

## **Question Driven Introductory Programming Instruction: A Pilot Study**

**Deepak Dawar**

**Recommended Citation:** Dawar, D. (2023). Question Driven Introductory Programming Instruction: A Pilot Study. *Journal of Information Systems Education*, 34(2), 231-242.

**Article Link:** <https://jise.org/Volume34/n2/JISE2023v34n2pp231-242.html>

Received: November 23, 2022

Revised: November 26, 2022

Accepted: January 25, 2023

Published: June 15, 2023

Find archived papers, submission instructions, terms of use, and much more at the JISE website:  
<https://jise.org>

ISSN: 2574-3872 (Online) 1055-3096 (Print)

---

# Question Driven Introductory Programming Instruction: A Pilot Study

**Deepak Dawar**

Department of Computer and Information Technology  
Miami University  
Hamilton, OH 45011, USA  
[daward@miamioh.edu](mailto:daward@miamioh.edu)

## ABSTRACT

For most beginners, learning computer programming is a complex undertaking. Demotivation and learned helplessness have been widely reported. In addition to the subject's complexity, low in-class involvement has been linked to poor student performance. This work introduces a novel instructional technique called Student-Driven Probe Instruction (SDPI) to address the low levels of in-class involvement in basic programming courses. The concept was straightforward: rather than the teacher lecturing/explaining material to the class and requesting questions, the students were shown a piece of code or other relevant material and given the opportunity to ask questions first. Explanations followed only after the questions had been asked, not before. Participation was tracked through two metrics: the number of questions asked in class and emails/Slack contacts with the instructor. Significant improvements were recorded for in-class participation. Average quiz scores also improved meaningfully. According to a course evaluation survey, students favored SDPI over the conventional lecture format since it piqued their interest in the material and gave them the confidence to ask questions in class.

**Keywords:** Teaching framework, Introductory programming, Pedagogy, Student engagement

## 1. INTRODUCTION

For most beginners, learning computer programming is a difficult task, and high failure rates are reported regularly (Allan & Kolesar, 1997; Beaubouef & Mason, 2005; Bennedsen & Caspersen, 2007; Howles, 2009; Kinnunen & Malmi, 2006; Mendes et al., 2012; Newman et al., 1970; Sheard & Hagan, 1998; Watson & Li, 2014). Students frequently become demotivated because of the subject's complexity (Kim & Lerch, 1997; Rogalski & Samurçay, 1990; Robins et al., 2003). A wide body of research shows that class participation improves student performance and has a meaningful impact on GPA (Credé et al., 2010). Many studies have reported that active class participation makes a considerable difference in student outcomes irrespective of whether the course is delivered synchronously or asynchronously (Duncan et al., 2012; Nieuwoudt, 2020).

The main motivation for this work is to study the reasons for low participation rates in programming classes and devise ways to increase them. After having taught many introductory programming classes over multiple years, the author has seen that roughly 20% of the students tend to ask the majority of the queries. About 30% occasionally interact, while the remaining seldom show any interest and simply watch. This is also corroborated by studies conducted by Bowers (1986) and the data collected for this study. This data is presented in Section 3.

Various strategies have been put forth to increase participation. Some of the most well-known include pair programming (Dongo et al., 2016; Williams et al., 2002) and gamification (Beavis, 2010; Majuri et al., 2018; Osatuyi et al., 2018; Seaborn & Fels, 2015). These methods are effective and

have been shown to have neutral to favorable effects on student outcomes. For instance, when problems are presented as puzzles, gamification encourages students to use computational thinking to solve them.

The author has used these strategies, and they appear to have increased the level of involvement for in-class problem-solving exercises. However, they had little effect on students' comfort level with asking questions. For instance, the author has frequently noticed that, after a pair-programming or gamification session, many students revert to their normal mode of non-participation when it comes to asking questions. Before and after these exercises, most the questions were raised by the same students who had been asking them earlier. The author contends that, in terms of the fundamental questions that students were asking, not much had changed.

Different techniques have been presented to motivate students to ask questions in class. Attaching weight to questions asked in class (Berdine, 1986; Smith, 1992) and the Random Selector Model (Allred & Swenson, 2006) are the notable ones. An excellent example of an external motivator is awarding points to students who ask questions, where the incentive for students to ask questions is to receive points. This may or may not influence student achievement or curiosity. Similarly, cold calling increases participation, but it also increases the stress levels of the class (Moguel, 2004).

Are there other ways to interest students in the material being presented and in asking questions in class? To explore this question, it is imperative to look at the deeper roots of the problem. The obvious question is: *Why do some students not participate or ask questions in class?*

Regardless of the physical characteristics of the classroom, a student's own anxieties about being incompetent/inadequate in front of others may prevent them from participating in class (Fritschner, 2000; Hyde & Ruth, 2002; Weaver & Qi, 2005), especially when it comes to expressing their doubts. In some studies (e.g., Armstrong & Boud, 1983; Wade, 1994), students reported that lack of confidence was what most discouraged them from asking questions. Many students do not participate because they are concerned about what other students think of them (Fritschner, 2000).

Additionally, for many beginners, the cognitive load of programming language concepts is unusually high (Kim & Lerch, 1997; Robins et al., 2003; Rogalski & Samurçay, 1990). Margulieux et al. (2018) compared the domain difficulty of three courses (computer programming, chemistry, and statistics) and found computer programming to be the most difficult due to the complexity of the content to be learned and handled at a given time. According to the value-expectancy (Keller, 1983) and cognitive load theories (Alexandron et al., 2014; Paas et al., 2010; Sweller, 1988, 1994), students will not participate in class if their perceived expectancy of success is low. A high cognitive load content lowers the expectancy of success for many students.

Due to the high cognitive load, many students may choose not to engage in class because it is difficult to understand several concepts simultaneously. This can exacerbate the insufficiency issue mentioned before. For example, to write the simplest of programs, one must understand the structure, syntax, and semantics of the programming language being used. One must also know what a compiler is and what runs a program. This is high cognitive load activity, and encountering such activities often during the course leads to student frustration. Subsequently, many students think they are incapable of succeeding. This is referred to as acquired helplessness (Crego et al., 2016).

Keeping these factors in mind, a pilot experiment called Student-Driven Probe Instruction (SDPI) was designed. The idea was simple but counterintuitive: instead of the instructor driving the class by offering explanations of the content and inviting questions, students were shown a piece of code/content. The instructor then invited questions without providing any explanations. The intention was twofold:

- 1) Reduce the initial cognitive burden and provide the students freedom to interpret the material subjectively. As a result, the connection between the student and the subject is entirely exploratory rather than being seen as something the student must understand or be tested on.
- 2) Let students, not the teacher, lead the class lesson through their questions. This adjustment gives the students more control, which can boost their self-esteem and help them let go of their sense of inadequacy.

As shown in Figure 1, the technique rests on three pillars – two of which have been described. The third pillar, maintaining a non-judgemental classroom environment, is critical to the success of the process. As every student will now see the content in their own way, they may ask the most basic questions. Judging their questions as lofty and irrelevant will immediately derail the process.

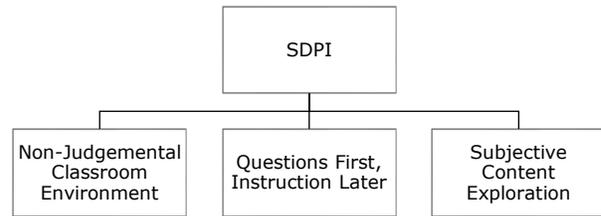


Figure 1. Student Driven Probe Instructional Technique

This study aims to address two research questions:

- a) What is the effect of the SDPI approach on student in-class participation?
- b) What is the impact of SDPI on student outcomes, if any?

The author could foresee at least two significant issues that could derail the potential acceptability of this technique:

- a) Will the unconventional nature of the technique, albeit with the right intentions, dissuade students from participating even more, thereby compounding the very problem the author is trying to tackle, i.e., lack of motivation? Constant testing has been associated with high student anxiety (Kaplan et al., 2005). Though, in this case, students are not being tested on their knowledge of the content but on their ability to sincerely ask questions about the information they do not know yet. An easy way to make students dislike programming is to put them under unnecessary stress (Goold & Rimmer, 2000).
- b) Many students could first be perplexed by the material because it is not explained at the outset. They may ask questions merely for the purpose of asking them, bringing very little to the conversation. The proposed intervention would be rendered ineffectual from the outset if asking questions merely turned into another task to finish.

Later in sections 3 and 4, these queries are investigated. In this study, the phrases participation and questions-asked are used interchangeably.

The rest of the paper is structured as follows. Section 2 touches upon the operational aspects of the technique and illustrates its working parts. Section 3 presents the preliminary results. Section 4 discusses the strengths and shortcomings of SDPI. Section 5 concludes the paper and briefly presents the foundations of future research.

## 2. Methodology

As a pilot, the research was conducted on a single section of the introductory programming course. Due to the unproven and counterintuitive nature of the SDPI, it was deemed too risky by the author to introduce it right from the beginning of the course. Therefore, the course was divided into two halves for this experiment. In the first half, students were taught with the conventional methods wherein the content was explained, and questions were solicited from the students. SDPI was introduced in the second half, during which only a piece of code

was shown to the students without any explanations. The approach can be summarized as follows:

- 1) Students will be shown a piece of code/content at the beginning of the class.
- 2) A certain amount of time is given to the students—generally two minutes to come up with questions about the content if they have any.
- 3) If the students have no questions, it is assumed that all the students know the content perfectly well. To test this, the instructor picks a random student and asks them question(s) about the content. This part of the process is critical as it conveys to the students that it is better to ask questions they have than to face questions from the instructor that they may not be able to answer.
- 4) As the questions start to flow, they are recorded on the source code as comments for future reference.
- 5) Once enough questions (usually 7-10) have been asked, or every student has asked at least one question, the instruction begins and is modeled around the questions. The questions have now become a tool to probe the content without having been offered any instruction.

### 2.1 Student Population

The student population of our department comprises of both traditional and non-traditional students, though the terms are not well defined in the literature. For this work, the author defines “traditional” as full-time students who are recent high school graduates. Non-traditional students are those who have full-time jobs, are part-time students, and/or are older and seek a new career for various reasons. The number of students in the group was 12. The course is mandatory for Computer Science (CSE) students but can be used as an elective for Information Technology (IT) majors. This group had 7 IT/CSE majors and five non-IT/CSE students.

### 2.2 Data Collected

Class participation has historically been measured in a multitude of ways. Coming to class or attendance has been a valuable metric for a long time because it has shown a strong correlation with student performance (Coldwell et al., 2008; Landin & Pérez, 2015; Romer, 1993; Teixeira, 2016; Zorio-Grima & Merello, 2020). Clickers and response cards have also been used (Christle & Schuster, 2003; Gardner et al., 1994; Stowell & Nelson, 2007). In the contemporary web environment, metrics like frequency of visited pages, course clicks, email conversations, and discussion boards are also considered to measure active participation (Bekkering & Ward, 2021; Romero et al., 2013). The performance of students has been measured using a variety of criteria. The most commonly used items are course grades (Teixeira, 2016) and cumulative GPA (Bekkering & Ward, 2021). Pre- and post-quiz scores (Omar et al., 2009) and student ratings (Felisoni & Godoi, 2018) are occasionally employed.

For this work, the following data were collected for each participating student before and after the introduction of SDPI:

- 1) Number of questions asked in class
- 2) Number of email/Slack contacts with the instructor
- 3) Quiz scores

Java was employed as the programming language. There were eight modules taught. Lessons 1-4 covered the basics of Java, variables, conditional expressions, and loops. The SDPI

technique was used to teach modules 5-8 on methods, arrays, file operations, and search/sort, respectively. Then, data points were contrasted to see whether there were any notable differences.

### 2.3 Sample Snippet

Figure 2 presents an introductory snippet used at the beginning of module 7, i.e., file operations.

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.Scanner;
4 //Q1. What is File?
5 //Q2. Is the name of the file input.txt or file itself?
6 //Q3. Where is input.txt stored?
7 //Q4. Will the Scanner run as long as there is some input in the file?
8 //Q5. Will the loop stop after the last line is printed?
9 //Q6. What does throws clause do?
10 //Q7. Why file in Scanner not the file name?
11 //Q8.
12 //Q9.
13 public class Files {
14     public static void main(String[] args){
15         try{
16             File file = new File("input.txt");
17             Scanner in = new Scanner(file);
18             while(in.hasNext()){
19                 String temp = in.next();
20             }
21         }catch(FileNotFoundException e){
22             System.out.println("File Not Found");
23         }
24     }
25 }
```

**Figure 2: A Sample Code Snippet Used in SDPI**

The snippet was presented without the listed questions, and no explanations were provided to the students. As the students started asking questions, they were recorded in the source file. The instruction began only after 7 questions were asked. The file was shared with students during/after the class.

## 3. RESULTS

The study’s findings can be divided into two categories: quantitative data analysis to look at the possible effects of SDPI on student involvement, and outcomes and student perception of SDPI, as revealed by an end-of-course survey.

Each student’s total number of questions was kept track of during the synchronous online class. The average number of questions asked in class by all participating students before and after the SDPI technique was introduced is shown in Table 1. Appendix A has the entire matrix that details how many questions each student asked for each module.

As can be seen from the data presented in Table 1, 63% of the questions were asked by the top 25% of the students during the first half - the traditional mode of instruction. The bottom 41% asked only 9% of the total questions asked by all students, i.e., 41% of the students contributed to only 9% of all questions asked during the first half of the semester. This difference is statistically significant and captures the low participation rates among certain students. This is also consistent with the author’s experience teaching computer programming. The average number of questions asked by the whole class was 2.91.

Student No	Average No. of Questions Asked for Modules 1-4 (Before SDPI)	Average No. of Questions Asked for Modules 5-8 (With SDPI)	Delta
1	1	4	3
2	11.5	7.75	-3.75
3	2.5	4.25	1.75
4	6.25	6	-0.25
5	2.75	4.75	2
6	1.25	4.5	3.25
7	0.25	5.25	5
8	1	5	4
9	3.25	4.75	1.5
10	0.75	4.25	3.5
11	0	3.25	3.25
12	4.5	5.75	1.25
Average	2.91 (3.27) Qs	4.95 (1.15) Qs	2.04 (2.3)

**Table 1. Average Number of Questions Asked by Each Student in Class**

As measured by the number of questions posed by each student, class involvement increased dramatically with the use of SDPI. The average number of questions asked in class rose by around 69%. The quality of questions was not considered when collecting this data. Taking a closer look at the table prompts the following inquiries:

- 1) Why did the number of questions asked increase?
- 2) Was the increase uniformly distributed among students, or were the students who were active and asking questions before the SDPI intervention primarily responsible for the rise?
- 3) Did SDPI encourage non-participative students to participative?

The students had to ask questions to move forward through the content because it was not explained throughout SDPI. This is unquestionably one of the causes of the increase in the number of inquiries. That the students who did not participate in the first half felt at ease asking questions is another potential factor (see students no. 1, 7, 8, 10, and 11 in Table 1). The same 41% of students now account for 36% (9% previously) of the total questions asked. This is an increase of 27% among the formerly non-participating students. Thus, the increase in the number of questions asked was not due just to the previously participating students; rather, SDPI was able to enhance class involvement among the non-participating students. This is a crucial and positive development. This is also corroborated by the data collected with an end-of-course survey where most of the students indicated that SDPI made them more participative in class. An ANOVA (analysis of variance) analysis between the populations was conducted and is presented in Table 1. The difference was significant, with a p-value of 0.054. This indicates that, statistically, participation did increase in a significant way. However, although the results are significant, they should be considered cautiously because of the small sample size. See Appendix C for a complete ANOVA report.

Table 2 shows the average number of instructor contacts by students before and after the introduction of SDPI. The contacts

considered were through email and Slack (a team collaboration tool). Though the average number of contacts for the whole class increased from 1.78 to 2.29, the increase was not statistically significant.

Student No	Average No. of Email/Slack Contacts for Modules 1-4 (Without SDPI)	Average No. of Email/Slack Contacts for Modules 5-8 (With SDPI)
1	2	2.25
2	4.25	5
3	1	2
4	3.25	3.5
5	1	2.25
6	1.25	1.50
7	0.5	1
8	0	0
9	3.5	4
10	0.5	1
11	1.2	1.5
12	3	3.5
Average	1.78 (1.38)	2.29 (1.44)

**Table 2. Average Instructor Contacts by Students**

Student No	Average Quiz Scores for Modules 1-4 (Without SDPI)	Average Quiz Scores for Modules 5-8 (With SDPI)	Delta
1	18.75	23	4.25
2	15.25	20.75	5.5
3	18.75	21.75	3
4	20.25	19.75	-0.5
5	17	16	-1
6	21	23.25	2.25
7	13.75	15.75	2
8	13.5	15	1.5
9	19	23	4
10	17	18.25	1.25
11	21.25	23.5	2.25
12	20.5	23.25	2.75
Average	18.1 (2.7)	20.2 (3.25)	2.27 (1.86)

**Table 3. Average Quiz Scores**

Table 3 presents the average quiz scores of students before and after the introduction of SDPI. There were eight quizzes (four before and four after the introduction of SDPI) worth 25 points each. Ten out of twelve students improved their quiz scores average, and the class average rose by 2.1 points. This is a significant number in terms of the students whose scores improved. Still, the variance analysis (Appendix C) between the quiz scores reveals that the results are significant only at the 0.1 level. Hence, this is a mixed yet positive result because the modules taught with SDPI covered complex topics, and usually, the quiz scores tend to be lower in those modules.

Table 4 shows the change in the average number of questions asked and the corresponding change in average quiz scores for each student after the introduction of SDPI. It is

noteworthy that nine out of ten students who demonstrated greater participation also saw an improvement in their quiz results. Only one student with a lower average quiz score also increased their engagement.

Student No	Change in Average No. of Questions Asked	Change in Quiz Scores
1	3	4.25
2	-3.75	5.5
3	1.75	3
4	-0.25	-0.5
5	2	-1
6	3.25	2.25
7	5	2
8	4	1.5
9	1.5	4
10	3.5	1.25
11	3.25	2.25
12	1.25	2.75

**Table 4. Change in Quiz Scores vs. Change in Number of Questions Asked**

There is more anecdotal evidence that an increase in class participation may affect student outcomes. More iterations of SDPI need to be run to see if these results hold. Detailed statistical analysis can be found in Appendix C.

**3.1 End of Course Survey**

Regarding SDPI, an anonymous final survey was undertaken. Table 5 lists a few survey questions. The whole survey is in Appendix B.

Nearly all the students that took part said that SDPI increased their participation. This is encouraging and above the expectations of the author. There was always a possibility that SDPI might make students less participative due to the unconventional nature of instruction. Therefore, it is encouraging to see students embrace the new technique. After the implementation of SDPI, 90% of the students claimed that their grasp of the subject had likely improved, while 10% claimed there had been no change.

Curiosity is a learning catalyst (Kidd & Hayden, 2015; Szumowska & Kruglanski, 2020). Students generally acknowledged that SDPI increased their focus on the lecture material. Notably, 90% of the students said that SDPI increased their interest in the subject matter.

Students were asked a crucial question about their degree of stress when using SDPI. This was one of the survey’s most important questions since a stressful learning environment might decrease motivation and learning effectiveness (Bowers,

1986). If having to ask questions during class caused students stress, SDPI would immediately fail. It is interesting to note that 60% of respondents stated that SDPI decreased their stress levels, 20% said it had no effect, and 20% indicated it had increased their stress levels in the class. This is a reasonable distribution that fits within the practical restrictions of any new intervention. The causes of the elevated stress experienced by 20% of the students still need to be investigated further.

Students were also asked which method of instruction they preferred: traditional or SDPI. The SDPI technique was favored by 70% of the students, the traditional approach by 20%, and no preference was expressed by 10%. This is a positive sign for the future of this investigation and, in the author’s opinion, a mini vote of confidence in SPDI.

**4. DISCUSSION**

With such a small sample size, it is too early to generalize the utility of this technique, but the initial results reveal some interesting insights.

**4.1 Potential Strengths**

According to the classroom and assessment data collected and student responses to the survey, most students found SDPI beneficial, even though they perceived the technique to be counterintuitive. This is affirmed by the considerable increase in class participation that followed the implementation of SDPI. Students who did not ask questions during the first half of class began to do so during the second, as shown in Table 1. The non-participating students in the first half were more comfortable asking follow-up questions after the content was explained. The author believes that one reason for this change is the mitigation of the inadequacy factor among non-participating students. Many students do not participate because of the fear of other students’ judgment of their questions (Lin et al., 2017). This factor is mitigated by SDPI, as the content is not explained to begin with. Many students hesitate to raise questions after the introduction of a certain idea/concept because they believe their inquiries might be perceived as silly. The author believes SDPI offers students a wide and open range of questioning without making them feel inadequate.

Average quiz scores improved meaningfully after the introduction of SDPI, though not at the statistically significant level of 0.05. It is worth noting that the modules taught with SPDI cover complex concepts of methods, arrays, files, and sorting operations. Over the years, the author saw lower quiz scores in these modules by approximately 1.5% typically. That quiz scores improved despite the content’s increasing complexity is thus interesting and encouraging.

Question	Definitely Yes	Probably Yes	Might or Might Not	Probably Not	Definitely Not
1.Made you more participative	50%	50%	0%	0%	0%
2.Improved understanding of material	60%	30%	10%	0%	0%
3.Made you curious about the content	50%	40%	10%	0%	0%
4.Made you pay attention to the material	60%	40%	0%	0%	0%

**Table 5. End of Course Survey Responses**

Additionally, 90% of the participants whose engagement increased also increased their overall quiz results. Can an increase in the number of questions asked indicate higher quiz scores? It is too soon to say.

#### **4.2 Potential Challenges**

One of the challenges of SDPI is in-class data collection. The author had to record the number of questions asked by each student during every class. A slight interruption in the class flow usually occurred as the author marked the question in a matrix and simultaneously wrote the question on the source file for everyone to see on a shared computer screen. This is not a significant challenge, though. The data entry collection can be delegated to a teaching assistant, or it can be collated later if the lecture recording is available. The author intends to use these methods for the next iteration.

Getting the students used to the idea that their questions, not the instructor's, would determine the direction of the session is another issue. The author struggled for several classes to get everyone on board. Students are used to discussing the material in the classroom first and waiting for the teacher to respond to their queries. During a traditional lecture setting, the control primarily rests with the instructor, and students are aware of that mechanism. In SDPI, however, part of that control is given to the students by having them devise their own questions and steer the instruction in a specific direction. Getting comfortable with this shift of power is expected and should take time.

Another critical challenge for the current form of SDPI is that it does not have a mechanism to note the quality of questions asked by students. A simple question about a symbol in the source code is treated the same as a complex question about the feasibility/optimality of a code fragment. This is a major drawback of SDPI in its current form. In future iterations of SDPI, the author intends to create a weighting mechanism to classify the questions asked by students into categories representative of their complexity.

The instructor's time management in class and thorough material coverage are two more issues. The author typically plans their lessons in advance and is aware of how much material will be taught during the class. Covering the intended topic in the first two weeks of using SDPI was difficult because the instruction was based on the students' inquiries. The questions were broad in scope and frequently required time that could have been spent on other topics that day. As the author uses SDPI for a more classes, this problem might be mitigated.

#### **5. CONCLUSION AND FUTURE WORK**

SDPI, an experimental teaching method, was presented in this paper. The goal of the work was to compare the participation rates and quiz results before and after the implementation of SDPI. A significant increase (69%) in in-class participation, as measured by the number of questions asked, was reported. Additionally, an increase of 11% in average quiz scores was observed after the introduction of SDPI. Anecdotally, the results suggest that, by utilizing SDPI, it may be possible to influence students' participation during an introductory programming class. It would take many additional iterations of classes taught with SDPI to determine whether the relationship was causal. Therefore, the author plans to conduct this experiment with control and experimental groups for at least two more semesters. To have a wider variety of data for

comparison in future studies, the author plans to gather assignment, midterm, and final exam scores in addition to the quiz results.

Given the numerous challenges this system currently faces and the lack of data spanning multiple semesters, it would be premature to view the SDPI approach as a viable strategy for influencing student engagement. However, the preliminary findings are positive and offer a clear path for further study.

#### **6. REFERENCES**

- Alexandron, G., Armoni, M., Gordon, M., & Harel, D. (2014). Scenario-Based Programming: Reducing the Cognitive Load, Fostering Abstract Thinking. *Proceedings of the 36th International Conference on Software Engineering* (pp. 311-320).
- Allan, V. H., & Kolesar, M. V. (1997). Teaching Computer Science: A Problem Solving Approach That Works. *ACM SIGCUE Outlook*, 25(1-2), 2-10.
- Allred, C., & Swenson, M. (2006). Using Technology to Increase Student Preparation for and Participation in Marketing Courses: The Random Selector Model. *Marketing Education Review*, 16(1), 15-21.
- Armstrong, M., & Boud, D. (1983). Assessing Participation in Discussion: An Exploration of the Issues. *Studies in Higher Education*, 8(1), 33-44.
- Beaubouef, T. B., & J. Mason (2005). Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations. *Inroads – The SIGCSE Bulletin*, 37(2), 103-106.
- Beavis, C. (2010). Literacy, Learning, and Online Games: Challenge and Possibility in the Digital Age. *Proceedings of the IEEE 3rd International Conference on Digital Game and Intelligent Toy Enhanced Learning*. Piscataway, NJ: Institute for Electrical and Electronics Engineers.
- Bennedsen, J., & Caspersen, M. E. (2007). Failure Rates in Introductory Programming. *ACM SIGCSE Bulletin*, 39(2), 32-36.
- Berdine, R. (1986). Why Some Students Fail to Participate in Class. *Marketing News*, 20(1), 23-24.
- Bekkering, E., & Ward, T. (2021) Class Participation and Student Performance: A Follow-Up Study. *Information Systems Education Journal*, 19(4), 77-91.
- Bowers, J. W. (1986). Classroom Communication Apprehension: A Survey. *Communication Education*, 35(1), 372-378.
- Christle, C. A., & Schuster, J. W. (2003). The Effects of Using Response Cards on Student Participation, Academic Achievement, and On-Task Behavior During Whole-Class, Math Instruction. *Journal of Behavioral Education*, 12(3), 147-165.
- Credé, M., Roch, S. G., & Kieszczynka, U. M. (2010). Class Attendance in College: A Meta-Analytic Review of the Relationship of Class Attendance With Grades and Student Characteristics. *Review of Educational Research*, 80(2), 272-295.
- Crego, A., Carrillo-Diaz, M., Armfield, J., & Romero, M. (2016). Stress and Academic Performance in Dental Students: The Role of Coping Strategies and Examination-Related Self-Efficacy. *Journal of Dental Education*, 80(2), 165-172.

- Dongo, T., Reed, A. H., & O'Hara, M. (2016). Exploring Pair Programming Benefits for MIS Majors. *Journal of Information Technology Education: Innovations in Practice, 15*(1), 223-239.
- Duncan, K., Kenworthy, A. L., Mcnamara, R., & Kenworthy, D. A. (2012). The Effect of Synchronous and Asynchronous Participation on Performance in Online Accounting Courses. *Accounting Education, 21*(4), 431-449.
- Felisoni, D. D., & Godoi, A. S. (2018). Cell Phone Usage and Academic Performance: An Experiment. *Computers & Education, 117*(1), 175-187.
- Fritschner, L. M. (2000). Inside the Undergraduate College Classroom: Faculty and Students Differ on the Meaning of Student Participation. *The Journal of Higher Education, 71*(3), 342-362.
- Gardner, R., Heward, W. L., & Grossi, T. A. (1994). Effects of Response Cards on Student Participation and Academic Achievement: A Systematic Replication with Inner-City Students During Whole-Class Science Instruction. *Journal of Applied Behavior Analysis, 27*(1), 63-71.
- Goold, A., & Rimmer, R. (2000). Factors Affecting Performance in First-Year Computing. *SIGCSE Bulletin, 32*(2), 39-43.
- Howles, T. (2009). A Study of Attrition and the Use of Student Learning Communities in the Computer Science Introductory Programming Sequence. *Computer Science Education, 19*(1), 1-13.
- Hyde, C. A., & Ruth, B. J. (2002). Multicultural Content and Class Participation: Do Students Self-Censor? *Journal of Social Work Education, 38*(2), 241-256.
- Kaplan, D. S., Liu, R. X., & Kaplan, H. B. (2005). School Related Stress in Early Adolescence and Academic Performance Three Years Later: The Conditional Influence of Self-Expectations. *Social Psychology of Education, 8*, 3-17.
- Keller, J. M. (1983). Motivational Design of Instruction. In C. M. Reigeluth (Ed.), *Instructional-Design Theories and Models: An Overview of Their Current Status* (pp. 383-434). Lawrence Erlbaum Associates.
- Kidd, C., & Hayden, B. (2015). The Psychology and Neuroscience of Curiosity. *Neuron, 88*(3), 449-460.
- Kim, J., & Lerch, F. J. (1997). Why Is Programming (Sometimes) So Difficult? Programming as Scientific Discovery in Multiple Problem Spaces. *Information Systems Research, 8*(1), 25-50.
- Kinnunen, P., & Malmi, L. (2006). Why Students Drop Out CS1 Course? *Proceedings of the Second International Workshop on Computing Education Research* (pp. 97-108). New York, NY: ACM.
- Landin, M., & Pérez, J. (2015). Class Attendance and Academic Achievement of Pharmacy Students in a European University. *Currents in Pharmacy Teaching and Learning, 7*(1), 78-83.
- Lin, Y., Durbin, J. M., & Rancer, A. S. (2017). Perceived Instructor Argumentativeness, Verbal Aggressiveness, and Classroom Communication Climate in Relation to Student State Motivation and Math Anxiety. *Communication Education, 66*(3), 330-349.
- Majuri, J., Koivisto, J., & Hamari, J. (2018). Gamification of Education and Learning: A Review of Empirical Literature. *Proceedings of the 2nd International GamiFIN Conference* (2186, pp. 11-19).
- Margulieux, L. E., Catrambone, R., & Schaeffer, L. M. (2018). Varying Effects of Subgoal Labeled Expository Text in Programming, Chemistry, and Statistics. *Instructional Science, 46*, 707-722.
- Mendes, A. J., Paquete, L., Cardoso, A., & Gomes, A. (2012). Increasing Student Commitment in Introductory Programming Learning. *Frontiers in Education Conference (FIE)* (pp. 1-6). New York, NY: IEEE.
- Moguel, D. (2004). What Does It Mean to Participate in Class?: Integrity and Inconsistency in Classroom Interaction. *Journal of Classroom Interaction, 39*(1), 19-29.
- Newman, R., Gatward, R., & Poppleton, M. (1970). Paradigms for Teaching Computer Programming in Higher Education. *WIT Transactions on Information and Communication Technologies, 7*, 299-305.
- Nieuwoudt, J. E. (2020). Investigating Synchronous and Asynchronous Class Attendance as Predictors of Academic Success in Online Education. *Australasian Journal of Educational Technology, 36*(3), 15-25.
- Omar, A., Bhutta, M. K. S., & Kalulu, D. (2009). Assessment of Student Outcomes in Management Information Systems Online Course Participation. *Information Systems Education Journal, 7*(83), 3-10.
- Osatuyi, B., Osatuyi, T., & de la Rosa, R. (2018). Systematic Review of Gamification Research in Is Education: A Multi-Method Approach. *Communications of the Association for Information Systems, 42*, 95-124.
- Paas, F., Renkl, A., & Sweller, J. (2010). Cognitive Load Theory and Instructional Design: Recent Developments. *Educational Psychologist, 38*(1), 1-4.
- Robins, A. V., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172.
- Rogalski J., & Samurçay R. (1990). Acquisition of Programming Knowledge and Skills. In J. M. Hoc, T. R. G. Green, R. Samurçay, & D. J. Gillmore (Eds.), *Psychology of Programming* (pp. 157-174).
- Romer, D. (1993). Do Students Go to Class? Should They? *Journal of Economic Perspectives, 7*(3), 167-174.
- Romero, C., López, M.-I., Luna, J.-M., & Ventura, S. (2013). Predicting Students' Final Performance From Participation in Online Discussion Forums. *Computers & Education, 68*, 458-472.
- Seaborn, K., & Fels, D. I. (2015). Gamification in Theory and Action: A Survey. *International Journal of Human-Computer Studies, 74*, 14-31.
- Sheard, J., & Hagan, D. (1998). Our Failing Students: A Study of a Repeat Group. *ACM SIGCSE Bulletin, 30*(3), 223-227.
- Smith, D. H. (1992). Encouraging Students' Participation in Large Classes: A Modest Proposal. *Teaching Sociology, 20*(4), 337-339.
- Stowell, J. R., & Nelson, J. M. (2007). Benefits of Electronic Audience Response Systems on Student Participation, Learning, and Emotion. *Teaching of Psychology, 34*(4), 253-258.
- Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science, 12*(2), 257-285.
- Sweller, J. (1994). Cognitive Load Theory, Learning Difficulty, and Instructional Design. *Learning and Instruction, 4*(4), 295-312.

- Szumowska, E., & Kruglanski, A. (2020). Curiosity as End and Means. *Current Opinion in Behavioral Sciences*, 35, 35-39.
- Teixeira, A. A. C. (2016). The Impact of Class Absenteeism on Undergraduates' Academic Performance: Evidence From an Elite Economics School in Portugal. *Innovations in Education and Teaching International*, 53(2), 1-13.
- Wade, R. (1994). Teacher Education Students' Views on Class Discussion: Implications for Fostering Critical Reflection. *Teaching and Teacher Education*, 10(2), 231-243.
- Watson, C., & Li, F. W. (2014). Failure Rates in Introductory Programming Revisited. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (pp. 39-44). New York, NY: ACM.
- Weaver, R. R., & Qi, J. (2005). Classroom Organization and Participation: College Students' Perceptions. *The Journal of Higher Education*, 76(5), 570-601.
- Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education*, 12(3), 197-212.
- Zorio-Grima, A., & Merello, P. (2020). Class Attendance and Online-Tests Results: Reflections for Continuous Assessment. *Journal of Teaching in International Business*, 31(1), 75-97.

#### AUTHOR BIOGRAPHY

**Deepak Dawar** is an assistant professor of computer and information technology in the CIT



department at the Miami University, Ohio. He joined Miami in 2016 and has taught an ensemble of computer science & I.T. related courses, including Java, Visual Basic, Operating Systems, and Computer Networking, to name a few. His research interests include

evolutionary computing, swarm intelligence, pattern recognition, and video-based vehicle classification. His current scholarship focuses on innovation in teaching computer programming and the application of artificial intelligence (A.I.) to the delivery of I.T. courses.

**APPENDICES**

**Appendix A. Raw Data for Number of Questions Asked by Each Student During the Course**

Student No.	Module1	Module2	Module3	Module4	Module5	Module6	Module7	Module8	Average No. of Questions Asked Before SDPI	Average No. of Questions Asked After SDPI	Delta
1	1	1	0	2	3	4	5	4	1	4	3
2	15	9	10	12	11	8	7	5	11.5	7.75	-3.75
3	3	4	2	1	3	5	4	5	2.5	4.25	1.75
4	10	6	8	1	5	6	7	6	6.25	6	-0.25
5	4	3	2	2	4	4	5	6	2.75	4.75	2
6	2	1	1	1	4	5	5	4	1.25	4.5	3.25
7	0	0	1	0	3	6	7	5	0.25	5.25	5
8	0	0	2	2	5	6	4	5	1	5	4
9	5	3	3	2	6	5	5	3	3.25	4.75	1.5
10	1	0	1	1	4	5	4	4	0.75	4.25	3.5
11	0	0	0	0	2	4	3	4	0	3.25	3.25
12	5	4	4	5	6	6	7	4	4.5	5.75	1.25

**APPENDIX B. Survey Instrument for SDPI**

Q1 Did the Student-Driven Probe Instructional Approach (SDPI) make you more participative in the class?

- Definitely yes. It made me more participative. (1)
  - Probably yes (2)
  - Might or might not (3)
  - Probably not. I avoided asking questions. (4)
  - Definitely not (5)
- 

Q2 During SDPI make you feel confident about asking opening questions?

- Definitely yes. I was confident since I could ask any question about the content. (1)
  - Probably yes (2)
  - May be (3)
  - Probably not. I avoided asking questions. (4)
  - Definitely not (5)
- 

Q3 What impact did SDPI have on your stress levels in class?

- It definitely reduced my stress levels. I felt free to ask any type of questions since nothing was explained about the content, to begin with. (1)
  - It probably reduced my stress levels. (2)
  - It had no impact on my stress levels. (3)
  - It increased my stress levels. (4)
- 

Q4 Did the SDPI approach improve your understanding of material?

- Definitely yes. It made me think deeply about the content since I was the one asking the opening questions. (1)
  - Probably yes (2)
  - Might or might not (3)
  - Probably not (4)
  - Definitely not (5)
- 

Q5 Did the SDPI approach make you more curious about the content taught in class?

- Definitely yes. By looking at the content that was not explained, I became curious about the content. (1)
  - Probably yes (2)
  - Might or might not (3)
  - Probably not (4)
  - Definitely not (5)
- 

Q6 Did the SDPI approach made you pay attention to the material being presented?

- Definitely yes (1)
  - Probably yes (2)
  - Might or might not (3)
  - Probably not (4)
  - Definitely not (5)
- 

Q6 Given an option, what mode of instruction would you prefer for this course?

- The SDPI approach wherein the instructor shows you material, and let you begin asking questions to accommodate everyone's questions and curiosity levels. (1)
  - The traditional approach wherein the instructor explains the content, and then they proceed to ask you questions about the content just explained. (2)
  - No preference (3)
- 

Q7 According to you, what changes should be made to the SDPI format to improve it further?

---

---

Appendix C. ANOVA Results

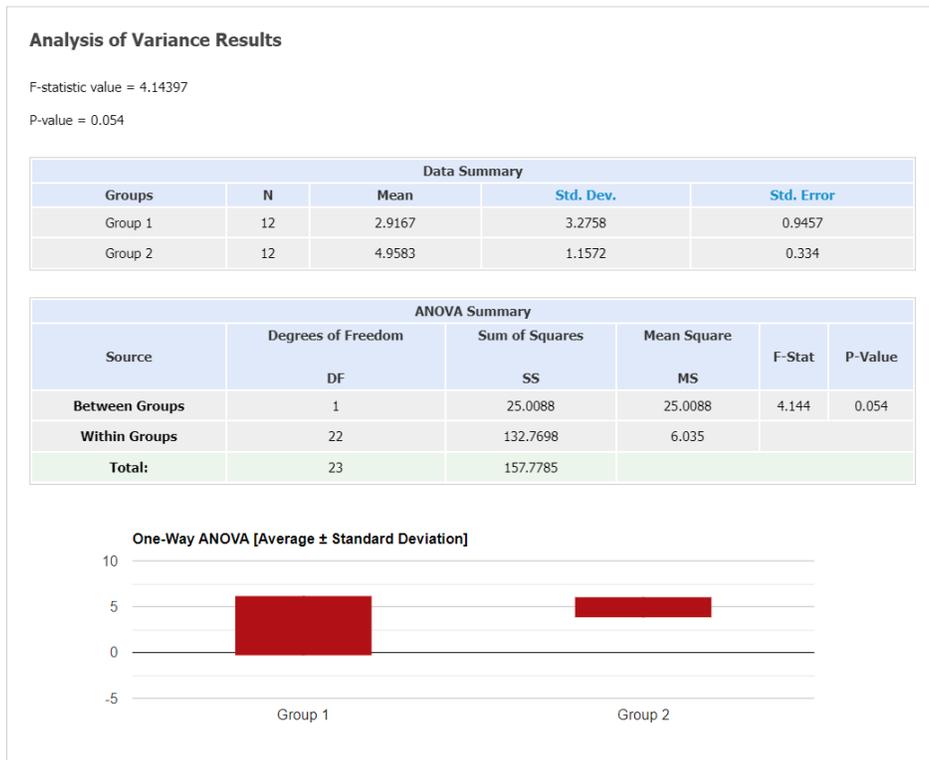


Figure C-1. ANOVA for Number of Questions Asked Before and After SDPI

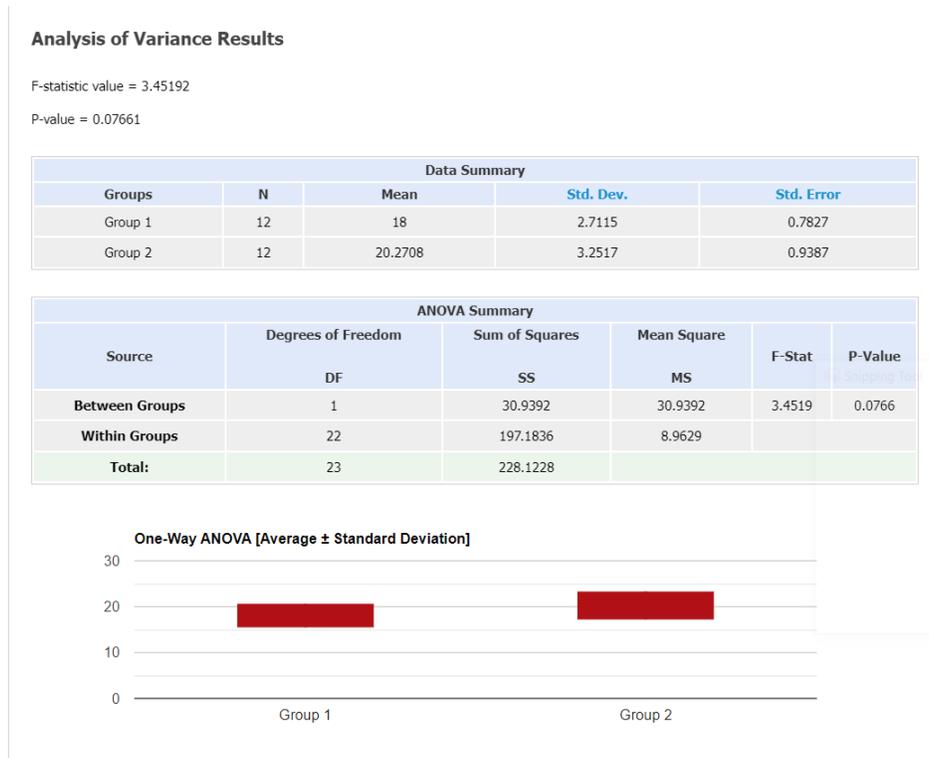


Figure C-2. ANOVA for Quiz Scores Before and After SDPI

#### **Appendix D. Sample Student Questions**

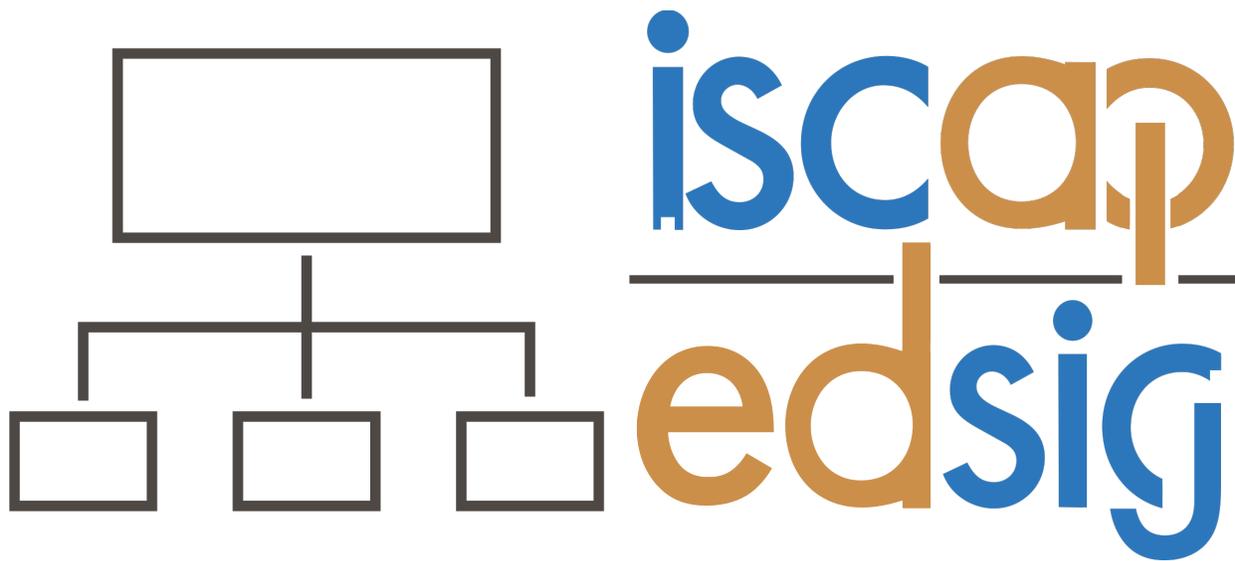
Example student questions – These are only a part of the questions collected and are collected from different modules. They are provided here to show what kind of questions asked when content wasn't explained to them.

//Q1. How about we remove @ on line 16?  
//Q2. How are the parameters transferred to methods?  
//Q3. Why is main not capitalized?  
//Q4. What is Character.isdigit()  
//Q5. Is /\*\* mandatory in method comments?  
//Q6. What if length is double in main and int in AreaOfRectangle()?  
//Q7. What is static?

//Q1. Where does the square method return to?  
//Q2. Will it compile?  
//Q3. sq is main, how can we have sq in square?  
//Q4. Why do calculation this way?  
//Q5. What does return mean in square?  
//Q6. What's int doing before square?  
//Q7. Why don't you need anything inside the () after square?  
//Q8. How do we use return?

//Q1. What is that 20?  
//Q2. What is the bracket doing there?  
//Q3. Why do you have assigned the values manually?  
//Q4. What us the new keyword?  
//Q5. Can we start the sequence form 1 or is it set at 0?  
//Q6. Can we store doubles or other data types in int array?

//Q1. Are < less than or greater sign?  
//Q2. Why A is caps?  
//Q3. Can we replace String with any data type?  
//Q4. Why do we not need () on right but not on left?  
//Q5. What is the max size of ArrayList?  
//Q6. Why import ArrayList?  
//Q7. What is line 14 doing?  
//Q8. Result of printing empty ArrayList?



**Information Systems & Computing Academic Professionals  
Education Special Interest Group**

**STATEMENT OF PEER REVIEW INTEGRITY**

All papers published in the *Journal of Information Systems Education* have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

Copyright ©2023 by the Information Systems & Computing Academic Professionals, Inc. (ISCAP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to the Editor-in-Chief, *Journal of Information Systems Education*, [editor@jise.org](mailto:editor@jise.org).

ISSN: 2574-3872 (Online) 1055-3096 (Print)