The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
ASEE
Paper ID #43386

Moving from Matlab to Python in a First-Year Engineering Programming Course: Comparison of Student Achievement and Assessment of Self-Learning

Dr. Robert Scott Pierce P.E., Western Carolina University

Robert Scott Pierce is an Associate Professor of Engineering and Technology at Western Carolina University. He received his Ph.D. in mechanical engineering from Georgia Tech in 1993. Prior to his teaching career, he spent 14 years in industry designing automated positioning equipment.

Dr. Chaitanya Borra, Western Carolina University

Moving from Matlab to Python in a First-Year Engineering Programming Course – Comparison of Student Achievement and Assessment of Self-Learning

Abstract

Most engineering curricula include an introductory programming course in the first or second year. This course typically assumes no previous programming experience and is intended to help students learn the skills that they will need to solve problems in their upper-level engineering courses. Learning outcomes for this course can include data typing, vector and matrix manipulation, programming structures, function definition, and programming strategy.

The choice of programming language for this course varies depending on the curriculum requirements. Common choices are Python, C/C++, Java, and MATLAB. At Western Carolina Engineering, the introductory programming course has traditionally been taught using MATLAB. MATLAB was selected for its relatively simple syntax, its focus on mathematical programming, and its common use as an engineering language.

In recent years, Python has become one of the most widely used general programming languages. The popularity of Python can be attributed to its relatively simple, readable syntax and the fact that Python and virtually all associated modules are open source. This open-source model has resulted in a huge base of modules in virtually every field of engineering and science.

In response to the popularity of Python, Western Carolina Engineering has changed from using MATLAB to Python in the introductory computing course. The course topics and learning goals for the course were not changed, and course lectures were only changed to reflect the change in programming language.

This paper compares student achievement between classes that took the MATLAB-based version of the course and those who took the Python-based version. Students in the two versions were given very similar exams and final project problems so that their achievement of course goals could be compared.

This work is the first phase of a longer-term project intended to assess the digital literacy of Western Carolina Engineering graduates. Students' programming skills will be assessed as they progress through the four-year engineering curricula. A particular focus of this longer-term work is to determine whether students who learn Python as their first programming language are better prepared to adapt to new languages and programming platforms.

Introduction

Programming is an important professional skill for most engineers. An introductory programming course is part of the first or second-year curricula in most engineering programs. However, it comes with many difficult challenges for both students and faculty [1,2,3,4]. The role played by the instructor in the development of these skills cannot be totally ignored but is found to be minimal [5]. Students usually learn by trial and error using tutorials, homework, textbook examples, peer learning, and web-based demonstrations [6]. Many studies [7,8] have indicated high failure rates in a first programming course. Most of the failures are associated with executing the logic with the proper syntax and the ability to debug. Some educators have

suggested starting out with block building programming tools, where the focus is on problem solving rather than syntax [10,11]. However, this might be a drawback when coding skills are necessary in higher-level engineering courses.

Python as a high-level programming language solves this problem as it has relatively simple syntax and is very algorithm centered. It was developed in the late 1980's and is becoming more and more prevalent in recent times due to its application in web development, engineering, and gaming industries [12-14]. The open-source nature of Python, along with the wide array of data types available, can be useful compared to the other common languages such as the family of languages based on C. C requires knowledge of data structures, operating systems, and specific compilers [4]. This steep learning curve can be intimidating to new programmers. C is considered difficult for students [1,9] due to concepts like pointers and dynamic memory allocation. Python programs are typically shorter and require less programming time than a program written in C. Python's interactive interpreter helps with testing various features as code is written. Python's standard library is also evolving with time, making it applicable to new fields of science and engineering.

The MATLAB programming environment is often used in introductory programming courses as an alternative to lower-level languages like C. The Matlab syntax is similar to C, however it does not require explicit variable type declaration, memory management, or array sizing. MATLAB also has add-in toolboxes for specific purposes and tasks used in engineering-based courses. Once a toolbox is purchased and installed, all of the toolbox functions are immediately available to the student, without explicit references to external libraries. Some of the prominent toolboxes used in many junior and senior engineering courses are Simulink, Simscape, Signal Processing and Image processing.

This paper describes an introductory programming course for engineering students at Western Carolina University (WCU). These were mostly first- or second-year BSE-Mechanical or Electrical Engineering students. There were also a few fourth-year students from the Engineering Technology program who were taking the course as a technical elective.

The paper presents modifications that have been made to the course to increase the overall digital literacy of our students. The most significant of these modifications was to change from MATLAB to Python as the primary programming language [21,22]. This change was made for three main reasons: 1) Python has more of the attributes common to "true" programming languages, such as explicit consideration of data types and references to external libraries, 2) Due to its widespread use and open-source architecture, the quantity and range of external resources for Python is far greater than those for MATLAB, 3) Market demand for engineers who are proficient in Python is far higher than the demand for proficiency in MATLAB [16-20].

This change of programming languages was not met with universal support amongst the engineering faculty at WCU. There are several courses in the engineering curricula that require extensive use of MATLAB. Faculty who teach these classes expressed concern that students who learned Python as their first language would not be able to transfer their skills to the use of MATLAB.

In response to this concern, the course was changed to include instruction in MATLAB near the end of the semester. Students were then allowed to choose either MATLAB or Python for their final project. Results of this choice are presented in the Results section.

There was also a concern that the higher level of complexity of Python could lead to a lower level of student achievement in the course. Python is a relatively easy language to learn, however MATLAB is even easier. Some faculty members felt that students learning Python might progress more slowly and finish the course with fewer programming skills than those who learned using MATLAB.

An objective of this paper is to explore the extent to which the second concern is true. Student work from two different semesters is compared. Instruction during Fall 2022 used MATLAB only; during Fall 2023, most of the course was taught using Python. Assignments used in this paper for comparison of student achievement are identical or almost identical between the two years. Student achievement at specific skills is measured using student self-assessment and assessment by the authors. It is hypothesized that the choice of first programming language will not have a significant, negative effect on the final level of students' programming skills.

Description of the Course

The introductory programming course described in this paper is EE 200, Computer Utilization. The catalog description for the course is as follows:

Computer Utilization: An introduction to computers and computing methods to solve engineering problems.

Course Outcomes: Upon completion of the course, students will be able to accomplish the following:

1. *Master fundamentals of high-level program coding.*
2. *Design and formulate logical flow of an algorithm from a problem statement.*
3. *Implement an algorithm by writing and debugging high level program code.*
4. *Demonstrate the use of software in engineering applications.*

The results presented in this paper come from two sections of the course that were taught during Fall Semester, 2022 and two sections that were taught during Fall 2023. The 2022 sections were taught using MATLAB only, the 2023 sections used Python during the first ten weeks, followed by a week of "MATLAB for Python Users." Table 1 compares the schedule of topics for the two versions of the course.

Both versions of the course were heavily oriented towards a "learn-by-doing" approach. Lectures were brief and were followed by in-class examples in which students tried to work a problem, then immediately discussed the solution. About half of the class time was spent with students working together on in-class or homework problems while the instructor answered individual questions.

Students were strongly encouraged to use online resources for all their programming assignments, including exams. During exams, and for the final project, students were told they could not use "live" references but that all online references (including A.I. tools) were allowed. This approach reflects the current state of programming, which has become less focused on memorizing syntax and more focused on effective problem-solving strategy.

	Course Topic 2022	Course Topic 2023
Week 1	Introduction and History of Computing	Introduction and History of Computing, the PyCharm IDE, Script Files, Structured Programming
Week 2	The Matlab IDE, Script Files, Structured Programming	Data Types, Mathematical, Relational, Logical, and Boolean Operators
Week 3	Mathematical Operators	Using the Python Standard Add-In Modules, Random Number Generation
Week 4	Relational, Logical, and Boolean Operators	Using Python Packages from the PyPI Repository, Mathematical Operations using numpy
Week 5	Random Number Generation, Arrays of Equally-Spaced Numbers	Lists, Vectors, and 1D numpy Arrays. Arrays of Equally-Spaced Numbers, Indexing
Week 6	Introduction to Vectors and Matrices, Indexing	Algorithms: for Loops, Plotting Using MatPlotLIb
Week 7	Input/Output functions	Matricies and Higher-Dimensional numpy Arrays
Week 8	Plotting Graphs	Matrix Math, Systems of Linear Equations
Week 9	Algorithms: if/if-else/if-elseif-else, switches, State Machines	Algorithms: while Loops, User Input Validation
Week 10	Algorithms: for loops, while loops	Algorithms: if/if-else/if-elseif-else, switches, State Machines
Week 11	User input validation	Using Matlab
Week 12	User-Defined Functions	User-Defined Functions
Week 13	Final Project	Final Project
Week 14	Final Project	Final Project
Week 15	Final Project	Final Project

Table 1: Schedule of Topics for the Two Versions of the Class

The courses started with a “History of Computing” lecture that was intended to put the hardware and software used in the course in its historical context. Next, students were introduced to the Integrated Developing Environment (IDE) that they used throughout the course. MATLAB was taught using the MATLAB IDE, Python was taught using the Python Interpreter and the PyCharm IDE. It is worth noting that both versions of the course started immediately with writing script files in a code editor, as opposed to an interactive Console window or an online command interpreter. This approach is efficient, as students immediately begin programming in the environment that they will use throughout the course.

For the Python-based version of the course, it was necessary to spend extra time at the beginning to discuss data types. Like most programming languages, Python requires close attention to data types and will frequently throw an error when the wrong type is used. This is less important in MATLAB, particularly for numeric data types. In MATLAB, all numeric values default to double-precision, floating-point numbers. This allows users to ignore most of the issues related to data types, however it is inefficient. Furthermore, students who only learn MATLAB may not be prepared to learn other programming languages such as Python or C where attention to numeric data typing is required.

Both versions of the course included mathematical, Boolean, logical, and relational operators in the first few weeks of the course. The courses then moved to library-defined functions, in the form of random-number generating functions. In the Python version, this is the point at which students learned to import outside libraries into their code. The process of importing and calling external functions, and methods for using Python standard modules and external packages from the PyPI repository were introduced. In particular, the numpy and MatPlotLib packages were introduced for advanced mathematical and plotting operations.

Vectors and matrices were introduced in the sixth week of the course. Two weeks were spent covering vector and matrix algebra and solution of linear systems of equations. These topics were significantly more challenging for the Python programmers. In MATLAB, the default data structure is an array, the size of which can be changed at will. Python requires programmers to distinguish between Python lists, numpy one-dimensional arrays, and numpy higher-dimensional arrays. Furthermore, the size and shape of a vector or array must be explicitly declared and modified.

Flow control algorithms such as loop structures and decision structures were introduced to the students on an as-needed basis. For loops were introduced at the same time as vectors in order facilitate repeated evaluation of an equation over a range of input values. While loops were introduced in the context of input validation, where a user must remain in a while loop until a valid input is entered. If, elif, …, else structures and switch structures were introduced in the context of state machine programming. A quick summary and comparison of these flow control structures was then included in Week Ten of the course.

In Week Eleven of the Python-based course, students were given a rapid introduction to MATLAB and the MATLAB IDE. This introduction leaned heavily on concepts and skills that the students had already learned using Python. Students were given a handout that compared most of the Python commands that they had already learned with the equivalent MATLAB commands. These handouts included code snippets from in-class problems that the students had already worked in Python. MATLAB code snippets were presented next to the equivalent Python snippets. This translation document was key to helping students transition very rapidly from Python to MATLAB.

At the end of Week Eleven of the Python-based course, students were given a homework assignment to be completed in MATLAB. This assignment consisted of homework problems that the students had already done in Python earlier in the semester. Using the translation

document and the programming skills that they had already learned; most students were able to re-solve the problems using MATLAB. After this assignment, students were given an exam that required them to solve problems using Python and using MATLAB.

Week Twelve of the course was spent introducing user-defined functions. This topic was covered in a pair of short lectures, then used heavily for the final project. Students in both versions of the course were given the same final project, which involved solving the equations of two-dimensional particle kinematics through multiple stages of motion. Students in the Python-based version of the course were allowed to use either Python or MATLAB for their final projects. The final project is discussed in more detail in the next section.

Data Collection

An objective of this paper is to examine whether the increased complexity of Python results in a lower level of overall student achievement in the course. Python has some distinct advantages over MATLAB in terms of user base, similarity to other programming languages, and cost, however these advantages could be outweighed if students have a reduced ability to solve engineering programming problems when they finish the course. To investigate this possibility, student achievement levels for individual programming skills are compared between the Fall 2022, MATLAB-only classes and the Fall 2023, Python plus MATLAB classes. Student achievement is quantified using student work from three sources: 1) Homework problems that were self-graded by the students, then verified by the authors, 2) Exam problems from the two years, and 3) Evaluation by the authors of selected sections of the final project. As previously mentioned, the assignments were identical or nearly identical between the two years.

Table 2 presents scores for skills that were assessed using assigned homework problems. Homework assignments in the course were focused on the use a newly acquired skill in the solution of an engineering problem. Figure 1 shows an example of one such problem. Students must solve an algebraic equation for the shear stress in a driveshaft for a range of shaft diameters. They must then construct a plot of the factor of safety against static yielding vs. shaft diameter, then use this plot to select a minimum safe shaft diameter.

It is worth noting that the homework assignments were originally graded by the students themselves. The course used a "self-grading" approach for homework in which the student was required to compare their work with the instructor solutions, evaluate the correctness of their work, and assign grades to their work based on a rubric provided by the instructor. Thus, the "Student Self-Assessment" columns of Table 2 are the students' own assessment of their achievement, normalized to a percentage. This self-grading approach has been well-received by students; it is frequently cited as one of the best aspects of the class in student evaluations of the course.

To ensure that the student self-grading was an accurate reflection of their actual achievement, the authors re-assessed a large sample of the homework problems. The results of this re-assessment are included in Table 2.

Student Self-Assissment 2022 n=36	Student Self-Assessment 2023 n=40	Faculty Assessment 2022 Work n=18	Faculty Assessment 2023 Work n=20	P values between student self-assessment 2022 and 2023	Was the Difference Between 2022 and 2023 statistically significant with a 95% confidence level?	P values between student grading and author grading	Was the difference between student grading and author grading statistically significant to a 95% confidence level?
Skill: Random number generation – Generate real and integer-valued random numbers drawn from a specified range.							
92.860	82.487	89.416	82.487	0.13	No	0.41	No
Skill: Equally-spaced numbers – Generate 1D arrays of equally-spaced real or integer-valued numbers within a specified range.							
90.977	91.944	91.796	91.944	0.87	No	0.86	No
Skills: Solve an algebraic equation. – Solve an algebraic equation for a range of input values, then plot the results. Use the plot to make a design decision (e.g. the diameter of a drive shaft).							
96.759	82.803	95.526	82.803	0.01	Yes	0.78	No
Skills: Evaluate trigonometric functions – Generate values for different trigonometric functions through a specified range of angles. Construct a properly-formatted plot of multiple trigonometric functions through a range of angles.							
85.263	87.058	82.57	87.058	0.80	No	0.68	No
Skills: Matrix algebra – Declare and populate matrices, perform algebraic manipulation of matrices.							
91.176	87.39	88.940	87.399	0.43	No	0.64	No
Skills: Implement a loop structure – Use a for or while loop to solve the one-dimensional kinematic equations of motion through a specified range of time or distance.							
65.387	83.30	65.387	83.30	0.06	No	1.0	No

Table 2: Skills Assessed via Student Self-Grading of Homework Problems

Table 3 presents assessment by the authors of skills that were evaluated as part of an in-class exam. The problems were short programming assignments that could be completed in about 20 minutes. Scores have been normalized to a percentage.

Table 4 gives assessment by the authors of skills that were evaluated as part of the final project. For each row in the table, the authors examined and ran the piece of student code that implemented the skills, then assigned a score of zero to three (where three represents the highest level of achievement) to each student's work.

The final project constitutes a large portion of the course grade and is intended as a cumulative test of the students' programming skills. Students were asked to write a computer game that

The picture below shows the drive shaft of a lawnmower. To a good approximation, the shaft is loaded in pure torsion. This implies that the main stress in the shaft is created by the torque that is being transferred from the motor to the blade. The stress that is created is called, "shear stress due to torsion" and is denoted using the Greek lowercase letter τ (tau).

The equation for the maximum value of shear stress due to torsion is: $\tau_{max} = \frac{Tc}{J}$

where:

T = the torque carried through the shaft (units = N-m)
c = the smallest value of the outer radius of the drive shaft (units = m)
J = the second polar moment of area of the shaft cross-section (units = m^4)

For the lawnmower shaft in torsion, we can calculate the "Factor of Safety Against Yielding in Shear":

$$F.S. = \frac{Yield\ Strength\ in\ Shear\ of\ the\ Shaft\ Material}{Maximum\ shear\ stress\ generated\ in\ the\ shaft} = \frac{s_{ys}}{\tau_{max}}$$

Maximum shear stress due to torsion will occur on the outside surface of the part of the shaft with the smallest radius

Next, run your code with the following input values:

Torque = 120 N-m
sys = 190 Mpa
dia_min = 5.0 mm
dia_max = 30.0 mm

Using these input values and your plot, what is the minimum shaft diameter that will result in a factor of safety of 1.50 against static yielding?

Problem 3 Grading Rubric:

Your code executes correctly and generates correct answers. Your plot has all of the required formatting (title, axis labels, grid, specified units) and has correct values. You were able to use your plot to find the correct answer to the question in Part 2. - 20 points

Your code executes correctly and generates correct answers, however your plot is not completely formatted. You answered the question in Part 2 correctly. - 19 Points

You wrote code that executes, however it does not give the correct answers. You created a plot. You answered the question in Part 2, however you did not get the correct answer. - 16 points

You wrote code, however you were not able to get it to execute. Therefore, you did not have a plot or an answer to the question in Part 2. - 12 points

You did not submit any substantial work on this problem. - 0 points

Problem 3 total = _____ / 20 possible

Figure 1: A sample homework assignment.

Faculty Assessment of Exam Submissions, Fall 2022	Faculty Assessment of Exam Submissions, Fall 2023	P-value between the Fall 2022 and Fall 2023 Exams	Is the difference between 2022 and 2023 Exam Performance Significant to a 95% Confidence Level?
Skills – Solve multiple algebraic equations for a range of values, then plot the equations together on a properly-formatted graph. 2022 – Exam 1 second script for both years			
79.2, n=53	87.7, n=72	.01	No
Skill: Matrix manipulation – Generate matrices, then add and remove rows or columns and manipulate individual elements using indexing. 2022 Exam 1 p1 2023 exam 2 p1			
92.4, n=60	92.1, n=63	0.90	No
Skill: Implement a while loop – Recognize the need for a while loop and correctly implement the loop structure in a counting problem. Exam 2 second script both years			
66.8, n=58	86.7, n=63	0.00	Yes

Table 3: Skills Assessed Using In-Class Exams

Faculty Assessment of Fall 2022 Student Work	Faculty Assessment of Fall 2023 Student Work	p-value between the two years	Is the difference statistically significant to a 95% confidence level?
Skill: Input validation – Prompt user for inputs, then ensure that user inputs are valid and will not result in an error. Prompt the user when invalid inputs are entered.			
1.71, n=42	1.57, n=53	0.56	No
Skill: Correctly implement an if, elif, …, else structure – Evaluate decision variables , then use the results to implement the correct action using an if, elif, …, else structure.			
2.36, n=42	2.40, n=48	0.89	No
Skills: Data matrix manipulation – Solve the two-dimensional equations of particle motion over a multi-segment path. Store position and speed coordinates throughout the path. Manipulate data using matrix indexing.			
1.89, n=53	1.52, n=61	0.05	Yes

Table 4: Skills Assessed Using the Final Project

simulates jumping a moon buggy over a crater on the Moon. Figure 2 describes the steps that the game should follow.

The project required the students to write several user-defined functions, and to call functions defined in a file provided by the instructor. The project utilizes most of programming skills that were taught during the semester. These include input verification, selection from alternatives, iteratively constructing a data matrix, loop structures, manipulating elements of a data matrix using indexing, and plotting.

The final projects that were used in both versions of the course were identical. Students in the 2023 class were allowed to use either Python or MATLAB for the final project. Of the 55 students in the 2023 classes who submitted the final project, 11 used MATLAB.

Results

Amongst the programming skills that were assessed using student self-graded homework (Table 2), most of the skills do not show a significant difference between the primarily Python-based class (Fall 2023) and the MATLAB-only class (Fall 2022). Students in both versions performed at roughly the same level. The Fall 2022 class did significantly better on the problem that required solving algebraic equations.

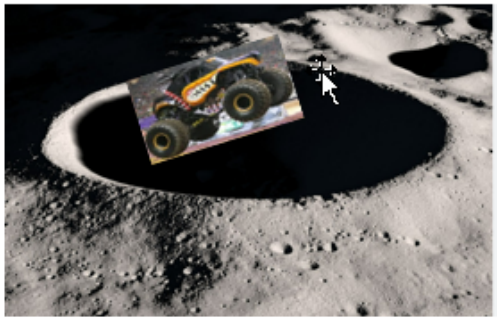
The scores from the student self-graded homework problems correlate well with the scores that were assigned by the authors when they re-graded these problems. This re-grading was done as a check on the correctness of the grades that students assigned to themselves. In all cases, there was no significant difference between the student self-grading and the author re-grading.

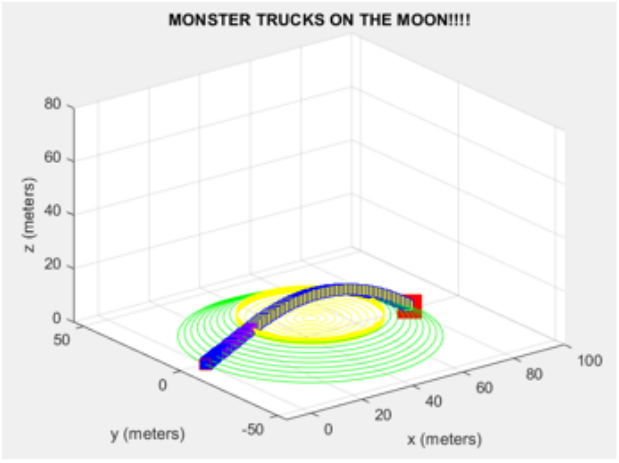
Table 3 shows relatively small differences between the two classes on in-class exam problems. The only significant difference was in implementing a while loop. The Fall 2023 class performed significantly better than the Fall 2022 class at this skill.

Table 4 is noteworthy, since the final project is by far the most challenging assignment of the course. Differences between scores on the input validation and decision structure skills were not significant. There was, however, a significant difference in student performance on the data matrix manipulation task. The data matrix manipulation task in the final project is by far the most challenging task assigned during the course. There are several calculations that must be performed and there is extensive manipulation of matrix elements using indexing. The Fall 2022 (MATLAB only) class performed better at this complex programming task than the Fall 2023 (primarily Python) class.

To investigate this disparity more thoroughly, scores from the Fall 2022 class were compared with the scores of only those students who used Python for their final projects in Fall 2023 (students who used MATLAB for the final project in 2023 were removed from the data set). Table 5 shows this comparison. With this modification to the 2023 data, the difference between the two years becomes more pronounced. The students who took the MATLAB-based class were significantly better than the Python users at performing complex, multi-step matrix manipulation tasks.

MONSTER TRUCKS ON THE MOON VIDEO GAME SPECIFICATIONS:





For this project, you will write a program for a video game that lets a single player simulate jumping a moon buggy over a crater on the moon.

Here's how the game will work:

1) When the main script is run, the player sees a welcome message.

2) The player is asked to hit Enter to start. The program waits for this to happen.

3) Once the player hits enter, the program generates a crater. The crater height, slope, top diameter, and depth are generated randomly (between specified bounds).

4) The program plots a 3-D plot of the crater and shows the starting point of the jump. It also reports the crater height, top diameter, base diameter, slope, and depth.

5) The program asks the user choose between 3 different moon buggies, each of which has different mass, output torque, and wheel diameter.

6) The program asks the percent of full throttle that the player would like to use during the jump.

7) The programs reports the user's selected buggy parameters and percent full throttle, then asks the player to hit enter to start the jump.

8) Once the player hits Enter, the program does the following:

a) Calculates whether the buggy rolls backwards, stays stuck at the starting point, or accelerates forwards, up the slope. If the buggy rolls backwards or is stuck, the program prints an entertaining message, **and the game is over.**

b) If the buggy accelerates up the front flank of the crater, the program finds the path up the flank at equally-spaced times and stores the path in a data matrix.

c) The program then finds the parabolic portion of the path at equally-spaced times and appends the parabolic part of the path to the data matrix.

d) The program then finds the landing point, which could be in the bottom of the crater, against the crater wall, on the back flank of the crater, or past the back edge of the crater. NOTE: LANDING ON THE BACK FLANK OF THE CRATER IS THE DESIRED OUTCOME OF THE GAME AND IS THE ONLY OUTCOME THAT DOESN'T RESULT IN A CRASH OF THE BUGGY.

Figure 2: The final project assignment

Faculty Assessment of Fall 2022 Student Work	Faculty Assessment of Fall 2023 Student Work- With Matlab Users Removed	p-value between the two years	Is the difference statistically significant to a 90% confidence level?
Skills: Data matrix manipulation – Solve the two-dimensional equations of particle motion over a multi-segment path. Store position and speed coordinates throughout the path. Manipulate data using matrix indexing.			
1.89, n=53	1.30, n=43	0.00	Yes

Table 5: Assessment of the Data Matrix Manipulation Skills in the Final Project With the Matlab-Users Removed from the Fall 2023 Data

Discussion of Results

For the most part, the achievement levels between the Fall 2022, MATLAB-only class and the Fall 2023, mostly-Python class were quite similar. The self-graded homework seemed to show that the Fall 2022 class was more skilled at solving algebraic equations, but the exam-based evaluation showed the Fall 2023 cohort to be better at identifying the need for a while loop and correctly implementing the loop structure.

The most concerning result is the significantly better performance of the MATLAB-only class at complex, multi-step matrix manipulation tasks. This difference is even more significant when the students who used MATLAB for the final project are removed from the 2023 data. It appears that the increased complexity of matrix manipulation in Python does affect the ability of the Python programmers to carry out these complicated matrix manipulation tasks. Since MATLAB was developed specifically for this sort of task, it is not surprising that the MATLAB programmers exhibited a higher level of achievement. Engineering calculations are packed with matrix manipulation tasks; therefore this issue needs to be addressed in future versions of the class.

Conclusions and Future Work

This paper has presented comparisons between the achievement of MATLAB-based, and Python-based versions of an introductory engineering programming course at WCU. In most cases, the level of achievement between the two versions was comparable. In the case of complex, multi-step matrix manipulation tasks, the MATLAB programmers demonstrated a significantly higher level of achievement. This shortcoming needs to be addressed in future versions of the class. It is possible that earlier introduction of matrix manipulation functions, combined with an extra assignment on the topic could help. Of course, adding new course content on matrix manipulation will require the removal of some part of the current content.

It is worth noting that, despite the more challenging matrix manipulation functions, there are still reasons to use Python in the introductory programming course. The increased use of data types and required references to external libraries make Python a closer match to lower-level languages that students may need to learn in the future. The much larger base of users, large number of external packages available, and open-source paradigm make Python a choice that must be considered.

The work described in this paper is intended to be the first results from a longer-term study of the effects of moving to Python as the programming language of choice at WCU. In the future, the achievement of the MATLAB programmers and Python programmers will be monitored as they apply their skills in higher-level engineering courses. A particular focus will be placed on monitoring the ability of students who learn Python as their introductory programming language to move to MATLAB in courses that require it. Use of Python programming in higher-level engineering classes and project-based courses such as the Senior Capstone class will also be monitored. This work is intended to guide the faculty in ongoing efforts to raise the level of digital literacy of our graduates.

References

1. Lahtinen, Essi, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. "A study of the difficulties of novice programmers." Acm sigcse bulletin 37, no. 3 (2005): 14-18.
2. Piteira, Martinha, and Carlos Costa. "Learning computer programming: study of difficulties in learning programming." In Proceedings of the 2013 International Conference on Information Systems and Design of Communication, pp. 75-80. 2013.
3. Sobral, Sónia Rolland. "The first programming language and freshman year in computer science: characterization and tips for better decision making." In Trends and Innovations in Information Systems and Technologies: Volume 3 8, pp. 162-174. Springer International Publishing, 2020.
4. Balreira, D.G., Silveira, T.L.D. and Wickboldt, J.A., 2023. Investigating the impact of adopting Python and C languages for introductory engineering programming courses. Computer Applications in Engineering Education, 31(1), pp.47-62.
5. Zingaro, Daniel. "Examining interest and grades in Computer Science 1: a study of pedagogy and achievement goals." ACM Transactions on Computing Education (TOCE) 15, no. 3 (2015): 1-18.
6. Porter, Leo, Mark Guzdial, Charlie McDowell, and Beth Simon. "Success in introductory programming: What works?" Communications of the ACM 56, no. 8 (2013): 34-36.
7. Bennedsen, Jens, and Michael E. Caspersen. "Failure rates in introductory programming." AcM SIGcSE Bulletin 39, no. 2 (2007): 32-36.
8. C. Watson and F. W. Li, Failure rates in introductory programming revisited, Proc. 2014 Conf. Innov. Technol. Comput. Sci. Educ., 2014, pp. 39–44.
9. Malik, Sohail Iqbal, Roy Mathew, Abir Al-Sideiri, Jasiya Jabbar, Rim Al-Nuaimi, and Ragad M. Tawafak. "Enhancing problem-solving skills of novice programmers in an introductory programming course." Computer Applications in Engineering Education 30, no. 1 (2022): 174-194.

10. Resnick, Mitchel, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner et al. "Scratch: programming for all." Communications of the ACM 52, no. 11 (2009): 60-67.
11. Pea, Roy D. "Logo programming and problem solving." (1987).
12. McGugan, Will. Beginning game development with Python and Pygame: from novice to professional. Apress, 2007.
13. Soikkeli, Eero. "Scaling out Big Data Distributed Pricing in Gaming Industry." (2019).
14. Da Silva, Josivan Pereira, Paulo Henrique Gonçalves Pimentel, Luciano Gonçalves Pimentel, and Ismar Frango Silveira. "Pixel Python RPG: Repurposing an Entertainment Game to an Open Educational Resource for Computer Programming Fundamentals." In 2021 XVI Latin American Conference on Learning Technologies (LACLO), pp. 326-333. IEEE, 2021.
15. Gutschnidt, Tom. Game Programming with Python, Lua, and Ruby. Premier Press, 2003.
16. Gurcan, Fatih, and Cemal Kose. "Analysis of software engineering industry needs and trends: Implications for education." International Journal of Engineering Education 33, no. 4 (2017): 1361-1368.
17. Gao, Xinkai. "Python based IT industry recruitment data automatic collection, warehousing, and analysis system." In 5th International Conference on Computer Information Science and Application Technology (CISAT 2022), vol. 12451, pp. 572-577. SPIE, 2022.
18. Kurennov, Dmitry V., Natalia G. Ryzhkova, Yury V. Serdyuk, Maya L. Mayants, and Elena A. Timokhova. "Formation of IT Competences of Future Mechanical Engineers." In ITM Web of Conferences, vol. 35, p. 01008. EDP Sciences, 2020.
19. Scardua, Leonardo Azevedo. Applied Evolutionary Algorithms for Engineers Using Python. CRC Press, 2021.
20. Ozgur, Ceyhun, Taylor Colliau, Grace Rogers, and Zachariah Hughes. "MATLAB vs. Python vs. R." Journal of data Science 15, no. 3 (2017): 355-371.
21. Guedes, Priscila FS, and Erivelton G. Nepomuceno. "Some remarks on the performance of MATLAB, Python and Octave in simulating dynamical systems." arXiv preprint arXiv:1910.06117 (2019).
22. Colliau, Taylor, Grace Rogers, Zachariah Hughes, and Ceyhun Ozgur. "MATLAB vs. Python vs. R." Journal of Data Science 15, no. 3 (2017).