Over the past decade more people around the world have been wanting to learn computer programming and related skills such as data science, machine learning, and AI (Artificial Intelligence) development. While these started out as niche specialties used by small numbers of professionals, in recent years millions of people from diverse backgrounds are finding a need to learn them. Being able to write code, analyze data, and use AI can help people ranging from scientists making research discoveries to healthcare analysts devising public health policies to journalists uncovering breaking news using government datasets. However, due to the intricate and invisible nature of software code, novices face immense cognitive barriers when trying to learn these skills. As a Human-Computer Interaction (HCI) and Design researcher, I address the challenges of people learning computer programming, data science, and AI by combining theoretical foundations from Cognitive Science with software techniques from Computer Science. This interdisciplinary approach enables me to uncover learning barriers and develop scalable ways to help people overcome them.

Throughout my faculty career I have been focusing on three research questions: 1) How can visualizations reveal the invisible nature of code to support novices learning programming? 2) How can we broaden access to data science, AI, and machine learning capabilities so that more diverse types of people can learn these important skills? 3) What are the increasingly diverse goals and experiences of people learning computing more generally, and how can new forms of media support their learning? **My early faculty career work has widespread international impact on both research and practice, resulting in highly-cited publications (my h-index is 48 on Google Scholar), 5 Best Paper Awards, 6 Honorable Mention Paper Awards, and one of the most widely-used and built-upon tools for learning programming – the Python Tutor code visualizer – with 20+ million users from over 180 countries throughout the past decade.**

1 Visualization-Based Scaffolding for Learning Computer Programming

My most impactful line of research involves developing, deploying, and evaluating visualizations to help people learn computer programming. A foundational challenge that everyone faces when learning programming is forming accurate mental models of what happens step-by-step as the computer runs code, which is hard since code execution is invisible. My work investigates new ways to use automatically-generated visualizations to reveal the otherwise-invisible nature of code in order to scaffold novice learning.

Python Tutor [1] for visualizing programming fundamentals: For the past 15 years I have been developing what is now the most widely-used and frequently-cited educational platform for visualizing the fundamentals of how computer code runs step-by-step. This platform, called Python Tutor, was originally made for the Python language but has expanded to Java, C, C++, and JavaScript, which collectively encompass the most popular languages used in educational settings. It innovates upon prior work via a novel visual grammar that automatically generates consistent diagrams for code in different programming languages.

Python Tutor has had lasting global impact on both research and practice. In terms of impact on practice, ***over 20 million people from over 180 countries have used it over the past decade to learn programming.*** In terms of research, it has contributed to over 55 publications from 35 research labs across 13 countries [1] – either by others building extension tools upon it or using it as a comparison for their experiments. Although my lab published several extensions [2, 3], the majority of Python Tutor-related publications come from outside of my lab. Its widely-adopted visual grammar serves as the foundation for other labs' projects and has become a standard comparison target in empirical studies. My publication summarizing its design principles and impact [1] won ***Best Paper Award at UIST 2021, the top venue for HCI technology research.***

Porta [4] for visualizing software tutorials: Once someone has learned programming fundamentals (using tools like Python Tutor), they move onto trying to build real-world software applications. To do so, they often follow tutorials for setting up and coding using a mix of software packages, libraries, and APIs. The problem is that tutorials frequently have hard-to-follow parts or leave out crucial details since it is impossible for tutorial creators to predict all the kinds of mistakes that novices may make when following along.

To alleviate this long-standing problem, I created a new visualization technique called *tutorial profiling*, which shows tutorial creators what a large number of learners are doing when following their tutorial and

writing associated computer code, then displays visualizations to hone in on which parts commonly lead to struggles. My lab developed an automated learner tracking system called Porta (Profiling Operating-system Recordings for Tutorial Assessment) to evaluate this idea [4], which won ***Best Paper Award at UIST 2018.***

Visual scaffolding for web development: Throughout the 2010s web browsers became a dominant platform for software development. However, a mess of intricate technical complexities make it hard for novices to get started on creating web software. During that time period, my lab did some of the foundational systems research on visual scaffolding to lower such barriers to web app development. Two representative systems here include CodePilot [5] (CHI 2017) and Fusion [6] (UIST 2018). CodePilot was the first IDE (Integrated Development Environment) to integrate real-time collaboration with coding, testing, visual previews, bug reporting, and version control [5]. Fusion introduced the concept of user interface (UI) mashups where novices can quickly prototype new web interfaces by adapting UI features from existing websites and combining them with visual scaffolding [6]. Ideas from these systems have impacted the state of practice in the past few years by being integrated into IDE products from companies such as Google and Microsoft.

Ongoing work: Thus far the visualization tools from my lab track how each line of code executes, but more advanced programmers draw diagrams at higher levels of abstraction to match their individual problem domain rather than the details of step-by-step execution. To infer a design space for these bespoke visualizations, we performed the first study of thousands of ASCII text-based diagrams embedded in source-code comments [7] (***Honorable Mention Paper Award at CHI 2024***). Inspired by the designs we discovered, we are extending tools like Python Tutor [1] to render higher-level visualizations and semantic explanations.

2 Broadening Access to Data Science, AI, and Machine Learning

Beyond programming, many people from diverse backgrounds are now learning data science, AI, and machine learning as they are finding these skills to be more in demand. However, the problem is that the tools commonly used in these fields were originally designed for use by experts who often have PhDs in mathematical specialties. This reality imposes a significant barrier to entry for the large numbers of novices who are now entering these fields. To address this inequity, my research broadens access to these skills by both studying the challenges faced by learners and inventing new tools to lower their barriers to entry.

Learning across diverse settings: To inform the design of new tools, we conducted a series of interview and survey studies that revealed challenges faced by people learning data science, AI, and machine learning across diverse settings. These include studies of both formal university courses [8, 9] and informal settings such as volunteer-run workshops [10] and online tutorials [11, 12]. Highlights here include the first study of a fast-growing phenomenon of software engineers teaching themselves AI and machine learning to keep up with rapidly-changing job markets [11] (***Best Paper Award***), and the challenges that industry practitioners face when teaching data science via workshops and online courses [10] (***Honorable Mention Paper Award***). These study findings inspired us to create tools like the following, which aim to lower the barriers to learning.

Programming-by-demonstration for data scientists: A classic technique for lowering barriers to writing code is programming-by-demonstration, in which a user demonstrates their intended actions then a tool automatically generates code for them. In the past decade my collaborators and I were amongst the first to build programming-by-demonstration systems to help the growing population of novice data scientists. My two most impactful systems along this trajectory are Wrex [13] (CHI 2020) and Bespoke [14] (UIST 2019).

Wrex [13] lets data scientists demonstrate how they want to clean up a dataset step-by-step, and then it generates readable Python code that they can edit. Wrex makes data cleaning more accessible to novices while scaffolding their learning of Python data science features. Co-developed with Microsoft Research, our paper on Wrex won ***Best Paper Award at CHI 2020***, has over 147 citations so far, and has impacted the state of practice by influencing the design of Microsoft Excel's integration of Python into spreadsheets.

Bespoke [14] helps novice data scientists and computational researchers (e.g., computational biologists) work with command-line tools such as scripts and data processing utilities. It allows an expert to demon-

strate running a set of command-line tools and then automatically generates a bespoke GUI (Graphical User Interface) that novices can use to run their workflows more easily than using the command line. Bespoke's impact includes other PIs' labs adapting its GUI-generation ideas into recent projects such as Harvard's *DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing* (Best Paper Award at CHI 2024).

A new twist on computational notebooks: Many data scientists and AI researchers now work within computational notebooks such as Jupyter; my lab wrote a widely-cited survey paper [15] that analyzed design variations among 60 notebook systems developed in both academia and industry. We discovered that a major usage barrier for novices is that they must configure custom libraries and then acquire and import their data into notebooks, at which point that data becomes separated from its original source context. To lower this barrier, my lab invented a new kind of *contextualized computational notebook* where we embed notebooks in-situ within any webpage that a user is viewing in their browser. This enables them to do data science and machine learning directly in the context of source webpages (e.g., analyzing statistics embedded in a Wikipedia article). We built and published two systems to evaluate this idea: DS.js for data science [16] (***Honorable Mention Paper Award at UIST 2017***) and Mallard for machine learning [17] (UIST 2019).

Ongoing work: Broadening out from technical tools research, my latest work along this direction focuses on the important but under-studied role of non-technical factors in data science and machine learning. These include new collaboration workflows [18] (CSCW 2021) and, most recently, how Generative AI (e.g., tools like ChatGPT) can help scientists with higher-level planning and strategy [19]. I have been building momentum on this trajectory with two recent Sloan Foundation grants (one for studying collaboration and another for Generative AI) and a Microsoft Research grant to use Generative AI as a data science consultant.

3 Diverse Goals, Experiences, and New Media for Learning Computing

Zooming out more, my broader research agenda encompasses how people learn computing – a more general field that includes not only programming, data science, and AI but also human-computer interaction (HCI) and user experience (UX) design. Prior research here often focused on people who are learning computing in formal school settings (e.g., university Computer Science departments) with the goal of becoming professional programmers. In contrast, my work innovates upon the status quo by studying learners from diverse age and demographic groups, with widely-varying goals and experiences that differ from typical Computer Science students, and using new forms of media beyond traditional lectures and textbooks.

Experiences of learners from diverse demographics: Over the past decade my collaborators and I published some of the most-cited papers in our field about challenges faced by computing learners from several diverse demographics (all these papers have over 90 citations): women programmers on online forums such as Stack Overflow [20], older adults aged 60 and over [21] (***Honorable Mention Paper Award at CHI 2017***), non-native English speakers around the world [22] (CHI 2018), and the differing experiences of women and men at college hackathons [23]. We also published the first account of an undergraduate female student's experiences as a peer tutor for computing classes [24] (***Best Paper Award at SIGCSE 2021***).

Conversational programmers: We also discovered a new category of learners called *conversational programmers* whose goal is to learn coding to communicate better with their programmer colleagues, even though they do not need to write code themselves. Conversational programmers are more diverse than their peers [25] (e.g., more women, more humanities and social sciences majors) but are not served well by existing learning resources, which focus too much on low-level logic of code rather than higher-level mental models of how software achieves user-facing goals [26] (***Honorable Mention Paper Award at CHI 2018***). This work has had three kinds of impact: 1) as a theoretical framework to extend the long-standing notion of end-user programmers, 2) as the foundation for follow-up empirical work by other labs such as Cunningham et al. (CHI 2021, Best Paper at SIGCSE 2022, ICER 2024), and 3) as a theoretical basis for new educational initiatives such as U. Michigan's interdisciplinary Program in Computing for the Arts and Sciences.

New media for teaching computing: We have also studied and built systems to support teaching via livestreaming on platforms like Twitch [27, 28] (***Honorable Mention Paper Award***), mixed-media tutorials that combine code examples and short video clips [29, 30], a new presentation format that mixes premade slides with live coding [31], and how artists use media when teaching computing within an art context [32].

Ongoing work: A new form of media that has been on many people's minds lately is using Generative AI (e.g., ChatGPT) to teach and learn programming. Last year I published the first study of how computing instructors plan to adapt to the growing prevalence of this new technology, highlighting both risks and opportunities for pedagogy [33]. This is now the most cited paper that was published at ICER 2023 (out of 35 total papers), with over 96 citations in its first year. I am continuing this line of work to study Generative AI's impact on teachers and learners, funded in part by the Sloan Foundation. Another line of work here is improving diversity in HCI and UX education via inclusive pedagogical design [34] and peer-led mentoring to teach the hidden curriculum of skills that are not normally taught in classes [35]. I am expanding this work with a grant from Google for broadening participation amongst underrepresented STEM students.

Summary of Service and Leadership

In terms of external service, two notable leadership roles I held were as ***Program Committee co-chair for the 2023 L@S (ACM Learning at Scale) and 2023 VL/HCC (IEEE Visual Languages and Human-Centric Computing) conferences.*** Both roles involved recruiting and working with dozens of international researchers (75 for L@S and 40 for VL/HCC) to peer-review paper submissions and decide on the set of accepted papers. In addition, I am on the Learning at Scale conference Steering Committee (2023–present) and recently finished an elected four-year term on the VL/HCC conference Steering Committee (2019–2023). Both roles involved consulting with each year's conference organizers to pass down institutional knowledge and ensure thematic continuity. I was also the 10-Year Most Influential Paper Award committee co-chair for VL/HCC (2021, 2022) and the Best Paper Award committee co-chair for UIST (2019). And I regularly serve on the program committees of international research conferences such as CHI, UIST, ICER, VL/HCC, Learning at Scale, and Educational Data Mining.

Within UC San Diego, my most notable service accomplishments have been related to my long-standing role as the Undergraduate Education Chair (called the *faculty undergraduate advisor*) in my department (2018–2023) where I work with our staff to meet curricular needs for our 2,100+ undergraduate majors. I represented my department at university-level councils related to undergraduate education. Relatedly, I have also been a long-term member of the university-side Instructional Technology Governance Committee (2016–present). During COVID lockdowns I was our department's faculty lead on instructional technology for the transition to emergency remote teaching. Lastly, I led two major educational accreditation efforts: 1) 2018–2019: I was the *WASC Senior College and University Commission (WSCUC)* campus-wide accreditation representative from the Cognitive Science Department where I wrote a comprehensive report about our educational programs for external reviewers, 2) 2022–2024: I led a committee of 4 faculty to co-author a 100-page self-evaluation report for the UCSD Cognitive Science Department to submit to university-level and external evaluators as part of our department program review that occurs once every 8 years. This was a longer-term service commitment that required coordinating many faculty, staff, and students to put together a comprehensive view of our department's strengths, areas of improvement, and forward-looking plans.

Summary of Research Mentoring

In total I have been the primary advisor for 10 Ph.D. students so far (3 MS/Ph.D. at the University of Rochester and 7 at UC San Diego). My Ph.D. graduates have obtained their first jobs in both industry and academia at institutions such as Meta Research (Xiong Zhang), Microsoft Research (Ian Drosos, postdoc, currently applying to academic jobs), Fred Hutch Cancer Research Center (Sean Kross), and UC San Diego (Sam Lau, tenure-track assistant teaching professor). Currently at UCSD I am advising 3 Ph.D. students in cognitive science and on the committees of 10 Ph.D. students in cognitive science and computer science.

I have also advised over two dozen undergraduate and masters students on research, with many becoming the lead authors on my lab's publications and then going onto notable Ph.D. programs in HCI and computer science. My undergraduate and masters lab alumni include Mitchell Gordon (Ph.D. at Stanford, now an assistant professor at MIT), Jeremy Warner (Ph.D. at UC Berkeley, now a research scientist at Apple), Hyeonsu Kang (Ph.D. at CMU), Dan Scarafoni (Ph.D. at Georgia Tech), Priyan Vaithilingam (Ph.D. at Harvard), Julia Markel (Ph.D. at Stanford), and Kendall Nakai (incoming Ph.D. student at MIT).

My students have also won competitive awards such as the NSF CISE CSGrad4US Graduate Fellowship (Kendall Nakai) and the CRA Outstanding Undergraduate Researcher Award (Mitchell Gordon).

Selection of Faculty-Era Publications – full publications list at https://pg.ucsd.edu/

[1] Philip J. Guo. Ten million users and ten years later: Python Tutor's design guidelines for building scalable and sustainable research software in academia. In *ACM Symposium on User Interface Software and Technology*, UIST, 2021 **Best Paper Award**.

[2] Philip J. Guo. Codeopticon: Real-time, one-to-many human tutoring for computer programming. In *ACM Symposium on User Interface Software and Technology*, UIST, 2015.

[3] Hyeonsu Kang and Philip J. Guo. Omnicode: A novice-oriented live programming environment with always-on run-time value visualizations. In *ACM Symposium on User Interface Software and Technology*, UIST, 2017.

[4] Alok Mysore and Philip J. Guo. Porta: Profiling software tutorials using operating-system-wide activity tracing. In *ACM User Interface Software and Technology*, UIST, 2018 **Best Paper Award**.

[5] Jeremy Warner and Philip J. Guo. CodePilot: Scaffolding end-to-end collaborative software development for novice programmers. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2017.

[6] Xiong Zhang and Philip J. Guo. Fusion: Opportunistic web prototyping with UI mashups. In *ACM Symposium on User Interface Software and Technology*, UIST, 2018.

[7] Devamardeep Hayatpur, Brian Hempel, Kathy Chen, William Duan, Philip J. Guo, and Haijun Xia. Taking ASCII drawings seriously: How programmers diagram code. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2024 **Honorable Mention Paper Award**.

[8] Sam Lau, Deborah Nolan, Joseph Gonzalez, and Philip J. Guo. How computer science and statistics instructors approach data science pedagogy differently: Three case studies. In *ACM Technical Symposium on Computer Science Education*, SIGCSE, 2022.

[9] Sam Lau, Justin Eldridge, Shannon Ellis, Aaron Fraenkel, Marina Langlois, Suraj Rampure, Janine Tiefenbruck, and Philip J. Guo. The challenges of evolving technical courses at scale: Four case studies of updating large data science courses. In *ACM Conference on Learning @ Scale*, L@S, 2022.

[10] Sean Kross and Philip J. Guo. Practitioners teaching data science in industry and academia: Expectations, workflows, and challenges. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2019 **Honorable Mention Paper Award**.

[11] Carrie J. Cai and Philip J. Guo. Software developers learning machine learning: Motivations, hurdles, and desires. In *IEEE Visual Languages and Human-Centric Computing*, VL/HCC, 2019 **Best Paper Award**.

[12] Rimika Chaudhury, Philip J. Guo, and Parmit K. Chilana. "There's no way to keep up!": Diverse Motivations and Challenges Faced by Informal Learners of ML. In *IEEE Visual Languages and HCC*, VL/HCC, 2022.

[13] Ian Drosos, Titus Barik, Philip J. Guo, Robert DeLine, and Sumit Gulwani. Wrex: A unified programming-by-example interaction for synthesizing readable code for data scientists. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2020 **Best Paper Award**.

[14] Priyan Vaithilingam and Philip J. Guo. Bespoke: Interactively synthesizing custom GUIs from command-line applications by demonstration. In *ACM Symposium on User Interface Software and Technology*, UIST, 2019.

[15] Sam Lau, Ian Drosos, Julia M. Markel, and Philip J. Guo. The design space of computational notebooks: An analysis of 60 systems in academia and industry. In *IEEE Visual Languages and HCC (VL/HCC)*, 2020.

[16] Xiong Zhang and Philip J. Guo. DS.js: Turn any webpage into an example-centric live programming environment for learning data science. In *ACM Symposium on User Interface Software and Technology*, UIST, 2017 **Honorable Mention Paper Award**.

[17] Xiong Zhang and Philip J. Guo. Mallard: Turn the web into a contextualized prototyping environment for machine learning. In *ACM Symposium on User Interface Software and Technology*, UIST, 2019.

[18] Sean Kross and Philip J. Guo. Orienting, framing, bridging, magic, and counseling: How data scientists navigate the outer loop of client collaborations in industry and academia. *CSCW conference + PACM HCI journal*, 2021.

[19] Philip J. Guo. Six Opportunities for Scientists and Engineers to Learn Programming Using AI Tools Such as ChatGPT. *Computing in Science & Engineering*, 25(3), 2023.

[20] Denae Ford, Justin Smith, Philip J. Guo, and Chris Parnin. Paradise unplugged: Identifying barriers for female participation on Stack Overflow. In *ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE, 2016.

[21] Philip J. Guo. Older adults learning computer programming: Motivations, frustrations, and design opportunities. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2017 **Honorable Mention Paper Award**.

[22] Philip J. Guo. Non-native English speakers learning computer programming: Barriers, desires, and design opportunities. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2018.

[23] Jeremy Warner and Philip J. Guo. Hack.edu: Examining how college hackathons are perceived by student attendees and non-attendees. In *ACM Conference on International Computing Education Research*, ICER, 2017.

[24] Julia M. Markel and Philip J. Guo. Inside the Mind of a CS Undergraduate TA: A Firsthand Account of Undergraduate Peer Tutoring in Computer Labs. In *ACM Technical Symposium on Computer Science Education*, SIGCSE, 2021 **Best Paper Award**.

[25] Parmit K. Chilana, Rishabh Singh, and Philip J. Guo. Understanding conversational programmers: A perspective from the software industry. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2016.

[26] April Y. Wang, Ryan Mitts, Philip J. Guo, and Parmit K. Chilana. Mismatch of expectations: How modern learning resources fail conversational programmers. In *ACM Conference on Human Factors in Computing Systems*, CHI, 2018 **Honorable Mention Paper Award**.

[27] Ian Drosos and Philip J. Guo. Streamers teaching programming, art, and gaming: Cognitive apprenticeship, serendipitous teachable moments, and tacit expert knowledge. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. short paper, 2021 **Honorable Mention Paper Award**.

[28] Ian Drosos and Philip J. Guo. The design space of livestreaming equipment setups: Tradeoffs, challenges, and opportunities. In *ACM Designing Interactive Systems Conference*, DIS, 2022.

[29] Alok Mysore and Philip J. Guo. Torta: Generating mixed-media GUI and command-line app tutorials using operating-system-wide activity tracing. In *ACM User Interface Software and Technology*, UIST, 2017.

[30] Kandarp Khandwala and Philip J. Guo. Codemotion: Expanding the design space of learner interactions with computer programming tutorial videos. In *ACM Conference on Learning at Scale*, L@S, 2018.

[31] Charles Chen and Philip J. Guo. Improv: Teaching programming at scale via live coding. In *ACM Conference on Learning at Scale*, L@S, 2019.

[32] Alice M. Chung and Philip J. Guo. Perpetual teaching across temporary places: Conditions, motivations, and practices of media artists teaching computing workshops. In *ACM Conference on International Computing Education Research*, ICER, 2024.

[33] Sam Lau and Philip J. Guo. From "Ban It Till We Understand It"' to "Resistance is Futile"': How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *ACM Conf. on International Computing Education Research*, ICER, 2023.

[34] Sean Kross and Philip Guo. Five pedagogical principles of a user-centered design course that prepares computing undergraduates for industry jobs. In *ACM Technical Symposium on Computer Science Education*, SIGCSE, 2022.

[35] Kendall Nakai and Philip J. Guo. Uncovering the hidden curriculum of university computing majors via undergraduate-written mentoring guides: A learner-centered design workflow. In *ACM Conference on International Computing Education Research*, ICER, 2023.