Choosing Our Computing Birthplace: VSCode vs Colab as GenEd IDEs*

Elena Miller, Katy Shaw, and Zachary Dodds
Computer Science Department
Harvey Mudd College
Claremont, CA 91711
`{elmiller, kshaw, dodds}@g.hmc.edu`

Abstract

First impressions are important. The initial environment in which our computing students express themselves helps shape their foundational understanding of *what computing is*, *what it's for*, and *who participates*. This work distills experiences and insights from offering Comp1 and Comp2[1] with two different IDEs: Microsoft's VSCode and Google's Colab. We identify and describe several axes along which we compare our students' experience of these two. This effort has changed the way we offer Comp1, a degree requirement of all students at our institution, and Comp2, an optional follow-up course, required by some computationally-themed programs.

1 Choosing our hometown?

Our birthplace exerts a powerful and lifelong force[8]. Although it does not grow roots as deep as our first language, culture, and family, our *computing birthplace* – the initial environment in which we express ourselves *executably* –

*Copyright ©2023 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

[1]In this work, Comp1 and Comp2 are generic CS1/CS2 names. At our institution, Comp1 is a GenEd degree requirement of every student. Comp2 is a choice, required by some programs.

has outsized impact on our students' relationship with computing. As a formative experience, it echoes long after the syntax - and instructor - of "Comp1" have faded.

Even for computing identities born elsewhere, we choose a present-day environment for our students to share: a *computational hometown*, perhaps. That environment sets the stage for shared experiences, community-building, and students' computational identity and self-efficacy.

In light of Comp1's evolution toward the role of universal GenEd at more and more institutions and programs, this work asks, "How do we decide on a hometown - or birthplace - for our students?"

2 VSCode and Colab

Let's acknowledge: the topic of computing environments is fraught! This truth shows the importance of our computational birthplaces. We may explore - even settle - in other realms, but we are enduringly shaped by our first.

What's more, there are many factors that influence the choice of computational birthplace (or hometown). Not all these factors are under our control. Indeed, it's worth asserting that the natural analogy always holds: *We don't have to decide!* Sometimes, holding on as the world spins is all the agency we have, and we are right to allow ambient forces to decide for us.

Other times, we have more agency. As a part of a consortium of small schools, we found the opportunity to run our introductory-computing courses, Comp1 and Comp2, in two different ecosystems: Microsoft's Visual Studio Code (VSCode)[6] and Google's Colaboratory (Colab)[3]. Briefly, VSCode is a widely-used professional editor/integrated development environment (IDE); Colab is a widely-used notebook interface supporting interlaced context and code cells. The next section shares the axes along which we have compared these two (and other) IDEs for introductory college computing, especially when a GenEd required of all students.

Our conclusion is not unexpected: *There are only good birthplaces.*

Yet we find there are also real differences in the *specific strengths and drawbacks* a particular IDE conveys as an initial experience. In sharing our (ongoing) journey, we hope these axes can help other schools and instructors as, together, we invite all students into their computational futures.

3 Our Opportunity Map

Figure 1 shows a large number of code-editors and IDEs across two important axes: the size of the community of current users and the ubiquity/suitability for professional settings. There is more than a little subjective judgment along both axes. Yet we found these comparisons helpful in conceptualizing the landscape of possibilities.

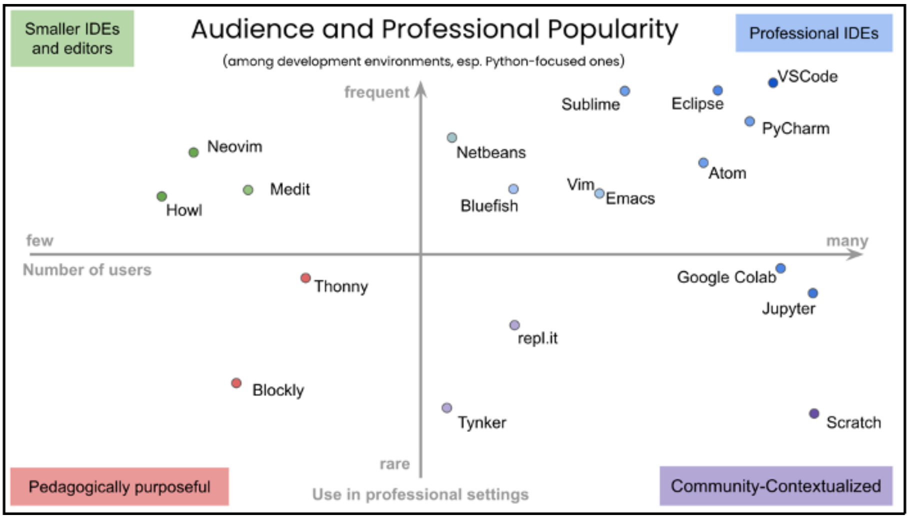

Figure 1: A comparison of several editors and IDEs (i.e., computational birthplaces), especially ones that might be used for Python/GenEd, along axes representing the relative size of the user base (horizontal) and ubiquity/suitability for professional practice (vertical). There is more than a little subjective judgment here: this provides a conceptual map of a "zeroth-order" landscape of possibilities.(Seasoned cs'ers and software engineers will appreciate that vim and emacs share a single point.)

For us, the important features of Figure 1 were (a) that both VSCode and Colab offer large communities of fellow users, and (b) that they differ in fundamental purpose. VSCode intends to serve software engineers well. It does. Colab intends to contextualize and share computational exploration. It does.

Because both environments are so successful, and because there are so many considerations that comprise a large, important, and widely shared college course, we expanded the dimensionality of our space. Figure 2 provides the twelve criteria, or "axes," along which we have contrasted VSCode and Colab. They are grouped into four rubrics: (1) sharability, important in establishing a campus-wide baseline for computing work, (2) suitability, that is, suitability for college students new to computing, (3) "sanity-preservation," which assesses logistical support for instructors and instructional realities, and (4) salability, i.e., how internal and external audiences feel about - and make themselves felt through - the interfaces.

Sharing is Caring: "Sharability Axes"	**Helpful and Real: "Suitability Axes"**	**Smooth for All: "Logistics Axes"**	**Commercial lock-in: "Salability Axes"**
Encouraging shared working-*experience*	Interface authenticity	Gradeability: Ease of assessment	Price and Accessibility
Encouraging shared work-*products*	Pathway into vs. Padding around	Universal setup and Upgradability	Behavioral "presumptiveness"
Personalization & Customization	Downstream overlap	AI integration and use	Fidelity to professional practice

Figure 2: An expanded set of axes by which we have compared the characteristics of VSCode and Colab as introductory computing environments for all students. The twelve considerations naturally group into categories of *sharability*, *Comp1- suitability*, *logistics support (instructors and students)*, and *"salability"* for each.

The next section elaborates on our - and our students' - experiences along each of these considerations.

4 Axes to Grind

4.1 Sharing is Caring

Introductory courses provide experiences shared by an entire class – in our case, an entire class-year. This common experience is important for creating connections and strengthens the sense of community within the class. Of course, *any* commonly used environment serves the purpose of **enabling shared work in subsequent courses**. For us, Colab and VSCode both have strong downstream proponents, with computational courses leaning toward VSCode and applied-science courses toward Colab notebooks. Even if there is a bigger-picture shift between years of changing code editors, the current class still shares their introduction to CS experience with all of their peers in their class. We find this an enormous advantage. *On this axis:* **Toss Up**

Sharing is caring and **communication of ideas and work products** opens the door to collaborations and community-building. As a Google document, Colabs have strong sharing features: multiple people can see and access a Colab at one time, though only one can edit at a time. As a local editor, VSCode does not support this capability out-of-the-box. However, its substantial library of extensions offers Live Share[4], a capable shared-editing experience – one we use extensively in CS3, as it requires additional set up/configuration. For our introductory Comp1/Comp2 experiences, the ease of Colab tops VSCode's deep bench. *On this axis:* **Colab**

The chance to actively **personalize/customize** one's environment goes a long way to establishing a sense of ownership over the work done there. Both VSCode and Colab offer color themes for tuning the editing experience. Here, VSCode's extensions are an accessible plus, with easy opportunities to try many options. VSCode-pets are an especially wonderful option[7]! Colab requires extensions for colorful skinning and has built-in surprises like power mode, a "Colab crab," (with Corgi and Kitty siblings, wandering the menu bar), and the marquee function. These allow additional connections between professors/TAs and students; they add elements of fun and elements of control for the students when learning CS might seem unduly unnatural *On this axis:* **Toss Up**

4.2 Suitability for Intro Computing

We have encountered strong student sentiment that College Computing/ Comp1/Comp2 courses should build different skills – and should *feel* different - from learning-to-code sites like Scratch and Tynker sometimes encountered in (pre)secondary settings. We find students receptive to required computing, as long as the toolsets and skillsets employed are **authentically representative of today's downstream work environments** along post-comp1 paths. Indeed, VSCode represents the consensus look-and-feel: its interface is the go-to background for media conveying "software development." What is more, VSCode holds today's plurality (though not majority) IDE use[5]. Colab, while not reminiscent of any particular learn-to-code site, is closer to pre-college experiences: its "cushioning," interleaving context with code cells, is popular in many data-science and natural-science contexts. We believe that Colab will be the look-and-feel of the future. For the present, *on this axis:* **VSCode**

That "cushioning" has a deeper implication: Colab does not offer students hands-on access to their own filesystems (or a *local* filesystem, in any case). VSCode provides access - in fact, insists on access - to the local filesystem, a conceptual model that is valuable *for some future paths*, though not only the ones with “computational" in their name. In short, VSCode offers a path into the universe of computing resources; Colab offers a well-padded abstraction of them. This is where audience-focus will likely carry the day: at our consortium, it’s where the largest differences hinge. *On this axis:* **Toss Up**

In a large class, it is important to consider the difference between presenting and creating results. Instructors seek to stage demos smoothly, which is where Colab shines - it is easy to set up and present results. However, when creating entire projects, VSCode wins. By having explanatory instructions intermixed with the executable code, Colab can become convoluted and confusing in demonstration settings. VSCode offers the context in a tab or panel *beside* the execution, a juxtaposition our students have come to expect . It is difficult to declare a clear winner here, as Colab is better for *reading* results, and VSCode shines when *presenting* results. *On this axis:* **Toss Up**

4.3 Logistical Axes: Keeping things running...

Whatever future workplaces might use, the constraints of the academic environment are important to instructors here-and-now. **Grading**, regardless of the philosophy or norms by which it is pursued[2], is a necessary part of a Comp1/CS1 experience. How do each of these platforms facilitate grading? Having edited and executed within VSCode, our students typically upload .py files for review. Autograders are used to grade some assignments, but far from all. As Colab is online and can be updated - and entirely lost, as does happen - we have learned to ask students to downloaded a snapshot of their notebook and submit that, instead of only its URL. This has the added advantage of maintaining a local copy on their own machines! (and ours!) Further, our Comp1 intersperses reading-responses and other types of content-creating alongside the software: Colab makes this natural. *On this axis:* **Colab**

Our Comp1 does ask students to run executable artifacts from their own machines. VSCode **installation and upgradability** has varied in ease over many semesters, but it has never been painless. Maybe this is a shared experience of frustration, unifying students and instructor against a common enemy, or maybe it’s time poorly spent. Either way, Google Colab doesn’t require setup, and students can familiarize themselves with Colab as they work through their first few computational challenges. *On this axis:* **Colab**

ChatGPT and other **AI-assistants**, e.g., github copilot, are currently better integrated within VSCode. Given the uncertainty around their incorporation into academic courses, this may be a plus or a minus. What’s more, we suspect that sooner rather than later, AI-assistance will be normalized, available, and configurable in both IDEs. For the moment, *on this axis:* **VSCode**

4.4 Commercial Concerns

VSCode and Google Colab represent not only two different companies, but two distinct **commercial ecosystems** and philosophies re: computing's future role. Microsoft owns VSCode, whereas Google owns Google Colab; Google seeks to become the computational engine that builds bridges from all disciplines' day-to-day work (via Drive and Docs) to support instances where scripting can help (via Colab). VSCode is converging to the same place, but from a software-centric starting point, expanding to embrace their users as they find themselves, perhaps unexpectedly, in the role of computational exploration. Bottom-up may not be how educational institutions are organized, but it *is* how education, and especially GenEd, is experienced: *On this axis:* **Colab**

Where the commercial ecosystems are least hidden is in the editors' **default behaviors**, which are especially important in students' earliest experiences. Both Colab and VSCode target an audience with experience, for example, with default popups that explain the parameters of built-in functions, such as print. Pedagogically, these are authentic, but unhelpful. The silver lining to their insistent attention-grabbing is that the vast majority of students automatically and unconsciously tune them out.[2] VSCode profiles allow instructors to smoothly customize their students' environments. Colab would benefit from a similar capability. *On this axis:* **VSCode**

Accessibility and price are crucial considerations. Not only is the price important when considering requiring a large number of students to use a particular platform, but it is also important after the class ends: an expensive toolset is less likely to be woven into future pursuits, even if it might help[1]. Both VSCode and Google Colab are free, for the moment. Because of their value to each titan's ecosystem, it seems likely they will be free for the foreseeable future. Both offer paid versions and extensions, whose capabilities are not important for either a GenEd introduction or, in our experience, anywhere in the undergraduate curriculum. Rather, they are far more likely to serve as a natural and familiar bridge to additional *compute-resources*, if - or when - those might be needed. The *"...you're the product..."* quip does apply, more for cloud-based Colab than locally-run VSCode. Yet fifteen years of that quip's normalization leads us to conclude, *on this axis:* **Toss Up**

5 Verdict and Evolution

If we were keeping score across the prior section's twelve axes, for our institution and cohorts, VSCode emerges the better choice on three of them, Colab on four of them, with "Toss Up" the most common categorization!

[2]Indeed, watching these message-boxes consistently fail to make it beyond their viewers' retinas is one of the silver linings of their otherwise depressingly quotidian presence.

For our situation, the experiment with using both Colab and VSCode for our GenEd Comp1 and its successor Comp2 has resulted in the following outcomes within our consortium:

- The Comp1 taken by all undergraduate students *across all academic disciplines* is best served by using Colab as its "computational hometown." Colab's ease of access, immediacy of exploration, and structural support for interlacing computing with context, explanation, and reflection – all of these well serve the foundational engagement the institution and students seek, regardless of the details of their future paths. For these students, Comp2 is also best served with Colab - or another jupyter notebook scaffolding, in order to build on the foundation established in Comp1. (VSCode offers an exceptionally capable environment for jupyter-notebooks; Comp2 can be horizon-expanding in platform, as well as computationally.)
- For the same reason, the Comp1 and Comp2 taken by our consortium's masters-degree students and other graduate students – in nominally non-cs disciplines – is also best served by a Colab-based introduction. In those programs, computing is not part of the program's identity, but a potentially valuable resource, brought to bear on problems and tasks insofar as it adds value. Here, facilitating and contextualizing exploration are the crucial criteria.
- On the other hand, the Comp1 taken by all students across *all STEM academic disciplines* is better served by a hybrid set of experiences. VSCode is our initial foundation for local file-interaction and execution, supplemented later with Colab-based explorations where that platform better supports exploration (e.g., turtle graphics) or when communicating contextualized results. A mix of environments is a challenge, but for a STEM cohort it is a worthwhile challenge *per se*: we hear from our sibling STEM departments that, because they already leverage so many different systems, their students benefit disproportionately from the adaptability that comes with successfully picking up new computational-authoring environments and solving problems in them. This is precisely our approach - and philosophy for Comp1/Comp2. (Incidentally, it's also our approach for CS1/CS2 and all the rest of our CS curriculum.)

It is not a surprise that one size does not fit all: there is no need!

All deliberately scaffolded introductions-to-computing have the opportunity to be positive. The considerations here have helped us converge on this moment's deliberate approaches, as we seek to make computing's Cambrian explosion as comfortable and positive as possible, for our students and ourselves.

6 Perspective

It would seem there is a verdict, but that the verdict is less about the "right" choice of computational IDEs and more about how rapidly computing's role is evolving in higher education. This work's taxonomy has been prompted, in part, by the momentum our institutions sense, both bottom-up and top-down, behind making Comp1 a universal General Education requirement.

As more institutions and students make computational authorship part of the fundamental literacies students practice in their college/university experience, the IDEs they use will continue to mature. The axes outlined here, we hope, will help assess such resources as we all respond, proactively, to the changes before us.

As computing instructors, we have the opportunity - and responsibility - of choosing our students' computational birthplace. Let's choose wisely!

Acknowledgements

The authors gratefully thank and acknowledge the support of NSF CISE Project #2142780 and support from Harvey Mudd College.

References

[1] S Bhattacharyya. "Is Matlab losing its Charm?" In: Analytics India Magazine (2022). `https://analyticsindiamag.com/is-matlab-losing-its-charm/`.

[2] Dan Garcia et al. "Achieving" A's for All (as Time and Interest Allow)"". In: Proceedings of the Ninth ACM Conference on Learning@ Scale. 2022, pp. 255–258.

[3] Google Colab. `https://colab.research.google.com/`.

[4] Live Share. `https://code.visualstudio.com/learn/collaboration/live-share`.

[5] Top IDE Index. `https://pypl.github.io/IDE.html`.

[6] VSCode. `https://code.visualstudio.com/`.

[7] VSCode-Pets. `https://marketplace.visualstudio.com/items?itemName=tonybaloney.vscode-pets`.

[8] World Happiness Report. `https://worldhappiness.report/`.