Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #45694

Python or Java in a Data Structures course? How about both?

Dr. Udayan Das, Saint Mary's College of California

Udayan Das is a computer science professor with over a decade of experience teaching computer science.

Python or Java in a Data Structures Course? How About Both?

Udayan Das
Mathematics and Computer Science
Saint Mary's College of California
Moraga, CA, US
udd1@stmarys-ca.edu

***Abstract*— This experience report relates success using both Python and Java in a data structures course. The course has been taught with Python in the most recent past as a follow-up to an introductory programming course that is taught in Python. However, due to student demand Java was included in Spring 2024. Students were surveyed at the end of the course on quantitative measures and also asked to share their perspective. Students found that Java helped them think more clearly about objects, while the use of 2 programming languages in parallel helped them to focus on higher level abstract concepts without getting bogged down in language details. My observation as instructor reflects this as demonstrated by student performance on a linked list implementation assignment as well as an unbalanced binary search tree implementation assignment. Both assignments showed improved performance; and quicker submission times with more than half students turning in the unbalanced binary search tree assignment before the deadline. Finally, more students chose an implementation project as their final project, between the choice of completing an implementation project or a project studying a data structure not covered in the course, than have typically done in the past. On this last point, I will share experiences with the breakdown of students opting for one or the other from many semesters teaching data structures across several different institutions. While this data is not statistically significant (n=35) it could be indicative of the benefit of using more than 1 language in a data structures course, particularly one that is relatively simple (Python) and one that is more strictly typed and object oriented (Java). This experience report will discuss the structure of the course in brief, including a discussion of where and when Python or Java was used. I will discuss what went well and what I would change in the future, and include a suggested timeline of topics along with my suggested programming language to use for those topics. The student survey results are also presented in detail.**

Keywords*— *CS2, Data Structures, Python, Java, Programming Languages, CS Education

I. Introduction

At Saint Mary's College (SMC) of California, the data structures course (CS 222) is titled Programming II: Data Structures and Algorithms. CS 222 spends 2 weeks reviewing intro programming topics because we sometimes have students come to the second programming course after a gap or there are transfer students. During the first week of CS 222 this Spring, some students expressed the desire to be exposed to a different programming language than the one that they had encountered before, viz. Python being the language used in our intro programming class CS 121: Programming I. This seemed like a great idea, given that I have taught data structures with Java a number of times and there are some object oriented concepts that are well taught with Java. Further, I felt this would be a great opportunity to not only have students learn Java but also teach the data structures parallelly in both Python and Java. Staying with the fact that the genesis of the idea to introduce something different was student driven I opted to democratize the choice of programming language options of the course (within reason). All of the students in the class were therefore polled to select one of 3 options for the course: 1) taught using Python alone; 2) taught using Java alone; 3) taught using both. A majority of students selected the third option. The course was modified to fit this approach.

This being the first time I had taught using both Python and Java, I also decided that this was a perfect opportunity for a CS education study and this paper relates both the experience and the results of using this approach. Once again, the fact that this was still during the "review intro programming" phase of the course was beneficial since I was able to begin modifying the course accordingly. (Luckily, I do not mention programming language specifics in my Syllabus so no changes were required there.) The other advantage was that I have over a decade of experience teaching data structures in several languages including C and C++ in addition to Java and Python.

What resulted was an intellectually stimulating exercise: a course I had taught many times, including 3 times in the recent past using Python, and about a half dozen times immediately preceding that in Java, and had become somewhat second nature now became somewhat of an active pedagogical/andragogical challenge. In hindsight, I am thankful to the initial handful of students who brought up the idea. There is something to be said about being open to feedback as an instructor and students being ready to provide feedback and ideas on their learning journey. The process proved more rewarding than I had originally expected because both student engagement and student performance improved as a result of the change. This further solidified the need for reporting on the results of this change to the data structures course. The experience report and results are presented in this paper.

II. Background and Related Work

A comprehensive review of the literature on the teaching of data structures and algorithms is to be found in Silva et al. [2], focused mainly on the second half of the 2010s. They looked at specific questions around pedagogical/andragogical techniques, course content coverage, and teaching support tools used. They also specifically looked at the question of programming languages used and found that Java was the most common language used in 40% of the studies cited [2], followed by Python, C++, and C respectively. Surprisingly they also found the use of Matlab and Ruby in some cases. Key pedagogical/andragogical techniques identified were problem- and project-based learning, peer instruction and pair programming, and gamification mechanisms. For a phenomenography of pedagogical/andragogical approaches in data structures courses, see Lister et al. [3], with key elements identified as developing transferable thinking, the need to improve student's programming skills, teaching them what's "under the hood," developing their knowledge of software modules and libraries, and developing component or modular thinking. I found that the use of more than one programming language also enabled students to more easily see the bigger picture, along with the supporting the aforementioned learning goals.

Necaise [4] reports on the opposite situation to that at SMC, focusing on a transition from Java to Python in the CS2 course and reported student benefits in terms of being less bogged down with *{syntax}* and being able to focus more on

problem solving. This runs somewhat counter to our experience in the CS2 course. To be clear, we have found Python an ideal language to use in the CS1 intro programming course with precisely the benefit identified by Necaise with students being able to focus on developing computational thinking and computational problem solving. However, in the CS2 course, the syntactical elements, particular those making references and other structural elements such as constructors more explicit, has strongly benefited students—both as observed by the instructor as well as reported by students in their post-course survey.

Maurer [5] presents an interesting comparative programming language course and their experiences therein. Given the experience reported in this paper with using both Python and Java concurrently, I believe there is great untapped potential value in including a formal comparative programming language course in undergraduate CS curriculum. (Although I will note that comparative approaches to programming languages do appear explicitly or implicitly in some undergraduate CS programming language courses; however, they tend to be more elective than required.) A course with 2 languages such as reported in this paper, is a step towards comparative programming language education in a more formal way than emerges when students end up using different languages in different CS courses.

Enbody et al. ([6], [7]) have reported on how effective Python in CS1 has been towards preparing students who take a C++ based CS2 data structures at an public R1 university. They found no difference between students having taken Python or not-Python as far as their performance in the CS2 course. This suggests that instructors should feel comfortable swapping out Python for something else in the CS2 even if students began their studies with Python. Note: here, I was specifically thinking about the counter-result that is that transitioning from Python to C++ did not result in worse performance for students as compared to having already studied C++ in CS1. I think this is a key finding of note for every CS2 instructor, inasmuch as switching to more complex syntax is not necessarily a big leap for students. (Of course, bearing in mind Enbody et al.'s public R1 context.) This finding is also supported by Gal-Ezer et al. [8] who found that different languages as well as different paradigms (objects-first, objects in the CS1 course, no objects in the CS1 course) had no appreciable effect on CS2 performance. As to this paper, my finding is that it is significantly beneficial to have more than one programming language in the CS2 data structures course.

Using more than one programming language has been more commonly tried in CS1 courses ([9], [10]) indicating definite benefits such as enabling students to focus better on computational thinking and computational problem solving then getting too caught up in syntax. Two programming languages have specifically been used by Ham et al. and Rybarczyk et al. respectively in [11], [12]. Ham et al. [11] report on the use of 2 different programming languages for different projects in a biomedical engineering course finding that there was no significant benefit to adding the second programming language. Worth noting however is that there was no comparative element to the approach and the authors appear to have been mainly interested in exposing students to more than one language, particularly given the limited scope of exposure for typical biomedical engineering students. They also found no significant negative impact of having more than one language. The combination of those factors suggests that if more than language is being used the educational framing for that choice should be clear and there should be pedagogical/andragogical approaches that support reflection of the differences and thereby gains may be achieved as has been my experience for this paper. The comparative aspect and the why/when/how is specifically emphasized by Rybarczyk et al. [12] and they identify this as being key to career preparedness for CS and CS-adjacent students and report improvements in student satisfaction, understanding, and learning. Having used the comparative approach throughout a semester long course I can attest that the results strongly support this finding. The value of reflective learning in the CS2 data structures is also highlighted by Resch et al. [13].

To close out this section I remind the community of Aharoni's [14] call to consider the cognitive processes of CS learners in more depth particularly in a course like data structures where there is a particular mix of the abstract and the concrete (see also: [3]).

Sort algorithms: $O(n^2)$ – Selection Sort, Bubble Sort
Sort algorithms: O(n logn) – MergeSort, QuickSort, HeapSort
Sort algorithms: O(dn) – Radix Sort
Fundamental Data Structures / ADTs: Lists – linked and array implementation
Fundamental Data Structures / ADTs: Stacks and Queues and their implementation
Search Algorithms: linear search and binary search
Hashing
Binary Heaps and Priority Queues
Unbalanced Binary Search Tree and implementation
Balanced Binary Search Trees
Graphs and their representation / BFS/DFS
Minimum Spanning Tree algorithms: Kruskal's and Prim's
Shortest Path algorithms: Dijkstra's and Floyd-Warshall
Big-O analysis: Worst-case and average case analysis

Fig. 1. Concept and Topic Coverage in CS 222 Programming II: Data Structures and Algorithms

III. CS 222: Programming II: Data Structures and Algorithms

CS 222 is a fairly typical CS2 data structures course and mostly aligns with the ACM curriculum guidelines that are currently under development (see ACM/IEEE Computer Society/AAAI CS curriculum guide 2023 beta version 2 [1]) covering most of the topics that are suggested in the curriculum guide for a CS2 data structures course. Figure 1 shows the full list of topics covered in the course. The topic coverage timeline is as follows:

- Weeks 1-2: Review
- **Week 3: Objects and Classes / Selection Sort**
- Week 4: Binary search / Big-O analysis intro
- **Week 5: Lists**
- **Week 6: Stacks / Bubble Sort**
- **Week 7: Queues**
- Week 8: Sort algorithms: Merge and Quick
- **Week 9: Unbalanced Binary Search Trees**
- **Week 10: Heaps and Heapsort**
- Week 11: Hashing

- **Week 12: Graphs / MST**
- **Week 13: Shortest Path Algorithms**
- Week 14: Balanced Binary Search Trees

In the weekly schedule above, **bolded** weeks show ones involving code demonstrations or in-class student coding exercises. In Spring 2024, in each of these weeks (except week 12 and 13) both Python and Java were used to develop code snippets, to run code demos, and to have students work on student coding exercises. Most of the student coding exercises are done in pairs or in groups of 3 depending on the complexity of the task. By week 12 and 13 when there were coding exercises, students were given the option of using a programming language of their choice.

The course has the following programming assignments:

1. Doubly Linked List Implementation: Students are asked to complete a baseline singly linked list implementation begun in class and then modify that to complete a doubly linked list (DLL) implementation. (Given that both Python and Java versions were developed in class, students had an option to choose which language to implement the DLL in.)
2. Comparison of Sort Algorithms: Source code for Selection Sort (developed in class), MergeSort, and QuickSort (partially developed in class, particularly to highlight list issues in Python[1]) are provided in both Python. Students compare the performance of sorts on instances of size 100-100,000,000 that are already sorted in ascending order, descending order, and random order. Students are required to generate 9 test cases (3 of each, ascending, descending, and random) for each size quantum (100, 1000, 10K, 100K, 1M, 10M, 100M). (Assignment used Python.) Extra credit provided for including a fourth sort algorithm (self-implemented) in the comparison. Extra credit for using both Python and Java (code provided) and comparing the results.[2]
3. Binary heap implementation: array- or list- based implementation. (Java)
4. Unbalanced Binary Search Tree implementation: student choice of programming language. Extra credit for a graphical representation of the BST.

In addition to these there is a final project in the class in which students are asked to either report on a data structure or algorithm not presented in class (ex: skip lists; A* search), or to implement a data structure or algorithm not developed in class or in an assignment (ex: Balanced BST, Union-Find disjoint set based Kruskal's algorithm).

IV. Experience Report

In this section, I will reflect on what went well in the course and include some observations on improvements that I saw. I will also discuss briefly what I might have done differently.

A. Experience and observations

As I indicated earlier, in many ways the teaching of the course was made more interesting than it has been in several years. I always teaching this course, and it is definitely one of my favorite courses to teach to undergraduates. However, this time I was challenged with developing more materials than normal as well as to think constantly about highlighting differences between Python and Java, in turn I think this resulted in me highlighting more elements of code development

[1] Using List operations such as *remove* and *append* results in much worse performance than swapping elements.

[2] Students had a significant mind-blown moment to see 10x speed improvement in Java.

than is typical for me in this type of course. In addition, it enabled me to comment on language specifics. Ex: natural use of arrays in Java vs lists in Python; spending more time discussing dynamic arrays.

Due to the constant challenge of shifting between languages I noticed that students generally appeared to be more engaged with in class exercises. This was most readily exhibited in two ways: 1) most students starting work right away instead of having to be poked and prodded in any way, and 2) on an empirical level, getting through more in-class exercises in the course of the semester (even if at times, it was because students were asked to repeat the same task in the other language).

A unique unexpected opportunity during in-class exercises was asking students to pair up and have each one do the task in one of the languages and then to compare their approaches and results. The value of this exercise seems to be considerable and I am debating whether to present an experience report on this topic alone in the future.

B. Linked List Assignment

Across institutions and departments, this one has historically been the most challenging assignment for students. I believe it is because working with pointers or references is new for students and some students have a hard time making the adjustment. Typically, students utilize the full assignment time available to complete, including up to about 10% late submissions (not more than a day or two unless there are exceptional circumstances). This was the first instance of this assignment where more than 50% of students had already turned in the assignment before the deadline, with all but 1 student turning it in by the deadline. This is not something I was expecting with the change. But I surmise something just *clicked* with seeing the implementation of the singly linked list in both Python and Java during class. Noting that part of the in class implementation is student's completing specific methods such as *delete()*.

C. Unbalanced Binary Search Tree

Students usually find this assignment much easier after having completed the linked list. (Yes, it always surprises me.) Half the students turned it in within 4 days. All but one turned in this assignment ahead of time, almost like they were excited to complete the assignment.

D. Final project, implementation option

Historically, across institutions, the implementation option is selected by 15-20% of students. Typically, 5-6 students out of 30. This time 16 students out of 35 opted for and completed an implementation. I believe this is indicative of greater confidence in being able to accomplish the final project.

E. Other notes

This is also a course in which I used GPT tools at different points, the results of which I present elsewhere. However, one of the amusing elements of this course was spending a fair amount of time including in-class thought exercises developing pivot strategies for a dependable O(n log n) QuickSort, only to then ask ChatGPT and Perplexity to generate QuickSort code which was exactly identical, to variable names, that was clearly the $O(n^2)$ worst-case.

The grade distribution skewed higher but I believe this is at least in part due to the rate of completion and the more in-time submissions. Both of these factors are of course positive for overall student performance.

V. Survey Results

Students were surveyed at the end of the course regarding the use of Python and Java in the teaching of the course. This section presents results from the survey (n=35).

A. Language preferences

1) Python vs Java

After encountering and working in both languages which language do students prefer working in. Most students preferred Python (figure 2) but a fair number (31.4%) indicated they like both Python and Java.

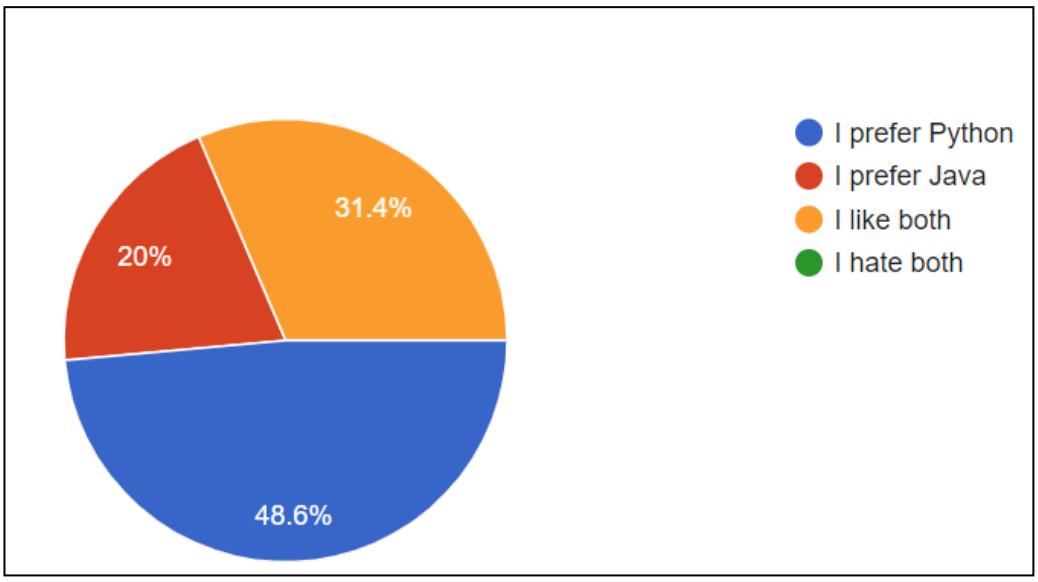

Fig. 2. Student responses on Python vs Java.

2) What will they use in the future

54.3% of students said that they would choose a language depending on the task. This is a huge victory for an instructor (figure 3).

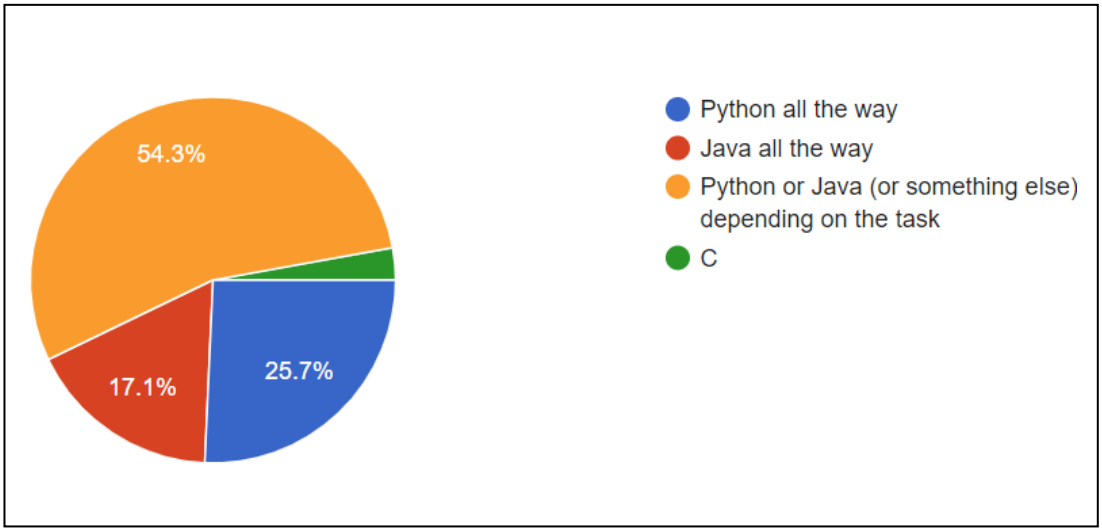

Fig. 3. Student responses on what they will use in the future. Note the overwhelming number for *depends*. (One student is old school.)

B. Learning data structures and objects

1) Did Java help think more clearly about objects?

80% of students felt that Java helped them think more clearly about objects (figure 4).

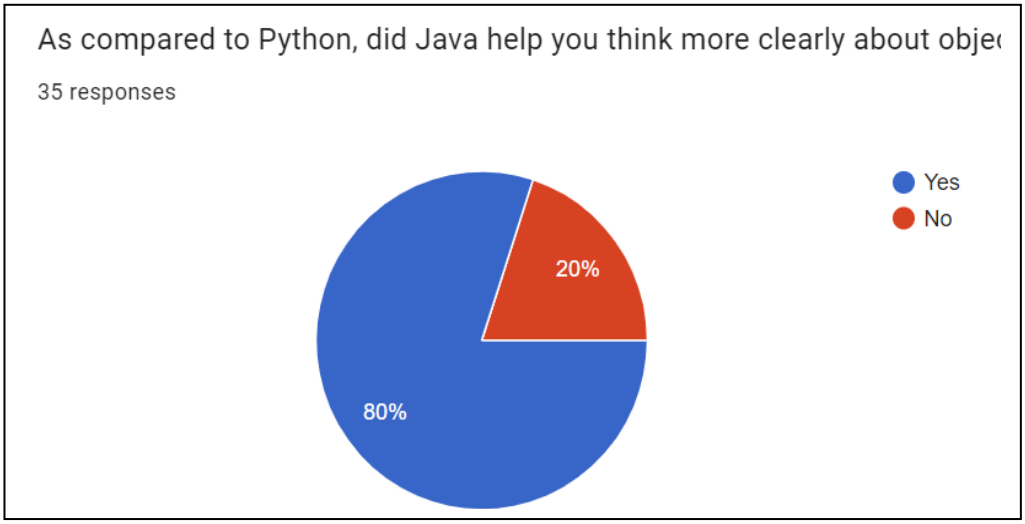

Fig. 4. Student response on whether Java helped them think more clearly about objects.

2) Thinking about data structures

72.7% thought that Java helped them think more clearly about references and the abstract structural aspects.

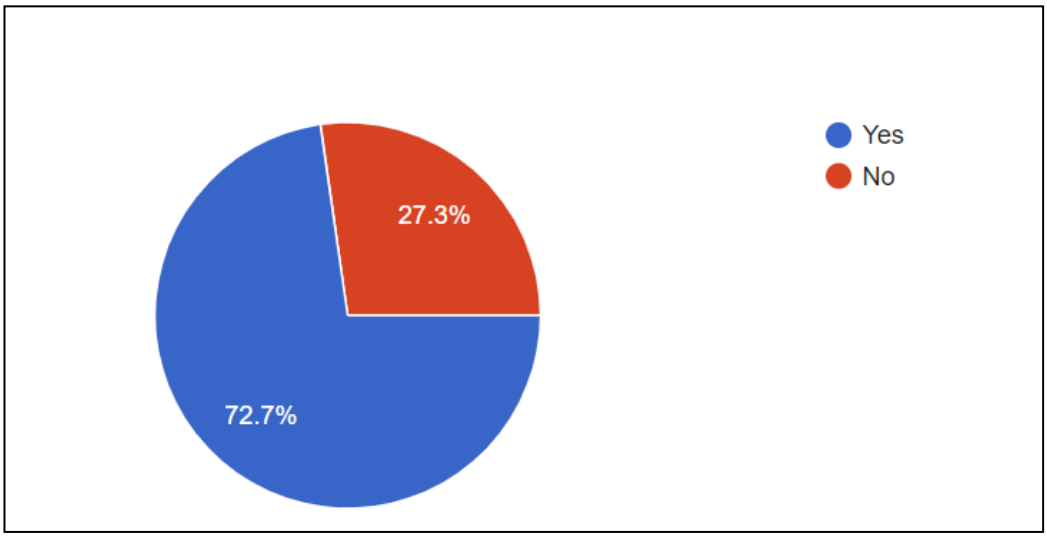

Fig. 5. Student response on whether Java helped them think more clearly about references and the abstract concepts around the structures.

3) Focus on conceptual and structural aspects

82% agreed with the statement “Having more than 1 language helped me to focus on the conceptual and structural aspects without getting stuck in the weeds of syntax.”

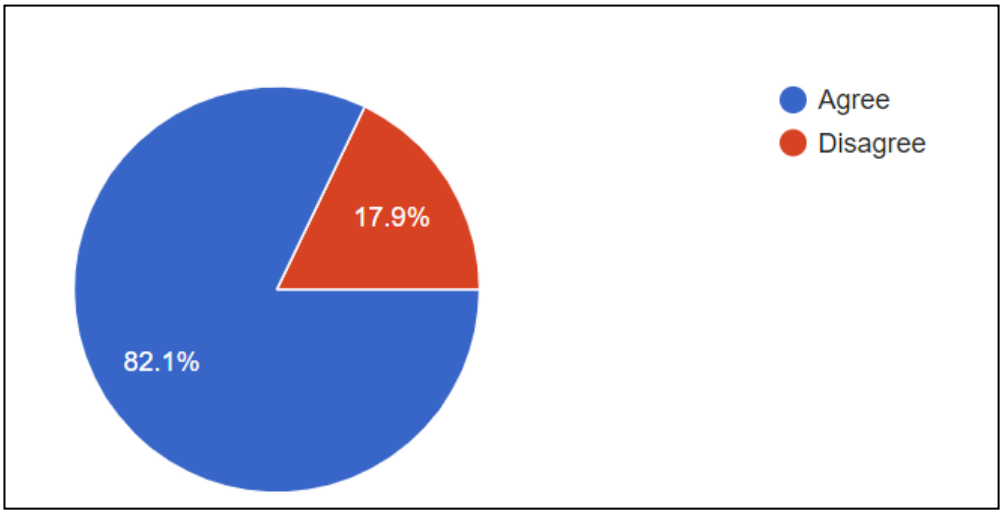

Fig. 6. Agree/Disagree: Having more than 1 language helped me to focus on the conceptual and structural aspects without getting stuck in the weeds of syntax.

C. Other takeaways from the survey

83% of students (29/35) agreed that including Java in addition to Python was the right choice. The subgroup (n=21) that had affirmatively voted to opt for including Java were surveyed on whether they now thought this was a good choice or whether they regretted making that choice. 2 students expressed regrets but did not follow-up with qualitative comments as to why.

Despite some of the results in the previous subsection when asked which language students preferred writing a data structure in, students split 54/46 Java vs Python.

D. Qualitative comments

Selected comments from qualitative questions on the survey. (Students were made aware post-survey that their responses may be used in a publication and no one expressed any reservations as long as it was anonymized.)

1) If you voted for us to include Java, what was the reason that drove you to ask for this choice?

- I wanted to broaden my knowledge of programming languages so that my resume would look more appealing.
- I wanted to learn something different from what I learned in programming 1.
- I only knew Python prior to the class and wanted to learn Java.
- Because I wanted to learn a new language and thought it would help my ability to think about algorithms.
- I did vote for Java and my reasoning was to learn a new language. However, I didn’t really get there this semester which is why I still lean towards Python. But I did like how you have to initialize what type the variable you are making is going to be. (Author: Maybe this was one of the regrets?)

2) Any other thoughts or comments?

- Coming in I only knew Python, so I wanted to do this course in Java so that I can learn it and become more familiar with it in order to expand my skill set. I would say that in general I prefer Python just because I am used to it, but I really did not mind using Java and learning to understand it. Language is nothing but a vehicle or medium to understanding different concepts and in this case programming, it is just the language in which you implement it that changes which is why I liked that we did Java since I got to learn the algorithms and structures and the "how" rather than focus on the nitty gritty syntax.
- I appreciate that you allow us to use both in most assignments as it is really helpful in the beginning to get back into the process of using java over python, but appreciate the general java taught format. Thank youI only knew Python prior to the class and wanted to learn Java.
- I think that Java was a much superior choice than Python to understand these ideas, even though I would've preferred to learn a different language than Java. C/C++ would've been especially nice, but I understand how this would've freaked out people who only had experience coding in CS-121. For that reason, I think Java was a good choice.
- I liked how the class was focused on the conceptual parts of algorithms and less on the coding.
- As someone with no prior Java knowledge, I had to learn Java code by myself and through concepts in class. Java feels better in terms of making nodes and functions versus Python. for example, making a balance tree and giving a node a left, right and parent values is a strength that Java has.
- I like the simple aspects like syntax in Python more, but Java classes are a lot more clear and easier to look at compared to Python.
- I felt that switching to Java was a good choice. I initially preferred Python because it was fresh in my mind from last semester taking CS 122 *[sic]* and the last time I had worked with Java was high school. This caused me to relearn a lot of syntax to do a lot of these assignments for this class but overall the language made it so much easier to understand how the sorting algorithms work and how to compile them.
- I had little to no knowledge of Java going into this course so it was a bit more difficult, but writing in Java really helped me understand the structures and algorithms better. Without it, I feel like It would have been harder to understand.

VI. Conculsions and Next steps

Overall, using both Python and Java in the course was hugely successful. As demonstrated in the survey results, despite having a strong preference for using Python now and in the future students nevertheless thought they learned better due to the use of both programming languages side-by-side, and particularly that Java enabled them to visualize and think through data structures better. This is unsurprising given: 1) Java's clear reference style, and 2) some peculiarities around Python object syntax (this was echoed by students saying Java helped them to understand object-oriented programming better). Students also said that use of more than one programming language enabled them to focus more on the structures and programming rather than the language and syntax. This is significant gain for higher order and abstract thinking and better positions students for gaining the desired skills from a CS2 data structures course.

Several data points suggest that there are material gains from the use of this approach: students turned in assignments on average much sooner and submitted a historically high rate of complete data structure implementations. Almost 3 times as many students as usual opted for an implementation (programming) project for their final project. This is not

only historically high for SMC, but also the highest proportion I have ever seen since offering the option of implementation (since Spring 2018).

This is a promising approach to teaching data structures and has the following key benefits:

1. Encourages higher order thinking.
2. Encourages abstract thought.
3. Develops computational problem solving and computational thinking.
4. Enables students to focus more on the data structure and programming than on syntax and language peculiarities.
5. In my opinion and the general student opinion: Some languages have better object-oriented syntax than others.

The n is too small to make any significant statistical conclusion, but I intend to continue utilizing this approach in future iterations of the course and collecting more data and to continue reporting results.

References

[1] "CS2023 – ACM/IEEE-CS/AAAI Computer Science Curricula." Accessed: Aug. 18, 2023. [Online]. Available: https://csed.acm.org/

[2] D. B. Silva, R. de L. Aguiar, D. S. Dvconlo, and C. N. Silla, "Recent Studies About Teaching Algorithms (CS1) and Data Structures (CS2) for Computer Science Students," in 2019 IEEE Frontiers in Education Conference (FIE), Oct. 2019, pp. 1–8. doi: 10.1109/FIE43999.2019.9028702.

[3] R. Lister, I. Box, B. Morrison, J. Tenenberg, and D. S. Westbrook, "The dimensions of variation in the teaching of data structures," in Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, Leeds United Kingdom: ACM, Jun. 2004, pp. 92–96. doi: 10.1145/1007996.1008023.

[4] R. D. Necaise, "Transitioning from Java to Python in CS2," 2008.

[5] W. D. Maurer, "The comparative programming languages course: a new chain of development," in Proceedings of the 33rd SIGCSE technical symposium on Computer science education, Cincinnati Kentucky: ACM, Feb. 2002, pp. 336–340. doi: 10.1145/563340.563472.

[6] R. J. Enbody, W. F. Punch, and M. McCullen, "Python CS1 as preparation for C++ CS2," in Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga TN USA: ACM, Mar. 2009, pp. 116–120. doi: 10.1145/1508865.1508907.

[7] R. J. Enbody and W. F. Punch, "Performance of python CS1 students in mid-level non-python CS courses," in Proceedings of the 41st ACM technical symposium on Computer science education, Milwaukee Wisconsin USA: ACM, Mar. 2010, pp. 520–523. doi: 10.1145/1734263.1734437.

[8] J. Gal-Ezer, T. Vilner, and E. Zur, "Has the paradigm shift in CS1 a harmful effect on data structures courses: a case study," in Proceedings of the 40th ACM technical symposium on Computer science education, in SIGCSE '09. New York, NY, USA: Association for Computing Machinery, Mar. 2009, pp. 126–130. doi: 10.1145/1508865.1508909.

[9] D. Lee, K. Hu, O. El Ariss, and K. Kwon, "Multiple Programming Languages for Improving Computational Thinking in CS1," in Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2, Toronto ON Canada: ACM, Mar. 2022, pp. 1377–1377. doi: 10.1145/3545947.3576322.

[10] O. Bälter and D. A. Bailey, "Enjoying Python, processing, and Java in CS1," ACM Inroads, vol. 1, no. 4, pp. 28–32, Dec. 2010, doi: 10.1145/1869746.1869758.

[11] T. R. Ham and R. Amini, "Learning two programming languages in one semester does not adversely affect undergraduate biomedical engineering student performance," presented at the 2017 ASEE Annual Conference & Exposition, Jun. 2017. Accessed: Jul. 17, 2024. [Online]. Available: https://peer.asee.org/learning-two-programming-languages-in-one-semester-does-not-adversely-affect-undergraduate-biomedical-engineering-student-performance

[12] R. Rybarczyk and L. Acheson, "Integrating A Career Preparedness Module into CS2 Curricula Through The Teaching C++ and Java Side-by-Side," in Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore Maryland USA: ACM, Feb. 2018, pp. 592–597. doi: 10.1145/3159450.3159552.

[13] C. L. Resch, C. G. McCune, and A. Kapoor, "Reflection and Transformational Learning in a Data Structures Course," presented at the 2021 ASEE Virtual Annual Conference Content Access, Jul. 2021. Accessed: Jul. 17, 2024. [Online]. Available: https://peer.asee.org/reflection-and-transformational-learning-in-a-data-structures-course

[14] D. Aharoni, "Cogito, Ergo sum! cognitive processes of students dealing with data structures," in Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, Austin Texas USA: ACM, Mar. 2000, pp. 26–30. doi: 10.1145/330908.331804.