Active and Collaborative Learning Based Dynamic Instructional Approach in Teaching Introductory Computer Science Course with Python Programming

Md Mahmudur Rahman, Monir H. Sharker and Roshan Paudel
Morgan State University, Baltimore, Maryland
md.rahman, monir_sharker, roshan_paudel@morgan.edu.edu

***Abstract*—In this era of smart devices, new technologies, gadgets, apps, and numerous systems and services available over online, teaching an introductory programming course by traditional lecture method faces challenges to draw student's attention; especially in their freshman year. In this work, we discuss our experience in teaching an introductory CS course by infusing both interactive and collaborative learning in pedagogy so that students can learn using interactive platforms, tools, technologies, systems, and services as available to them and collaboration within and among groups. For interactive learning, students used an interactive programming environment (e.g. *repl.it* classroom) as well as online eBooks. We designed several in-class exercises, assignments, small lab-based projects with example codes and expected outputs, and unit tests by using built-in unit tests library. We also, in the middle of semester, introduced collaborative learning through teamwork on well-defined projects during the learning time and submitted at the end. The collaborations include use of basic task management tools and multi-player tool of repl.it that the students can critic, supplement, improve peer works and learn. To evaluate the impact of this infusion, a pre- and post-survey were conducted on student cohort in two different semesters. The initial evaluation of the survey results and performances (final project and final grades) show evidence to conclude that the proposed pedagogical approach increased student motivation and engagement and facilitated learning to entry-level computer science students.**

Keywords – introductory programming, interactive learning, active learning, collaborative learning, retention.

I. Introduction

Learning is a cognitive phenomenon that works differently for different individuals. Even, for the same individual, different learning methods become effective at different ages, circumstances, for different subjects, and pedagogies. Teaching computer programming in undergraduate level using traditional approaches such as lecture, lab, homework, grading faces several challenges especially in this age of internet, smart devices, technologies, gadgets, apps, and numerous systems and services available. Drawing student's attention (especially in freshman and sophomore years), transferring the knowledge itself, relating the programming knowledge to real life applications, student success in terms of grading and graduation are to mention a few of such challenges. Introductory Programming course is among the most challenging courses in the Computer Science (CS) curriculum, particularly for those who are new in programming. Many students get panicked by the core concepts and extents of this very first programming course, perform poorly, and eventually quit CS program [1]. In traditional approach, despite different efforts to make introductory programming languages easy to comprehend, many beginners still struggle with the syntax and semantics of this course. The failure rates in introductory programming courses are as high as 30% [2] and most drop out students in CS major drop after taking those gate-keeper courses. Consequently, a paradigm shift in teaching and learning by infusing effective methods in pedagogy to teach introductory programming clearly needs attention.

Educational research community proposed numerous active and collaborative learning-based techniques to teach in introductory programming courses effectively [3, 5]. The approach of traditional education and learning has been transforming with the advent of massive open online courses (MOOC) as used in the United States and JMOOC (Japan MOOC) in the form of flipped classroom [6-8]. Also, in the evolution of autonomous robot, human-robot interaction is used for teaching through body language, gestures, facial expressions, gaze, touching, and computational power [9–13]. But, given the outcomes, the effort surely needs more effective pedagogy. This paper investigates the impact of infusing interactive and collaborative learning through available resources in pedagogy. This is an experimental testbed where students can learn using interactive platforms, tools, technologies, systems and services as available to them and through collaboration within and among random groups. For interactive learning, students used interactive programming environment (e.g. repl.it classroom [14]) and interactive eBooks through runestone project [15]. In interactive programming, we designed several in-class exercises, assignments, small lab-based projects including instructions with example codes and expected outputs, and unit tests by using built-in unit tests library (e.g. Python). Interactive eBooks offers animation and software visualization tool where students can step through code line-by-line (forward or backward) and a program editing and execution area where students can execute examples, change and execute the updated code, and visualize the output in their own devices and platforms. Accordingly, in recent years, efforts to improve the teaching of computer programming to freshmen students with active and interactive learning is getting a lot of attention from researchers [2, 4, 16, 17].

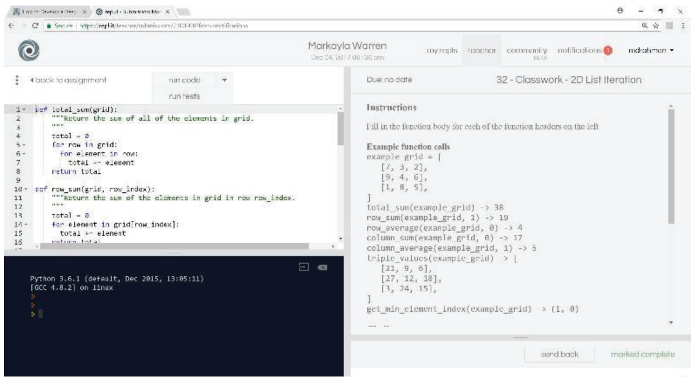
Collaborative learning is a predominant form of active learning focusing on the role of peer work for educational success [3]. Working collaboratively is one of the most important skills for programmer as, in real life, a programmer needs to work with other programmers in industry setting [18]. Working collaboratively in academic setups can be helpful for students acquiring better learning experiences compared to working individually. Collaborative learning is introduced around mid-semester, while students are already taught programming concepts, through teamwork (e.g. group of 2-3 students) on well-defined project where they learn during the rest of the semester and submit completed project works at the end. The collaborations include use of basic task management tools, platforms, and multiplayer tool so that the students can critic, supplement, improve peer works and learn through the process.

There is a significantly large number of articles published and techniques implemented based on active and collaborative learning from educational research community to address the issue in introductory programming courses [8,10]. Sometimes, a combination of approaches might enhance the classroom teaching environment. Hence, instead of "reinventing the wheel", we implemented a combined instructional approach by infusing both interactive and collaborative learning in teaching an introductory CS course (Introduction to Computer Science I, COSC 111) in Python.

In this paper, we describe our experience of instructing this course in Fall 2018 and Spring 2019 in seven different sections by three different instructors. To evaluate the impact of this infusion of interactive and collaborative learning pedagogy on student learning outcomes through robust statistical analysis, a pre- and post-survey were conducted on student cohort. The null hypothesis is, H0: There is no positive impact of infusing interactive and collaborative learning in teaching introductory programming. The evaluation, data analysis, and results complement us to conclude that the proposed approach increases student engagement, facilitates learning, increased student motivation, and contributed to student progress by reducing failure rate and improving retention rate in introductory programming course as well as in other related courses in CS.

II. INFUSING INTERACTIVE LEARNING

For a dynamic classroom, we used an interactive computer programming and development environment ***repl.it*** as a top level shell, where the name comes from the read-eval-print loop. It takes single user inputs (i.e. single expressions), evaluates them, and returns the result to the user for basically every language [11]. The idea behind ***repl.it*** is to let students focus on coding, collaborate, and build and minimizing the time spent acquiring, setting up, supporting, debugging, or adding a third-party package to an IDE. Once completes the registration, students can create an environment with a click [19].



```
def total_sum(grid):
    """Return the sum of all of the elements in grid.
    """
    total = 0
    for row in grid:
      for element in row:
        total += element
    return total

def row_sum(grid, row_index):
    """Return the sum of the elements in grid in row row_index.
    """
    total = 0
    for element in grid[row_index]:
      total += element
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
>
>
```

Instructions

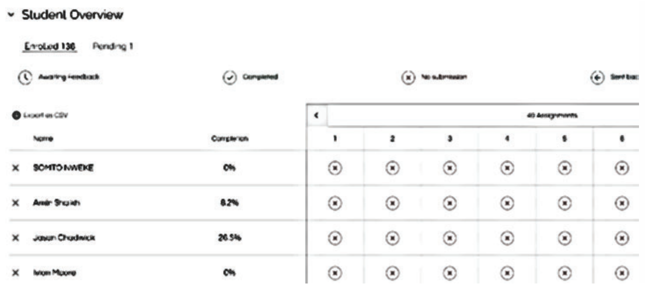
Fill in the function body for each of the function headers on the left

Example function calls

```
example_grid = [
   [7, 3, 2],
   [9, 4, 6],
   [1, 8, 5],
]
total_sum(example_grid) -> 38
row_sum(example_grid, 1) -> 19
row_average(example_grid, 0) -> 4
column_sum(example_grid, 0) -> 17
column_average(example_grid, 1) -> 5
triple_values(example_grid) -> [
   [21, 9, 6],
   [27, 12, 18],
   [3, 24, 15],
]
get_min_element_index(example_grid) -> (1, 0)
```

Fig. 1. User Interface of *repl.it*

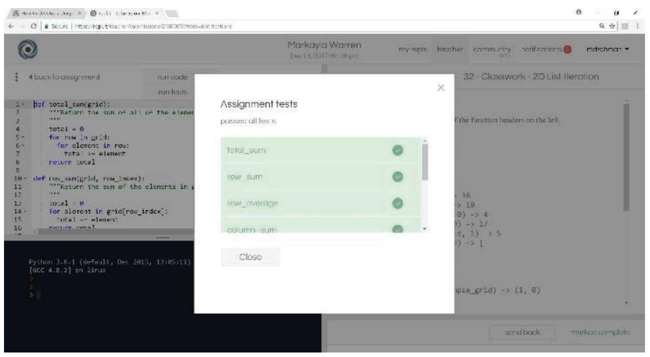
When students select an environment, the user interface (Fig. 1) splits between text editor (top left), written instructions (right), and an interpreter on the terminal (bottom left). ***repl.it*** Classroom is the application of this incredibly robust technology to a classroom setting. In this setting, students are invited to join the virtual classroom by email address or invitation code, allowed to collaborate with teacher and fellow students, and assignments are created and automatically evaluated by Input/output matching and unit testing. This auto-grading tests provides students immediate feedback on their code submissions to help them improve their works [12].



	Name	Completion	1	2	3	4	5	6
×	SOMTO NWEKE	0%	⊗	⊗	⊗	⊗	⊗	⊗
×	Amir Shaikh	8.2%	⊗	⊗	⊗	⊗	⊗	⊗
×	Jason Chadwick	26.5%	⊗	⊗	⊗	⊗	⊗	⊗
×	Ivion Moore	0%	⊗	⊗	⊗	⊗	⊗	⊗

Fig. 2. Monitoring students' progress in teacher dashboard

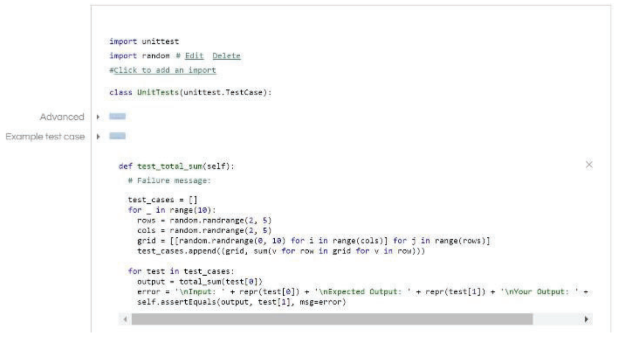
repl.it classroom works on any computer since it is totally browser based making it perfect for BYOD (bring-your-own-device) environments or schools where students use different available computers. It includes a handful of management tools like a teacher dashboard that provides ability to monitor progress of students (Fig. 2), interface for creating and submitting assignments, notifications, and more.



```
def total_sum(grid):
    """Return the sum of all of the elemen
    """
    total = 0
    for row in grid:
      for element in row:
        total += element
    return total

def row_sum(grid, row_index):
    """Return the sum of the elements in r
    """
    total = 0
    for element in grid[row_index]:
      total += element
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

Fig. 3. A snapshot of running unit testing for assignment



```
import unittest
import random # Edit  Delete
#Click to add an import

class UnitTests(unittest.TestCase):

  def test_total_sum(self):
    # Failure message:

    test_cases = []
    for _ in range(10):
      rows = random.randrange(2, 5)
      cols = random.randrange(2, 5)
      grid = [[random.randrange(0, 10) for i in range(cols)] for j in range(rows)]
      test_cases.append((grid, sum(v for row in grid for v in row)))

    for test in test_cases:
      output = total_sum(test[0])
      error = '\nInput: ' + repr(test[0]) + '\nExpected Output: ' + repr(test[1]) + '\nYour Output: ' +
      self.assertEquals(output, test[1], msg=error)
```

Fig. 4. Creating unit test for an assignment

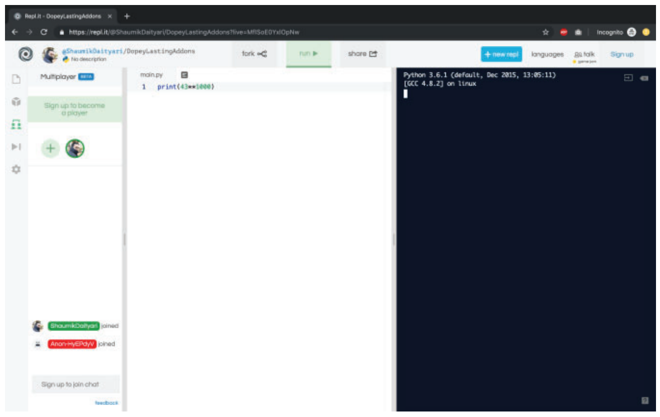
We created several in-class exercises, assignments, and small lab-based projects by creating instructions with example codes, expected outputs, and unit tests (Fig. 3) by using the built-in ***unit tests*** library of Python 3 (Fig. 4) for each problem which is automatically evaluated based on input/output matching and unit test settings. Students were engaged to do in-class exercises in this environment for the last 20 minutes immediately after discussing a topic/concept in our 50 minutes lecture. In our approach, the students spend more time actively engaged in small group exercises and interactive demonstrations instead of listening to a traditional lecture. In this context, the value of active learning is realized in our classroom through interactive lecturing styles.

We also used interactive online book from runestone project [15] and code visualization webpage (http://pythontutor.com/) which offer animation and software visualization tool where students can step through code line-by-line(forward or backward) and a program editing and execution area where students can execute examples, change and execute the updated code, and visualize the output in their own devices and platforms.

III. Infusing Collaborative Learning

Collaborative learning is another predominant form of active learning focusing on the role of peer work for educational success [8]. Usually in a four-year CS curriculum, collaborative effort and teamwork is done only in the later years, i.e., in the third and fourth year mainly in Software Engineering-related courses. Even then the collaboration is driven more by division of labor due to size rather than complexity. Many a times, assignment or project is designed after forming the groups in the classroom, when the teacher knows the strength of a particular group. Such approaches do not necessarily develop teamwork skills. Researchers have felt the need for a way to facilitate students to work together with clearly defined boundaries [18].

Today, collaborative programming environments that run in the browser have become more popular than ever. Having such an environment removes any set-up necessary for pair-programming. The ***repl.it* multiplayer** feature helped in providing the collaborative 'team-based' learning environment intended for the students as this feature makes it possible for all members of the group to code on the same editor and execute their programs in the same interpreter.


Repl.it - DopeyLastingAddons
https://repl.it/@ShaumikDaityari/DopeyLastingAddons?live=MflSoE0YxlOpNw
Incognito
@ShaumikDaityari/DopeyLastingAddons
No description
fork
run
share
+ new repl
languages
talk
Sign up
Multiplayer BETA
Sign up to become a player
main.py
ShaumikDaityari joined
Anon-HyEPdyV joined
Sign up to join chat
feedback

```
print(43**1000)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

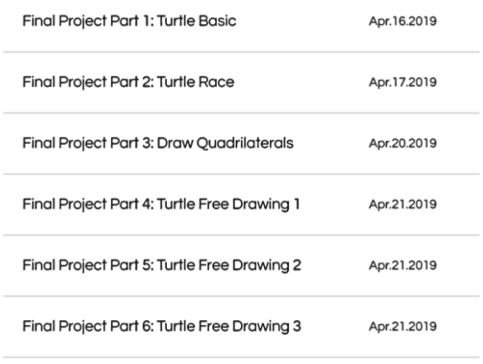
Fig. 5. Interface for *repl.it* multiplayer feature

The students took advantage of this and could work together as a team at their convenience without having to physically have meetings with group members outside the class environment (Fig. 5). The fact that everyone shares the same compute infrastructure means they all see the same errors and the same output, which is crucial for collaboration [19]. After activation of this multiplayer feature, students can create a URL with a potential collaborator (group member) to either contribute to or view the status of the project in real time.

A. Final Team Projetcs

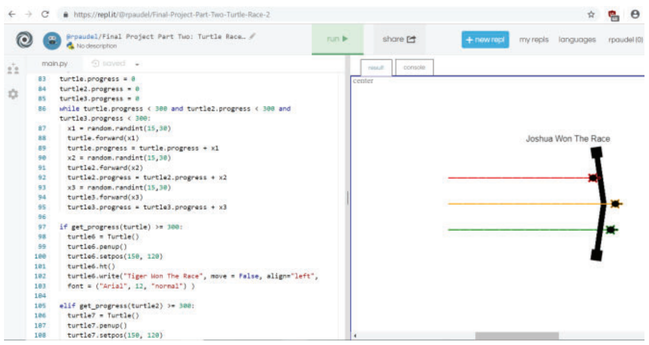
We introduced collaborative learning as early as in this introductory CS course in the form of a final team (group of 3 students) project based on Turtle graphics library submitted at the end of the semester. The students had to undertake the project based on all the Python programming concepts learned throughout the semester, which mainly includes Arithmetic Operations, Loops, Lists, and Functions.

During the middle of the semester, students were given a final project on Turtle graphics. There were 6 total mini projects: 3 instructional drawings and 3 free drawing type. Students were encouraged to be more creative in solving open ended problems. Each group is formed by 3 students and they are required to work in their own group. First 3 problems every member of the group must attempt. Last 3, we let the member within the group choose one and complete all three as a group. Below is the list of projects posted in repl.it (Fig. 6).



Final Project Part 1: Turtle Basic	Apr.16.2019
Final Project Part 2: Turtle Race	Apr.17.2019
Final Project Part 3: Draw Quadrilaterals	Apr.20.2019
Final Project Part 4: Turtle Free Drawing 1	Apr.21.2019
Final Project Part 5: Turtle Free Drawing 2	Apr.21.2019
Final Project Part 6: Turtle Free Drawing 3	Apr.21.2019

Fig. 6. List of mini project parts under final project posted in *repl.it*



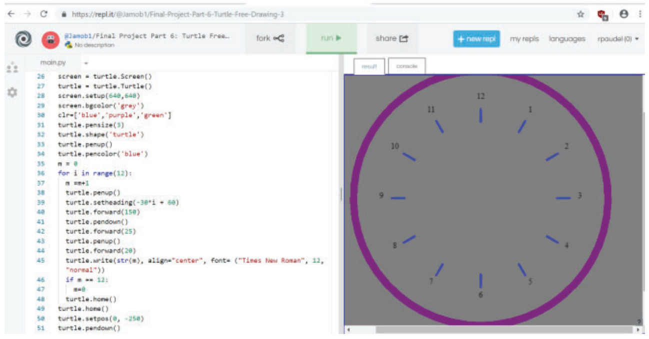
```
turtle.progress = 0
turtle2.progress = 0
turtle3.progress = 0
while turtle.progress < 300 and turtle2.progress < 300 and
turtle3.progress < 300:
  x1 = random.randint(15,30)
  turtle.forward(x1)
  turtle.progress = turtle.progress + x1
  x2 = random.randint(15,30)
  turtle2.forward(x2)
  turtle2.progress = turtle2.progress + x2
  x3 = random.randint(15,30)
  turtle3.forward(x3)
  turtle3.progress = turtle3.progress + x3

if get_progress(turtle) >= 300:
  turtle6 = Turtle()
  turtle6.penup()
  turtle6.setpos(150, 120)
  turtle6.ht()
  turtle6.write("Tiger Won The Race", move = False, align="left",
  font = ("Arial", 12, "normal") )

elif get_progress(turtle2) >= 300:
  turtle7 = Turtle()
  turtle7.penup()
  turtle7.setpos(150, 120)
```

https://repl.it/@rpaudel/Final-Project-Part-Two-Turtle-Race-2
@rpaudel/Final Project Part Two: Turtle Race...
No description
run
share
+ new repl
my repls
languages
main.py
saved
result
console
center
Joshua Won The Race

Fig. 7. Sample student submission of Final Project Part 2 (Turtle Race)

Part 1 (Turtle Basic) of the project helps students to understand how to repeat same behavior by using loop. In Part 2 (Turtle Race), Students are going to create 3 different turtles and use while loop to race them (Fig. 7 shows a sample of student submission). Part 3 (Draw Quadrilaterals) provides students more practice on loop and function (write a function once and call it many times).

Final three parts of project are made free drawing (open ended) type and as a group they must finish all three of them. It is expected that students are going to apply all the techniques learnt in the class and come up with some meaning full drawing not just random stuff as shown in Fig. 8 of a sample submission by a group.



```
screen = turtle.Screen()
turtle = turtle.Turtle()
screen.setup(640,640)
screen.bgcolor('grey')
clr=['blue','purple','green']
turtle.pensize(3)
turtle.shape('turtle')
turtle.penup()
turtle.pencolor('blue')
m = 0
for i in range(12):
  m =m+1
  turtle.penup()
  turtle.setheading(-30*i + 60)
  turtle.forward(150)
  turtle.pendown()
  turtle.forward(25)
  turtle.penup()
  turtle.forward(20)
  turtle.write(str(m), align="center", font= ("Times New Roman", 12,
  "normal"))
  if m == 12:
    m=0
  turtle.home()
turtle.home()
turtle.setpos(0, -250)
turtle.pendown()
```

https://repl.it/@Jamob1/Final-Project-Part-6-Turtle-Free-Drawing-3
@Jamob1/Final Project Part 6: Turtle Free...
No description
fork
run
share
+ new repl
my repls
languages
main.py
result
console
12
1
2
3
4
5
6
7
8
9
10
11

Fig. 8. Sample student submission of Turtle Free Drawing

The essence of these projects was to establish collaborative learning, and this was achieved as it is noticed that the group members divided tasks among themselves and had times where they had to brainstorm together in solving some of the challenges they faced during their development. After the two weeks of development, the projects were displayed on the screen and presented by the group members who took the class through the logic behind their program.

IV. RESULTS AND DISCUSSIONS

For evaluation and result analysis, five sections of COSC 111 in Fall 2018 and two sections of COSC 111 in Spring 2019 are included in this study which were offered in a consistent manner (similar course materials and instructional approach) by three different CS instructors. A total of 168 students were enrolled in across these sections. Data available across two semesters were:

A. *Student self-report survey administered near the beginning of the course.*
B. *Student self-report survey administered near the end of the course which in part parallels the early course survey.*
C. *Final project scores and final grades.*

The result is focused on student perceptions of various pedagogical techniques, what they did or did not find helpful, their belief about how helpful the course was in developing their coding skills and, finally, the impact of the course on their intention to enroll in further computer science courses. In Fall 2018, a total of 116 students were registered for the course across 5 sections. Of that total, n = 92 completed the initial survey, for almost 80% response rate and 66 students completed the final survey. Final grades were available for 63 students. For some analyses in this report, only the n = 63 for whom grades, and survey responses could be matched are included. For Spring 2019, a total of 52 students were registered for the course in two sections. Of that total, n = 33 completed the initial survey, and n = 18 completed the final survey. Grades for the Spring, 2019 students were unavailable.

Analysis suggests that the two samples were not significantly different in the amount of coding experience that they brought to the course (F (1) = 0.18; p = 0.672).

In both groups, the majority were male (73.9% in F2019, 60.6% in Sp2019) and first-year students (72.8% in F2018, 51.5% in Sp2019). Most reported their high school grade average to be in the 'B' range (64.1% in F2018, 66.7% in Sp2019). Most also reported not having completed any computer science courses in high school (53.3% in F2018, 42.4% in Sp2019). To assure that conclusions drawn from the final survey responses were representative, the two samples were compared (initial survey vs final survey) using class status and their rating of how much programming knowledge they had as they began the course.

TABLE 1. STUDENT GLOBAL PERCEPTIONS OF COURSE EXPERIENCE

	F2018 Final N=66	S2019 FINAL N=18
Having completed this course, I now feel that I can program or solve small problems in Python		
Strongly Disagree	0.0%	16.7%
Disagree	3.0%	11.1%
Neutral	28.8%	5.6%
Agree	25.8%	55.6%
Strongly Agree	42.4%	11.2%
As a result of my experiences in this class, I plan to take more computer science courses.		
Strongly Disagree	10.6%	27.8%
Disagree	4.5%	16.7%
Neutral	10.6%	16.7%
Agree	28.8%	11.1%
Strongly Agree	45.5%	27.8%
Overall, the course reduced my fear of programming		
Strongly Disagree	7.6%	11.1%
Disagree	7.6%	16.7%
Neutral	30.3%	38.9%
Agree	24.2%	11.1%
Strongly Agree	30.3%	22.2%
This class reduced my interest in computer science		
Strongly Disagree	56.1%	27.8%
Disagree	19.7%	27.8%
Neutral	9.1%	22.2%
Agree	7.6%	11.1%
Strongly Agree	7.6%	11.1%

Overall, students who completed the final survey reported favorable experiences. These perceptions are summarized in Table 1. The majority of those who responded *agreed* or *strongly agreed* that they felt capable of programming in Python (86.2% in F2018, 54.6% in Sp2019), and that they planned to take more computer science courses (74.3% in F2018, 39.9% in Sp2019). Most students also *agreed* or *strongly agreed* that the course reduced their fear of programming (54.5% in F2018, 33.3% in Sp2019), though more students selected the *neutral* option (38.9%) in the Sp2019 sample. And finally, the majority *disagreed* or *strongly disagreed* that the course reduced their interest in computer science (75.8% in F2018, 57.5% in Sp2019).

In terms of student ratings on various pedagogical approaches used in the course, overall, students rated class exercises done in repl.it favorably, as well as, class discussion to clarify programming concepts. Most students also *agreed* or *strongly agreed* that they got help from the TAs and got help with programming assignments. Finally, the majority of students also rated the following course materials somewhat *useful* or *very useful*: lecture slides (86.4% F2018; 66.7% Sp2019), *repl.it* classwork/homework (89.4% F2018, 77.8% Sp2019), lab slides and problems (89.4% F2018, 77.8% Sp2019), *repl.it*/Google Docs Notes (77.2% F2018, 66.4% Sp2019).

TABLE 2. STUDENT ATTITUDES TOWARD EBOOKS VS HARD COPY

	F2018 Initial N=92	F2018 Final N=66	SP2019 Initial N=33	SP2019 Final N=18
I prefer to read a textbook in hard copy rather than eBook on a screen				
Strongly Disagree	17.4%	43.9%	21.2%	55.6%
Disagree	16.3%	18.2%	30.3%	11.1%
Neutral	37.0%	27.3%	18.2%	22.2%
Agree	17.4%	9.1%	21.2%	5.6%
Strongly Agree	12.0%	1.5%	9.1%	5.6%

Student attitudes toward using an eBook or reading from a computer screen seems to have shifted in a more favorable direction over the duration of the course. As Table 2 shows, in the initial surveys 29.4% (F2018) and 30.2% (Sp2019) *agreed* or *strongly agreed* with the statement "I prefer to read a textbook in hard copy rather than an eBook on a screen."
For the final survey, however, only 10.6% (F2018) and 11.2% (Sp2019) of students agreed or strongly agreed with that statement (see Table 7). For F2018 the differences between the initial and final survey responses were significant (F(1) = 19.49; p<0.001). For Sp2019, the difference was marginally significant (F(1) = 6.075; p = 0.06).

Analyses also explored whether or not student characteristics made a difference in their perceptions of the course experience. Chi-square analyses showed no significant differences by gender in student beliefs about their ability to program in Python ($X^2_{(3)}$ = 3.17; p = 0.366), whether the course reduced their fear of programming ($X^2_{(4)}$ = 4.61; p = 0.33), their intent to take more computer science courses ($X^2_{(4)}$ = 7.01; p = 0.14) nor their interest in computer science ($X^2_{(4)}$ = 2.69; p = 0.61). Some differences were noted, however, when comparing students' responses by final grade earned, as perhaps might be expected. These specific analyses (Table 3) should be interpreted with caution, however, given the number of empty or low frequency cells. Chi-square analyses showed that students who earned higher grades were more likely to *agree* or *strongly agree* that they felt able to solve problems in Python, while student who earned lower grades were more likely to be *neutral* or to *disagree.* This difference was significant ($X^2_{(12)}$ = 24.86; p = 0.015). The pattern of responses was the same in students' reported intent to take more computer science courses and whether the course reduced their interest in computer science (X2(16) = 39.66; p<0.001). There was also a significant difference in student responses about whether or not the course reduced their fear of programming, with more 'A' and 'B' students indicating that they did plan to, with more 'D' and 'F' students indicating that they were neutral or did not plan to ($X^2_{(16)}$ = 43.96; p <.001)

TABLE 3. STUDENT PERCEPTIONS OF PROGRAMMING ABILITY BY FINAL GRADE(F2018)

	A N = 24	B N = 13	C N = 11	D N = 7	F N = 8
Having completed this course, I now feel that I can program or solve small problems in Python					
Strongly Disagree	0	0	0	0	0
Disagree	0	0	1	0	1
Neutral	2	3	3	4	5
Agree	6	3	5	1	2
Strongly Agree	16	7	2	2	0
As a result of my experiences in this class, I plan to take more computer science courses.					
Strongly Disagree	1	1	2	1	1
Disagree	0	0	1	2	0
Neutral	0	0	1	3	3
Agree	7	3	5	1	2
Strongly Agree	16	9	2	0	2
Overall, the course reduced my fear of programming.					
Strongly Disagree	3	0	2	0	0
Disagree	1	1	0	0	2
Neutral	6	1	2	7	3
Agree	2	8	5	0	1
Strongly Agree	12	3	2	0	2

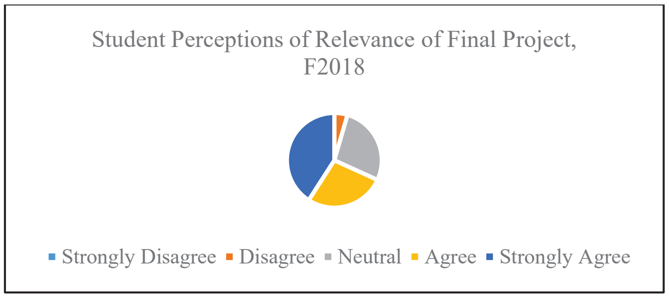


Fig. 9. Student Perceptions of Relevance of Final Project (F2018 responses only)

On the survey administered near the end of the course, students were asked, the final project was relevant to what I have learned in this course throughout the semester. Twenty-six students from the F2018 cohort did not complete the final project or for other reasons scored zero ("0") points on the project. Those 26 students were removed for a more detailed look at the final project grades. Analyses show that the modal response was to strongly agree (40.9%) with this statement (See Fig. 9 & 10). Further, the combination of 'agree' and 'strongly agree' responses totaled 68.2% of responses. Further analyses showed that there was no significant difference in responses by final grade earned (X2(12) = 14.93; p = 0.245).

TABLE 4. DISTRIBUTION OF FINAL PROJECT GRADES BY PERCENTILE, (F2018)

SCORE	N	QUARTILE
36-84	24	25TH
88-97	23	50TH
98-101	25	75TH
102-105	23	100TH

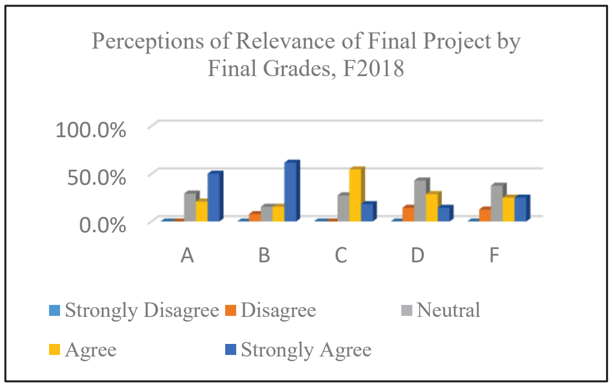


Fig. 10. Final Project by Final Grades (F2018 only)

For the remaining group of 98 students, the scores ranged from 36 to 105 with a mean of X = 90.87, ds = 16.78. The skewed distribution is likely a reflection of the number of students who gained the necessary skills and did quite well on the project. The Table 4 immediately reflects the distribution by quartiles, showing that scores as high at 84 were within the bottom 25% of the distribution, and that one had to score at least 102 to be within the top 25% of the distribution.

V. CONCLUSIONS

We proposed to implement some already successful pedagogy in teaching introductory programming in Python by incorporating both interactive and collaborative learning in the classroom through using a web-based interactive computer programming environment and actively engaging students to stimulate students' interests instead of listening to a traditional lecture. We expect that our ongoing instructional approach would result in a faster acquisition of the needed skills and in producing more confidence in students in programming code development, positioning them well prepared for more advanced courses in the curriculum, and retaining them to continue in the CS program. The initial evaluation and data analysis showed that we are heading in a positive direction by increasing student's learning experience and motivation toward programming.

ACKNOWLEDGMENT

This research was supported by an NSF grant (Award ID 1623335), entitled "Targeted Infusion Project: Infusing Computational Thinking and Visual Learning into an Introductory Computer Science Course to Promote Students' Success and Retention".

REFERENCES

[1] A. Forte and M. Guzdial. Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses. Education, IEEE Transactions on, 48(2):248–253, May 2005.

[2] Bryant, R., Sutner, K., and Stehlik, M., Introductory Computer Science Education at Carnegie Mellon University: A Deans' Perspective. Online: http://reportsarchive.adm.cs.cmu.edu/anon/anon/home/ftp/2010/ CMU-CS-10-140.pdf , 2010.

[3] Scott G., Mark J. and Van G., A practical approach to integrating active and collaborative learning into the introductory computer science curriculum, In Proc. of the seventh annual consortium on Computing in small colleges midwestern conference, Consortium for Computing Sciences in Colleges, 2000, pp. 95–100.

[4] Bonwell, C. and Eison, J. Active Learning: Creating Excitement in the Classroom. ASHE-ERIC Higher Education Report No. 1. Washington, D.C.: 1991.

[5] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. A survey of literature on the teaching of introductory programming. SIGCSE Bull., 39:204–223, December 2007.

[6] Kennedy J. Characteristics of Massive open online courses (MOOCs): A research review, 2009–2012. J Interact Online Learn 2014;13(1):1–16.

[7] Ghadiri K, Qayoum MH, Junn E, Hsu P, Sujitparapitaya S. Revolution of learning caused by MOOC and flipped classes—challenges of San Jose State University, USA. 2013 JUCE J 2013;3:2–15. (in Japanese)

[8] Mori A. The path of university reform—The 1st flipped class, Between April and May 2014 Edition, p 34–35; 2014.

[9] Yamaoka F, Kanda T, Ishiguro H, Hagita N. Interacting with a human or a humanoid robot. IPSJ J 2007;48(11):3577–3587. (in Japanese)

[10] Kanda T, Hirano T, Eaton D, Ishiguro H. Participation of interactive humanoid robots in human society Application to foreign language education. J Robot Soc Jpn 2004;22(5):636–647. (in Japanese)

[11] Kanda T, Imai M, Ono T, Ishiguro H. Numerical analysis of body movements on human-robot interaction. IPSJ J 2003;44(11):2699–2709. (in Japanese)

[12] Lutkebohle I, Peltason J, Schillingmann L, Elbrechter C, Wachsmuth S, Wrede B, Haschke R. A mixedinitiativeapproach to interactive robot tutoring: Towards Service Robots for Everyday Environments. STAR 76, p 483–502; 2012.

[13] Matsui Y, Kanoh M, Kato S, Itoh H. Generating interactive facial expressions of Kansei robots using simple recurrent network. J Robot Soc Jpn 2010;28(3):360–368.

[14] Repl.it, instant IDE to learn, build, collaborate, and host all in one place, accessed Aug 2019, https://repl.it/repls

[15] Runestone project, Democratizing textbooks for 21st century, accessed Aug 2019, http://runestoneinteractive.org/

[16] McConnell, J. Active Learning and its use in computer science. In Proc. of ITiCSE '96, pp. 52-54.

[17] Mitra, S., Lopez-Herrejon, R.E.; Zimmaro, D., Johnson, M., and Schulman, S. An Assessment of the Effectiveness of Interactive Technology in an Introductory Programming Course for Non-Majors, Frontiers in Education, 2005. FIE '05. Proceedings 35th Annual Conference, pp. S3C

[18] Preston, D. Pair programming as a model of collaborative learning: A review of the research. Consortium for Computing Sciences in Colleges, 2005, pp. 39-45

[19] https://repl.it/site/blog/multi, accessed Aug 2019.