepts, mathematical functions, selections, loops, and functions. The fourth section was composed of questions designed to gather data on the level of difficulty of the four main components, which were: lecture content, practical laboratory content, problem-solving practices, and coding with Python. The fifth section included two questions: how much do you rely on recorded lectures? and to what extent do you benefit from laboratory applications in understanding the material? Those two questions were based on a five-point Likert scale: "high-5 to low-1". The last section included open-ended questions, with two questions asking respondents to list some of the best things they liked and the most difficult things they faced in the course.

V. RESULTS

To answer the two main questions in this research, we present the results by di-viding them into five objectives, as follows:

- Objective 1: compare the knowledge before and after attending the course.
- Objective 2: compare the students' programming skills before and after attending the course between the three types of schools.
- Objective 3: study the factors that were affected by the students' satisfaction with the programming course after attending the course.
- Objective 4: compare the degree of difficulty of the subjects in the course.
- Objective 5: compare the degree of difficulty of the elements in the course.

1) *Objective 1*

In this study, we aimed to compare the difference between the students' programming skills before and after attending the course. The students' programming skills before and after attending the course had a significantly moderate positive correlation ($r(1361) = 0.25$, $p < 0.01$). Additionally, a Wilcoxon signed-rank test was con-ducted to determine whether there was a difference in the students' programming skills before ($Mdn = 1$) and after ($Mdn = 2$) attending the course. The results of that analysis indicated that there was a significant difference in the students' programming skills ($V = 13,830$, $p < 0.01$). As a result, the students' programming skills were found to have improved after attending the course, and the improvement is shown in Fig. 2–4.

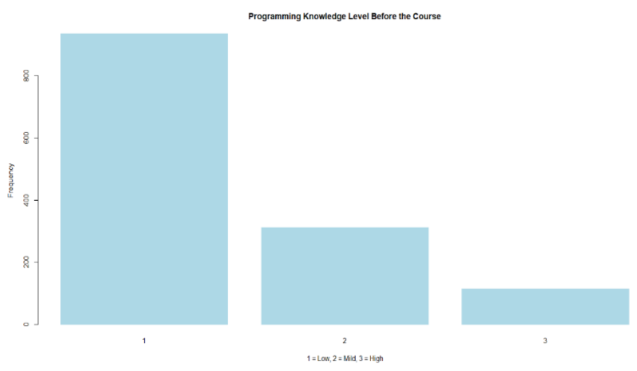


Fig. 2. Programming knowledge level before attending the course.

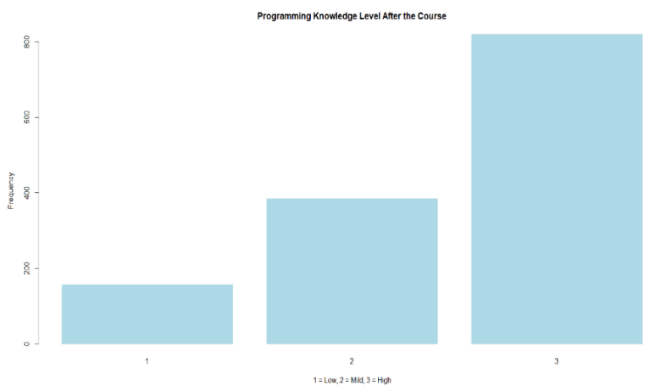


Fig. 3. Programming knowledge level after attending the course.

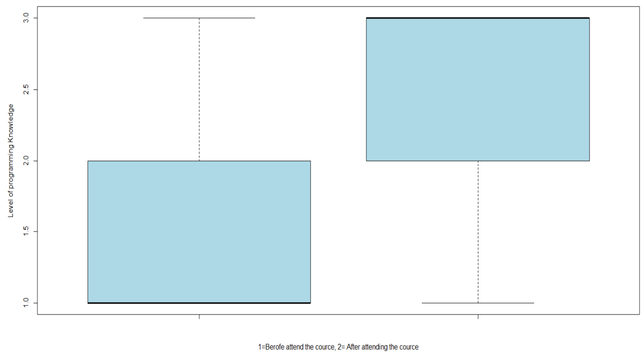


Fig. 4. Comparison between programming knowledge level before and after attending the course.

2) Objective 2 and 3

A Kruskal–Wallis test showed that the type of school (public, private, or international) significantly affects programming skills before attending the course (Mdnpub = 1, Mdnprv = 1, Mdnint = 3, H(2) = 15.68, p < 0.001). Post-hoc Wilcox signed-ranked tests using a Bonferroni-adjusted alpha level of 0.017 (0.05/3) were used to compare all pairs of groups. The programming skills of the students that came from international schools were higher than those of the public and private school students (W(Npub = 1069, Nint = 54) = 22,607, p < 0.001, W(Nprv = 238, Nint = 54) = 4632, p < 0.001); whereas there was no significant difference in programming skills between the students of public and private schools (W(Npub = 1069, Nprv = 238) = 135,238, p = 0.06). However, when a Kruskal–Wallis test was used to compare the students' programming skills after attending the course, the results changed, and we found that there was no significant difference in the programming skills among the students from the different schools (Mdnpub = 3, Mdnprv = 3, Mdnint = 3, H(2)= 0.38, p = 0.84; see Fig. 5 and Fig. 6.

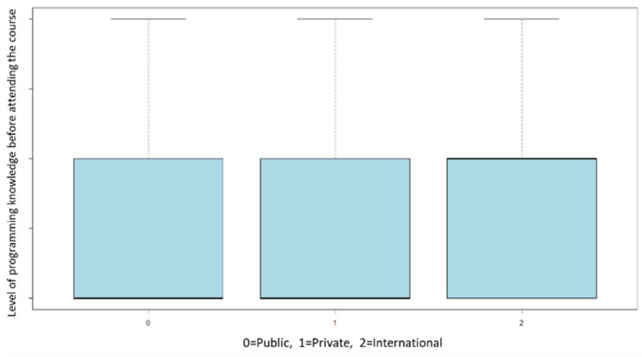


Fig. 5. Programming knowledge level before attending the course among the three different schools.

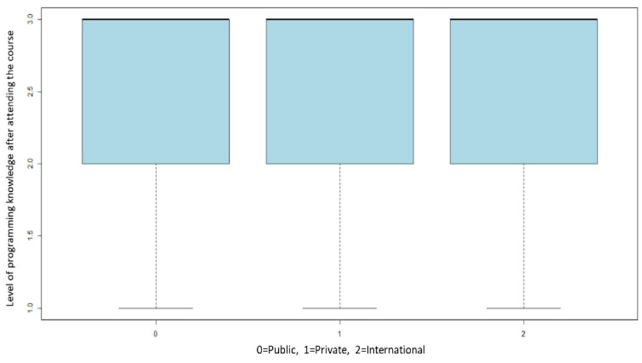


Fig. 6. Programming knowledge level after attending the course among the three different schools.

3) Objective 4

Regarding modeling and studying the predictors of the students' programming skills, a random forest (RF) classifier was applied. RF was helpful in classifying and predicting the level of the students' programming skills, as well as investigating the important factors affecting the students' programming skills.

The target variable we aimed to predict was students' programming skills. "K.A", is an ordered, categorical variable indicating the students' programming skills after attending the course, with three levels: good (coded 3), neutral (coded 2), and bad (coded 1). "Weighted.Ratio" is a continuous variable indicating a weighted ratio of students before being registered at university. "K.B" is an ordered, categorical variable indicating the students' programming skills before attending the course, with three levels: good (coded 3), neutral (coded 2), and bad (coded 1). "Lec.Rec" is an ordered, categorical variable indicating the satisfaction level with the recorded lecture videos: high (coded 3), medium (coded 2), and low (coded 1). "LAB.Ass" is an ordered, categorical variable indicating the satisfaction level with laboratory activities: high (coded 3), medium (coded 2), and low (coded 1).

We trained the random forest with the parameters (ntree = 1000 trees trained, and mtry = 3 factors chosen for each iteration).

To address how many of a classifier's predictions were correct, and when they were incorrect, we created a confusion matrix. In the confusion matrices below (Table 1), the rows represent the actual level of programming skills, and the columns indicate the predicted levels. The values on the diagonal show the number of times the RF predicted correctly; whereas the values on the off-diagonal represent an incorrect prediction or misclassification of the levels.

TABLE I. CONFUSION MATRIX

		Predicted Level		
		Bad	Neutral	Good
Actual level	Bad	12	8	8
	Neutral	9	28	39
	Good	24	55	177

There are three levels of students' programming skills (bad, neutral, and good); however, they are imbalanced, as shown in Table 1. As is known, accuracy is not a great measure of classifier performance when the classes are imbalanced. Therefore, we need more information to understand how well the model really performed (see Table 2).

First, we noted in Table 3 and 4 that the accuracy (number of correct predictions divided by the total number of predictions) was 60.28% with a 95% confidence interval of 0.5502 and 0.6537, meaning that there is a 95% likelihood that the true accuracy for this model lies within this range. The no-information rate is 0.6222. This is the accuracy achievable by always predicting the majority class label. The Kappa statistic shows how well our classifiers predictions matched the actual class labels while controlling for the accuracy of a random classifier. The Kappa for this model was 0.1966, which is low.

Moreover, sensitivity, also referred to as the true positive rate or recall, shows the proportion of the positive class correctly predicted, and the highest sensitivity was in class 3 (good). However, specificity, also referred to as the true negative rate, shows the proportion of the negative class correctly predicted, and the highest specificity was in class 1 (bad). Balanced accuracy essentially takes the average of the true positive and true negative rates, and achieved 0.60794, 0.56463, and 0.6046 for the bad, neutral and good classes, respectively.

In Fig. 7, "Mean Decrease Accuracy" represents how much the model accuracy decreases if we drop that variable, and "Mean Decrease Gini" is a measure of variable importance based on the Gini impurity index used for the calculation of splits in trees. As a result, we discovered the factors that affected students' programming skills, in order from highest to lowest, as follows. The most important factor predicting the level of the students' programming skills was the Weighted Ratio. Next, the important fac-tors were Lab Ass, Lec.Rec., and then K.B, while a less important factor was School.Type.

4) Objectives 5 and 6

A Friedman test was carried out to compare the difficulty level for the five course topics (basic concepts, mathematical functions, selections, loops and functions). A significant difference in difficulty level was found between the topics. Next, for the multiple comparison tests, Nemenyi post-hoc tests were carried out, and there were significant differences between them all ($p < 0.001$). In order, the difficulty level from the most difficult to the easiest topic was: loops, functions, selections, mathematical functions, then basic concepts (see Fig. 8).

TABLE II. FREQUENCY DISTRIBUTION OF STUDENTS' PROGRAMMING SKILLS

	Bad	**Neutral**	**Good**
Count	159	382	820
Percent%	11.68%	28.07%	60.25%

TABLE III. EVALUATION OF RANDOM FOREST CLASSIFIER: OVERALL STATISTICS

Accuracy	**0.6028**
95% CI	(0.5502, 0.6537)
No Information Rate	0.6222
Kappa	0.1966

TABLE IV. EVALUATION OF RANDOM FOREST CLASSIFIER: STATISTICS BY CLASS

	Bad	**Neutral**	**Good**
Sensitivity	0.26667	0.30769	0.7902
Specificity	0.94921	0.82156	0.4191
Balanced Accuracy	0.60794	0.56463	0.6046

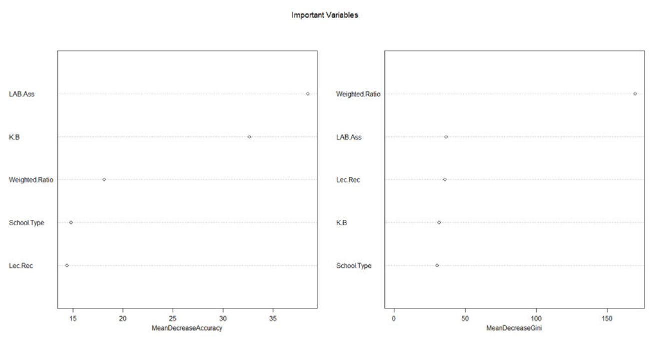


Fig. 7. Important variables.

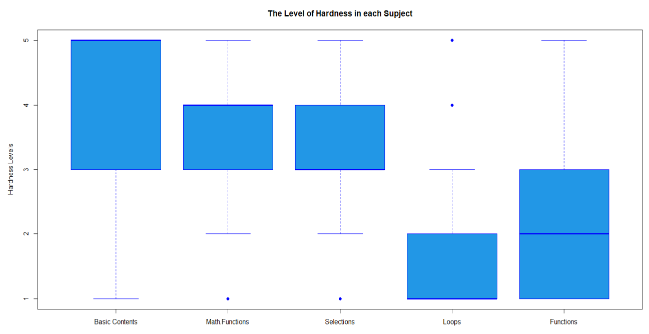


Fig. 8. The level of difficulty for each element.

In addition, a Friedman test was carried out to compare the difficulty level for the four course elements (lecture content, laboratory, problem-solving, and coding). A significant difference was found between the elements. Next, for the multiple comparison tests, Nemenyi post-hoc tests were carried out, and there were significant differences between them all ($p < 0.001$). In order, the difficulty level from the most difficult to the easiest element was problem-solving, coding, lecture content, and laboratory.

VI. Discussion

We believe that our findings have identified some interesting traits in the novice programmers who attended our introductory course. The assessment findings indicate that the course led to significant increases in the students' programming skills after attending the course. The learning objectives attained showed that the students' programming skills were significantly improved before and after attending the course. These findings also show that the students with a limited knowledge of programming skills also benefited from attending our course and showed a better performance in their results at the end of the course. We found statistically significant differences in performance between students from different types of schools (public, private and international) regarding their programming skills before attending the course. The programming skills of the students who came from international schools were higher than those of the students from public and private schools. The compared results indicated that, after attending the course, there was no significant difference in the programming skills among the students from the different schools.

The preparation of students for the university stage is an important topic to shed light on due to its great impact on the success of students at the university level, which undoubtedly affects their future careers. Over time and after many attempts aimed at harmonizing the outcomes of public and higher education, there is still a need to codify this process. Our results proved this when investigating the knowledge of students from different types of schools regarding software skills before studying the course. The results showed that the programming knowledge of the students who graduated from international schools was higher than for those who graduated from public and private schools. This difference may cause difficulties in managing classrooms to de-liver information to all students equally without resulting in some students feel bored or others feeling that they are at a lower level than their classmates. Unifying the learning outcomes in the general education school stage will make it easier for university education decision-makers to build stronger curricula that suit all students and elevate them and their level of thinking. The good news is that, despite the challenges faced by the teachers of programming regarding the difference in the students' prior programming knowledge, the students were able to reach a level of knowledge after studying the course. Furthermore, this was not affected by the type of school the student graduated from. However, on the other hand, there may be an educational loss that it was possible for students who graduated from international schools to acquire based on their previous knowledge and reach to know the superiority of their peers from other schools.

When studying the factors that help predict the level of students' knowledge of programming skills, we found that the percentage of students accepted into university is one of the most important factors affecting their success. Perhaps this is normal, as it is related to the nature of students and their diligence. This is followed by laboratory work, which is considered the second most important factor affecting the level of stu-dents' knowledge of programming skills. This confirms that programming is a practical and applied subject, suggesting more laboratory hours as a recommendation from the outputs of this research for decision-makers at the university. Recorded lectures were the third most important factor as they allowed students to listen to the lecture at any time and in any place before or after the lecture. Continuing to access the lectures is seen as beneficial, desirable and recommended. Fortunately, the tribal knowledge of programming and the type of school were among the minor influential factors in predicting the level of students' programming knowledge because of the differences, as mentioned previously.

One of the objectives of this study was to discern what content the students may find the most difficult to focus on in the future, and to re-design the curriculum to suit the needs of the students. We found that looping is the most difficult topic to under-stand and apply. This is in agreement with previous studies that confirmed that looping is a challenge for students, especially beginners. The course is designed to include all the basic concepts that students need for programming, which is one of the most important subjects to stimulate analysis and problem-solving. However, based on the results we obtained, it is recommended that the course content be redistributed to include more hours of looping instruction, in addition to the development of modern methods to communicate the ideas more easily, especially for millennial students. For example, using gamification and other attractive methods for concepts that students find difficult, including converting problem analysis into programming code.

In addition to discovering the content that students find most difficult, we studied the most difficult elements that students face during their studies. We found that problem-solving is one of the biggest challenges that students find difficult to practice easily. This is also confirmed by previous studies. Although problem-solving is included in all exercises and questions during lectures and in laboratory sessions, it is still the first and most significant challenge. It may be worth experimenting to discuss problem-solving in some parts of the course, especially at the beginning, regarding real-life issues that are not related to programming. Discussing this skill in general education outcomes is also very important. The arrival of students at the university stage with limited problem-solving skills may have an impact on their scientific and professional careers.

VII. Conclusions

This study aimed to examine the factors that may affect the process of learning computer programming for preparatory year students in one of the largest universities in the Kingdom of Saudi Arabia, specifically for the first batch of students. The aspects that help predict students' knowledge of programming skills were also investigated. For the purpose of this research in studying the inclusion of a computer programming subject for the first time in the preparatory year, a questionnaire was designed and distributed to students at the end of the semester. The survey contained several different sections that were collected and analyzed using several statistical methods, including a Wilcoxon signed-rank test, a Kruskal–Wallis test, a random forest, and a Friedman test. It should be noted that this study was not concerned with discovering solutions to the problems and difficulties that students face in learning programming. Instead, it sought to uncover the factors that may cause difficulties in learning programming, and the factors that may help to predict the level of students' programming knowledge, especially for preparatory year students. We can confidently draw some preliminary conclusions regarding the factors that may affect the process of learning programming for this group of students, such as a discrepancy in prior programming knowledge at the general education stage; a specific time and style of teaching the looping concept, which students considered to be one of the most difficult concepts in programming; and the skill of problem-solving, which was classified as one of the big-gest challenges that students faced during their studies. On the other hand, laboratory assignments and recorded lectures were among the most critical factors that may help to predict a good level for students in programming.

The results of this research may help decision-makers to modify or take measures to meet the requirements of students at this stage, and align the learning outcomes of general education with higher education. Furthermore, other exploratory studies were conducted and compared to this study after making the adjustments and appropriate decisions. In the future, we intend to search for solutions to the problems discussed and investigate mechanisms of organization, teaching, and course design that may assist in attaining desirable results.

References

[1] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*, New York, NY, USA, Jun. 2014, pp. 39–44. doi: 10.1145/2591708.2591749.

[2] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," *SIGCSE Bull.*, vol. 39, no. 2, pp. 32–36, Jun. 2007, doi: 10.1145/1272848.1272879.

[3] R. P. Medeiros, G. L. Ramalho, and T. P. Falcão, "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education," *IEEE Transactions on Education*, vol. 62, no. 2, pp. 77–90, May 2019, doi: 10.1109/TE.2018.2864133.

[4] H. Brdesee and W. Alsaggaf, "Is There a Real Need for the Preparatory Years in Higher Education? An Educational Data Analysis for College and Future Career Readiness," *Social Sciences*, vol. 10, no. 10, Art. no. 10, Oct. 2021, doi: 10.3390/socsci10100396.

[5] A. Assiri, A. Almalais, H. Brdesee, and H. Baaqeel, "Towards an Innovative Educational Knowledge Model for Intelligent Academic Advising," *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, p. 12A11B: 113, 2021, doi: 10.14456/ITJEMAST.2021.212.

[6] H. Brdesee, "A Divergent View of the Impact of Digital Transformation on Academic Organizational and Spending Efficiency: A Review and Analytical Study on a University E-Service," *Sustainability*, vol. 13, no. 13, Art. no. 13, Jan. 2021, doi: 10.3390/su13137048.

[7] A. Assiri, A. AL-Malaise, and H. Brdesee, "From Traditional to Intelligent Academic Advising: A Systematic Literature Review of e-Academic Advising," *International Journal of Advanced Computer Science and Applications*, vol. 11, Jan. 2020, doi: 10.14569/IJACSA.2020.0110467.

[8] H. Brdesee, "Outstanding Development in Student E-services: A Case Study of the Electronic Standardized Letters of Recommendation (E-SLOR)," *ICERI2019 Proceedings*, pp. 1095–1102, 2019.

[9] H. Brdesee, "A mixed method analysis of the online information course withdrawal system," *Behaviour & Information Technology*, vol. 37, no. 10–11, pp. 1037–1054, Nov. 2018, doi: 10.1080/0144929X.2018.1495764.

[10] H. S. Brdesee, "An Online Verification System of Students and Graduates Documents and Certificates: A Developed Strategy That Prevents Fraud Qualifications," *IJSEUS*, vol. 10, no. 2, pp. 1–18, Apr. 2019, doi: 10.4018/IJSEUS.2019040101.

[11] H. Brdesee, A. Madbouly, A. Y. Noaman, and A. H. Ragab, "A Comprehensive Data Mining Framework Used To Extract Academic Advising Knowledge From Social Media Data," *INTED2017 Proceedings*, pp. 7691–7700, 2017.

[12] W. Alsaggaf, K. Asad, N. Algrigri, F. Alsaedi, and H. Brdesee, "An Electronic Students Attendance System Using Indoor Positioning and Mobile Apps Technologies," *INTED2017 Proceedings*, pp. 7781–7788, 2017.

[13] A. Y. Noaman, A. Madbouly, H. Brdesee, and F. Fouad, "Assessing the Electronic Academic Advising Success: An Evaluation Study of Advisors Satisfaction in Higher Education," *INTED2017 Proceedings*, pp. 7701–7709, 2017.

[14] H. Brdesee and W. Alsaggaf, "Academic Advising and Social Media: A Case Study on the Twitter Account of the Deanship and Registration of King Abdulaziz University," presented at the the Conference of Academic Advising in Higher Education of the Gulf Cooperation Council States: Reality and Hope, 253–267, 2015.

[15] “Deanship of Admission and Registration—Preparatory Year (kau.edu.sa).” https://admission.kau.edu.sa/Pages-260921.aspx (accessed Jan. 08, 2022).

[16] Computer Skills Unit, “Aboutcpit110ar (kau.edu.sa).” https://csu.kau.edu.sa/pages-aboutcpit110ar.aspx (accessed Jan. 08, 2022).

[17] F. J. Agbo, S. S. Oyelere, J. Suhonen, and S. Adewumi, “A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions,” in *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, New York, NY, USA, Nov. 2019, pp. 1–10. doi: 10.1145/3364510.3364521.

[18] E. Shein, “Python for beginners,” *Commun. ACM*, vol. 58, no. 3, pp. 19–21, Feb. 2015, doi: 10.1145/2716560

[19] T. Koulouri, S. Lauria, and R. D. Macredie, “Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches,” *ACM Trans. Comput. Educ.*, vol. 14, no. 4, p. 26:1-26:28, Dec. 2015, doi: 10.1145/2662412.

[20] T. Babbitt, C. Schooler, and K. King, “Punch Cards to Python: A Case Study of a CS0 Core Course,” in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, New York, NY, USA, Feb. 2019, pp. 811–817. doi: 10.1145/3287324.3287491.

[21] J. Figueiredo and F. J. García-Peñalvo, “Building Skills in Introductory Programming,” in *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, New York, NY, USA, Oct. 2018, pp. 46–50. doi: 10.1145/3284179.3284190.

[22] Y. Feaster, L. Segars, S. K. Wahba, and J. O. Hallstrom, “Teaching CS unplugged in the high school (with limited success),” in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, New York, NY, USA, Jun. 2011, pp. 248–252. doi: 10.1145/1999747.1999817.

[23] Y. Qian and J. Lehman, “Students&#x2019; Misconceptions and Other Difficulties in Introductory Programming: A Literature Review,” *ACM Trans. Comput. Educ.*, vol. 18, no. 1, p. 1:1-1:24, Oct. 2017, doi: 10.1145/3077618.

[24] Prince of Songkla University and J. Pattanaphanchai, “An Investigation of Students’ Learning Achievement and Perception using Flipped Classroom in an Introductory Programming course: A Case Study of Thailand Higher Education,” *JUTLP*, vol. 16, no. 5, pp. 36–53, Dec. 2019, doi: 10.53761/1.16.5.4.

[25] J. Troya, J. A. Parejo, S. Segura, A. Gámez-Díaz, A. E. Márquez-Chamorro, and A. del-Río-Ortega, “Flipping Laboratory Sessions in a Computer Science Course: An Experience Report,” *IEEE Transactions on Education*, vol. 64, no. 2, pp. 139–146, May 2021, doi: 10.1109/TE.2020.3016593.

[26] Y. Qian, S. Hambrusch, A. Yadav, S. Gretter, and Y. Li, “Teachers’ Perceptions of Student Misconceptions in Introductory Programming,” *Journal of Educational Computing Research*, vol. 58, no. 2, pp. 364–397, Apr. 2020, doi: 10.1177/0735633119845413.

[27] K. Nørmark, “Using Short Videos in an Introductory Programming Course: International conference on E-Learning,” in *International Conference on E-learning*, La Laguna, Spain, Sep. 2014, pp. 254–260.

[28] V. Picardo, P. Denny, and A. Luxton-Reilly, “Lecture Recordings, Viewing Habits, and Performance in an Introductory Programming Course,” in *Australasian Computing Education Conference*, New York, NY, USA, Feb. 2021, pp. 73–79. doi: 10.1145/3441636.3442307.

[29] S. I. Malik, “Assessing the Teaching and Learning Process of an Introductory Programming Course With Bloom’s Taxonomy and Assurance of Learning (AOL),” *IJICTE*, vol. 15, no. 2, pp. 130–145, Apr. 2019, doi: 10.4018/IJICTE.2019040108.

[30] J. Figueiredo and F. J. García-Peñalvo, “Teaching and learning strategies of programming for university courses,” in *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality*, New York, NY, USA, Oct. 2019, pp. 1020–1027. doi: 10.1145/3362789.3362926.

[31] V. Allan, V. Barr, D. Brylow, and S. Hambrusch, “Computational thinking in high school courses,” in *Proceedings of the 41st ACM technical symposium on Computer science education*, New York, NY, USA, Mar. 2010, pp. 390–391. doi: 10.1145/1734263.1734395.

[32] A. Luxton-Reilly *et al.*, “Introductory programming: a systematic literature review,” in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, New York, NY, USA, Jul. 2018, pp. 55–106. doi: 10.1145/3293881.3295779.

[33] A. Sloane-Seale, “Research Design: Qualitative, Quantitative, and Mixed Methods Approaches,” *Can. J. Univ. Contin. Educ.*, vol. 35, 2009.

[34] A. Bryman, E. Bell, A. Mills, and A. Yue, “Business research strategies,” *Bus. Res. Methods*, pp. 226–238, 2007.

Wafaa Alsaggaf is an Assistant Professor at the Faculty of Computing and Information Technology in King Abdul Aziz University, Jeddah, Saudi Arabia. She holds a PhD in Computer Science from RMIT University, Australia. Her research interests are in the areas of mobile and ubiquitous learning, educational technologies, computer science education, and machine learning.

Hanan Mohamed Baaqeel is an Assistant Professor in Statistics, Statistics Department, Faculty of Sciences, King Abdulaziz University (KAU), Jeddah, Saudi Arabia. She received her Ph.D. degree in Statistics from the University of Nottingham, UK. Her research encompasses Theoretical and Applied Statistics, Applied Multivariate Analysis, Random Graph Models, and Statistical Network Analysis.

Hani Brdesee is an Associate Professor in Information Systems (IS), Electronic Business and E-trends, and associated with the Computer and Information Technology Department, Faculty of Applied Studies, King Abdulaziz University (KAU), Jeddah, Saudi Arabia. He received his Ph.D. degree in Information Systems from RMIT University, Australia. He is a Vice-dean of the Deanship of Admission and registration, KAU, and as a General Supervisor of the University Academic Departments Heads Forum (HSAD).

Samar Alkhuraiji is an Assistant Professor of Computer Science at King Abdulaziz University (KAU)/ Faculty of Computing and Information Technology. She holds a Ph.D. in Computer -Science from the University of Manchester, the United Kingdom. She did her master’s degree in software engineering from the Florida Institute of Technology, Melbourne, Florida, United States. My interest in Adaptivity in e-learning system based on students' behavior, Applied intelligent learning system, Smart Learning Environments, Virtual Reality, machine learning and imagining processing. She is Deputy Director of Computer Skill unit for the preparatory year at KAU.

.