Older Adults Learning Computer Programming: Motivations, Frustrations, and Design Opportunities

Philip J. Guo
UC San Diego
La Jolla, CA, USA
pg@ucsd.edu

ABSTRACT

Computer programming is a highly in-demand skill, but most learn-to-code initiatives and research target some of the youngest members of society: children and college students. We present the first known study of older adults learning computer programming. Using an online survey with 504 respondents aged 60 to 85 who are from 52 different countries, we discovered that older adults were motivated to learn to keep their brains challenged as they aged, to make up for missed opportunities during youth, to connect with younger family members, and to improve job prospects. They reported frustrations including a perceived decline in cognitive abilities, lack of opportunities to interact with tutors and peers, and trouble dealing with constantly-changing software technologies. Based on these findings, we propose a learner-centered design of techniques and tools for motivating older adults to learn programming and discuss broader societal implications of a future where more older adults have access to computer programming – not merely computer literacy – as a skill set.

Author Keywords

older adults; learning programming; computational literacy

ACM Classification Keywords

K.3.2 Computers and Education: Computer and Information Science Education – *Literacy*

INTRODUCTION

Computer programming is now such an in-demand skill that government and corporate leaders are issuing calls for widespread programming education. For instance, countries such as the U.K. and Japan, along with major U.S. cities such as Chicago, San Francisco, and New York City, have made pledges to offer programming classes in all public primary and secondary (*K-12*) schools [15, 46, 55]. The CEO of General Electric, one of the world's largest companies, recently stated that he wanted all employees to learn this skill: *"If you are joining the company in your 20s, unlike when I joined, you're going to learn to code. It doesn't matter whether you are in sales, finance or operations. You may not end up being a programmer, but you will know how to code."* [32]

So far, the vast majority of these learn-to-code initiatives have focused on the youngest members of society: those under the age of 25. There is an abundance of research on techniques and tools for teaching programming to students in elementary school [20, 48], middle school [23, 33], high school [34, 61], and college [10, 26, 51], and to training new employees in the workforce [6]. In recent years, HCI researchers have also started studying motivations for learning programming amongst mid-career professionals, usually in their 30s and 40s [12, 14, 17]. However, a notable portion of the population has not been mentioned in any of these initiatives: older adults who are retired or in the latter stages of their careers.

Older adults now comprise a significant and rapidly-growing fraction of the global population. The United Nations estimates that by 2030, over 25% of the populations of North America and Europe (and 16% of the overall world population) will be at least 60 years old [41]. The HCI research community has long recognized the importance of this age group by studying how they interact with and learn to use a wide variety of computational technologies [5, 8, 9, 30, 35, 36, 37, 39, 43, 44, 53, 58, 60]. Yet there has been no research on how older adults learn to use one of the most expressive, foundational, and powerful technologies of all: programming languages that enable them to directly control computers.

Why care about older adults learning computer programming in particular? Programming skills can empower this large and fast-growing population to: 1.) improve their quality of life and social ties by engaging in this challenging, creative, and collaborative activity, 2.) maintain gainful part- and full-time employment as they age, which again improves quality of life and further diversifies the currently youth-skewed technology workforce [11, 31, 59], and 3.) serve as a potential pipeline of K-12 teachers necessary for educating future generations about computing in a scalable way, which helps alleviate the current shortage in primary and secondary schools [16]. In addition, we believe that recent efforts to broaden participation and inclusion in computing [21, 25] should expand to encompass the now-underserved population of older adults. However, before taking steps to achieve any of these goals, we must first understand why and how they are now learning.

CHI 2017, May 06–11, 2017, Denver, CO, USA

ACM 978-1-4503-4655-9/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025945

Thus, this paper's main contribution is presenting *the first known study of older adults learning computer programming*. We deployed an online survey to a programming education website (pythontutor.com) and analyzed data from 504 respondents aged 60 to 85 who are from 52 different countries. We investigated why older adults are learning programming, how they are learning, and what frustrations they are facing.

We found that older adults were often motivated by age-related reasons such as keeping their brains challenged as they aged, making up for missed learning opportunities during their youth, connecting with younger family members, and improving their job prospects. They frequently used free online resources such as MOOCs, web-based tutorials, and digital textbooks. Finally, they reported a diverse variety of frustrations, including age-related challenges such as a perceived decline in cognitive abilities, lack of social opportunities to interact with tutors and peers, and trouble dealing with constantly-changing software technologies.

Based on these findings, we propose a learner-centered design [25] of techniques and tools to motivate older adults to learn programming, using these learners' motivations and frustrations as a starting point. Example design ideas include contextualizing project-based programming lessons to domains of high interest to this population (e.g., digital healthcare, storytelling with personal life narratives), leveraging their prior knowledge of older technologies to make bridging analogies to newer technologies that they are now learning, and developing learning resources and programming aids that take age-related cognitive impairments into account.

In sum, this paper's contributions to HCI are:

- An empirical study of older adults learning computer programming, based on survey responses from 504 online learners who are 60 to 85 years old and from 52 countries.
- A learner-centered design of techniques and tools to motivate older adults to learn programming, and potential societal implications of more older adults doing programming.

RELATED WORK

The study we present in this paper fills a gap in two major lines of HCI work: 1.) research on learning computer programming that targets specific age groups, and 2.) studies of older adults' use of computational technology. To our knowledge, no prior work has attempted to bridge those two areas to study how older adults learn computer programming.

Age-Targeted Research on Learning Programming

Although there is a vast, broad, and mature literature on tools and techniques for learning programming, here we focus only on research that targets specific age groups.

Programming education for elementary-school-aged children is motivated by engaging learners through gaming, storytelling, or "think like a computer" activities. Classic work such as Papert's Mindstorms [48] used the Logo language with embodied turtle graphics to teach algorithmic and creative thinking to young children. Blocks-based programming languages popularized by Scratch [1] and Android App Inventor [62] make programming more accessible to novices by eliminating syntax errors and by tying it to domains such as interactive storytelling and mobile app development, respectively. Although these tools can be used by all age groups, their development is mainly driven by the needs of children. Scratch is designed for ages 8 to 16 [1], and a simplified mobile-device version called ScratchJr is for ages 5 to 7 [20].

Programming education for middle- and high-school-aged children often use higher-fidelity environments that incorporate 3-D microworlds and programmable virtual agents. For example, Storytelling Alice [33], and its descendant Looking Glass [23], enable students to create 3-D stories situated in virtual worlds, and have been especially engaging to female middle-school students in particular [33]. Descendants of Logo such as StarLogo TNG [34] and NetLogo [61] have been used to teach mathematics, network phenomena, and experimental science in middle and high school classrooms.

The majority of age-specific research on learning programming currently focuses on college students (e.g., ages 18 to 22), due in part to this learner population being the most readily-accessible to university researchers and to it being the direct pipeline to the technology workforce. Significant initiatives here include diversifying the population of computer science (CS) majors [10], preparing and retaining students in introductory CS1 courses [51], and teaching programming effectively to non-CS majors [26].

Research on working-aged adults learning programming often focuses on vocational needs. People have studied why professionals such as graphic and web designers [14], computer science teachers [17], and sales, marketing, and product managers [12] learn to code on-the-job. Continuing education is critical since people in the tech industry must constantly retrain to learn new, fast-evolving programming technologies to stay competitive on the job market. An age-related undercurrent is technology workers over 40 facing issues of age discrimination as the median age in top firms (e.g., Apple, Facebook, Google) hovers around 30 years old [11, 31, 59].

All of the aforementioned work share a human-centered tradition of first trying to discover the motivations, values, and needs of each age group before designing tools and interventions for them. Our work in this paper extends this long disciplinary tradition to a population that, to our knowledge, has not been studied in prior work: older adults aged 60 and over.

Older Adults' Use of Technology

The other main family of work that inspired our study is HCI research on how older adults use computational technologies, a topic that has grown popular over the past decade. A 2015 survey paper found that of the 162 papers in SIGCHI venues that are primarily about aging and technology, 80% of them have been published since 2007 [57].

Most research in this area have focused on studying older adults as consumers of technology. These include uses of digital resources of especially high interest to this population such as health websites [37], healthcare patient portal

sites [35], and electronic presence indicators for remote caretakers [5]. It also includes studying how older adults use – or choose not to use – widespread technologies that are popular with all age groups, including social networking sites such as Facebook [30, 44], ridesharing services [39], mobile devices [36], and video games [58]. Recurring themes in such work include older adults using technology to sustain good quality of life as they age, to ward off social isolation, and to maintain connections with family members.

Of more direct relevance to our own work is research that positions older adults as producers rather than consumers of digital content. Creation-based activities that have been studied for this age group include blogging [9], performing microtask crowd work [8], making digital music with an Arduino-based DIY toolkit [53], participating in online discussion forums [43], and sharing photos and messages with iPad apps [60]. These studies surfaced participant feelings such as personal empowerment, creative self-expression, meaningful engagement, and joy, in contrast to the more utilitarian life-maintenance themes revealed by consumer-based studies of technology and aging. However, these activities still involve older adults using existing apps to create content, sometimes under experimenter supervision, rather than programming entirely new software applications on their own by writing code.

Our work follows a progression in this literature from studying older adults first as consumers of technology, then as producers of digital content, and now to potentially becoming producers of new technologies via computer programming. In the Discussion section, we will reflect in detail on some of the similarities between our findings and what was discovered by these prior studies of older adults' use of technology.

METHODOLOGY: INTERNATIONAL ONLINE SURVEY

For this study, we wanted to reach a broad global population of learners, so we deployed an online survey to the programming education website `pythontutor.com` (Python Tutor [24]). Learners use this website to practice writing code and to visually debug their errors using a step-by-step run-time data structure visualizer. Despite its legacy name, Python Tutor now supports learning several popular programming languages including Java, C, C++, Ruby, and JavaScript. It has been operating for the past six years and has had over 2.5 million total users from over 180 countries.

We deployed our survey to the Python Tutor website because it is a free and widely-used resource for learners who come to the site from a diverse variety of online learning channels such as: Massive Open Online Courses (MOOCs) by all three major providers (Coursera, edX, Udacity), Khan Academy, Codecademy, Stack Overflow, digital textbooks, and many coding tutorials, blogs, and discussion forums. Thus, even though a survey on any single website will reach only a limited population, deploying to Python Tutor reaches a broader population than say, deploying to a single MOOC, since users come to Python Tutor from an array of referring websites.

The Survey Instrument

We added the following sentence to the bottom of the Python Tutor website with a link to a survey hosted on Google Forms: "If you are at least 60 years old and would like to help our research on how older people learn programming, please fill out this survey [link URL]." We made the survey link legible but unobtrusive so as not to unnecessarily distract learners on the site. Participation was voluntary; we did not pay survey respondents. We opened this survey in March 2015 and closed it in August 2016.

We made our survey as short as possible to collect basic demographic information and self-reflections about why, how, and when people were learning. It contained ten questions:

1. What is your age in years?
2. Where do you live? (e.g., country, state, city)
3. What is your gender?
   Choices: {Female, Male, Other, Prefer not to say}
4. What is your current employment status? Choices: {Working, Retired, Semi-retired, Other (free-response)}
5. What is/was your job/occupation/profession?
6. Why are you learning computer programming? (e.g., what is your motivation for learning?)
7. How long have you been trying to learn programming?
8. How many hours per week do you devote to learning programming?
9. What resources are you currently using to learn programming?
10. What has been the most frustrating thing so far about learning programming?

To give respondents the most flexibility, all questions except for gender (Question 3) and employment status (Q4) were open-ended (free-response), and none were mandatory.

Age-60 threshold: Although there is no universally agreed-upon definition of what "older adult" means, we chose age 60 as the threshold for this study so that we could get a mix of working (late-stage career), semi-retired, and retired people in our respondent pool. The United Nations has also adopted age 60 as a threshold for "older persons" in their World Population Ageing report [41]. Also, the retirement age across many countries is around 60 to 65 [54], and in the U.S., employees are eligible to withdraw their 401k/IRA retirement savings at age 59.5 and Social Security Benefits at age 62.

Data Overview and Analysis

After inspecting all responses, we filtered out several that were either spam or filled out by people younger than age 60 who did not understand the survey's instructions. We received 504 valid responses, which we manually coded. To classify occupation types (Q5), we used the International Standard Classification of Occupations (ISCO-08) [47] adopted by the United Nations. To classify motivations for learning programming (Q6), resources used to learn (Q9), and frustrations (Q10), we iteratively developed a set of codes based on an inductive analysis approach [13], which resulted in around a dozen categories for each question (see Tables 2, 3, and 4). The sole author coded all survey responses side-by-side with a research assistant who is currently learning programming in order to iterate on the codes and to double-check the results.

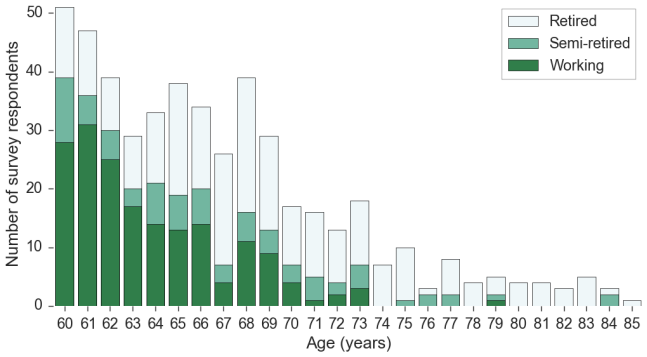

Figure 1. Age distributions of the 504 people aged 60 and over who responded to our online survey about older adults learning computer programming. The median age was 65.5 years; mean age was 66.5.

Study Design Limitations

The goal of this study was to capture an overview of older adults learning computer programming. Ideally we would uniformly sample from *all people* around the world aged 60 and above who are learning programming. But since we deployed this survey to a U.S.-made English website (Python Tutor), which is frequently linked to by other English-centric websites (e.g., MOOCs, Stack Overflow, coding tutorials), our sample is skewed toward people from English-speaking countries or those with English literacy skills [28].

Additionally, our sample likely comes from the more autodidactic, technology-literate, self-motivated, and self-reflective end of the general population [49], since those people are more willing to take the initiative to pursue online learning options and to take a survey reflecting on their own learning. We are likely missing the population of older adults who are taking in-person programming courses or private tutoring but not seeking any help online. But since we do not know of any programming courses or meetups that target older adults, our intuition is that far more people in this age group are currently self-learning online than getting face-to-face formal instruction. In the future, performing a detailed study of in-person courses containing older adult students, as well as a replication targeting people from non-English-speaking countries, would complement the findings from our current study. Finally, this study involved only one version of an online survey; additional iterations of this survey and in-depth interviews with selected respondents could augment our findings.

DEMOGRAPHICS: WHO ARE THESE LEARNERS?

Before presenting our findings on why and how older adults learn computer programming and what frustrations they face, we first describe the demographics of our respondents to provide context about where these perspectives are coming from.

Figure 1 shows that respondents varied in age from 60 to 85, with fewer responses from older people (median age=65.5 years). 35% of respondents (177 of 504) reported that they were working, 15% were semi-retired, and 46% were retired. The remaining 4% chose "other" (not shown in Figure 1 due to space reasons). Of those, 12 people identified as "unemployed" (and actively searching for a new job), 2 as "homemaker," 2 as "disabled," and 2 left it blank. Figure 1 also shows that older people were less likely to still be working.

1. Managers	8%
1.1. Chief executives & senior managers	1%
1.2. Administrative & commercial managers	2%
1.3. Product/manufacturing/technology managers	5%
2. Professionals	68%
2.1. Scientists & (non-software) engineers	18%
2.2. Health professionals (e.g., doctors, dentists)	6%
2.3. Teaching professionals (e.g., K-12, college)	18%
2.4. Business & administration professionals	6%
2.5. Software developers	12%
2.6. Legal, social, and cultural professionals	8%
3. Technicians and Associates	15%
3.1. Science & engineering technicians	3.6%
3.2. Healthcare technicians	2%
3.3. Business associates (e.g., loan officer)	0.8%
3.4. Legal, social, and cultural associates	0.6%
3.5. I.T. and software technicians, tech support	8%
4. Clerical Support Workers	1%
5. Services and Sales Workers	2%
6. Agricultural, Forestry, and Fishery Workers	0.4%
7. Craft and Trade Workers	4%
8. Plant and Machine Operators (e.g., bus driver)	0.8%
9. Elementary Occupations (e.g., janitor)	0.2%
0. Armed Forces	0.4%

Table 1. Percent of respondents (N=476) whose self-identified current or former occupations fit into each top-level category of the 2008 International Standard Classification of Occupations (ISCO-08). For categories 1, 2, and 3, we further classified using ISCO-08 sub-categories.

84% of respondents identified as male, and 15% as female. Of the remaining 1%, 2 people identified as other, and 4 declined to state gender. Although we would have preferred a more balanced gender representation, these proportions are consistent with what prior surveys of online learning and MOOCs have found [7, 18, 29] – that men are now overrepresented in the population of people learning STEM topics online.

Respondents came from 52 countries, although there was a heavy skew toward North America and Europe. The four countries with the most respondents were the U.S. (55%), U.K. (7%), Canada (5%), and Australia (4%). Within the U.S., respondents came from 41 out of the 50 states.

Respondents also came from diverse kinds of professions. Table 1 shows the percentage in each category of the 2008 International Standard Classification of Occupations (ISCO-08) [47], a taxonomy used by the United Nations. People from all ten top-level categories (from Managers to Armed Forces) responded to our survey, although the majority were Managers, Professionals, and Technicians/Associates. For those three categories, we further classified respondent occupations into ISCO-08 sub-categories and found that the most common roles were scientists/engineers (18%), teachers (18%), and software developers (12%). Note that most teachers were not computer science teachers; they were often STEM teachers who wanted to learn programming either as enrichment or to complement their own subject teaching.

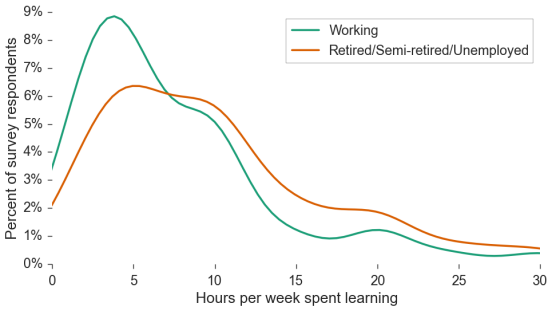

Figure 2. Kernel density estimation curves for how many hours per week respondents said they devoted to learning programming (N=443).

TIME DEVOTED TO LEARNING PROGRAMMING

How long have people been trying to learn programming, and how many hours per week do they devote to it?

We grouped the 469 responses to *"How long have you been trying to learn programming?"* into five buckets: 13% have been trying for less than one month, 19% for less than one year, 17% for 1–3 years, 8% for 4–10 years, and 43% for more than 10 years. In general, most respondents were not new to programming. But although 43% reported greater than 10 years, that does not necessarily mean that they had been *continually* learning for all these years. One common sentiment we noticed was people reporting that they first tried learning several decades ago when they were younger but stopped due to lack of time; and now they are attempting to re-learn since they have more free time in retirement.

Figure 2 shows smoothed kernel density estimation (KDE) curves that estimate the distribution of how many hours our respondents (N=443) reported that they spent each week on learning programming. (If someone reported a range – e.g., "2 to 4 hours" – their vote gets scaled down and evenly distributed over that time range.) Unsurprisingly, working people spent less time each week (median=5 hours) than retired/semi-retired/unemployed people (median=10 hours).

WHY ARE OLDER ADULTS LEARNING PROGRAMMING?

Table 2 summarizes the 494 open-ended responses to the question: *"Why are you learning computer programming? (e.g., what is your motivation for learning?)"* Total percentages add up to more than 100% since 13% of people wrote responses that we classified into more than one category.

For the remainder of the reported results, we use excerpts of quotes from respondents to show representative examples of emergent themes from each coded category in Tables 2, 3, and 4. Also, not all findings apply exclusively to older adults; the most age-relevant findings are marked with ⋆ in those tables.

Age-Related Motivations

The two most common motivations for learning programming were directly age-related (marked with ⋆ in Table 2).

22% of respondents wanted to make up for missed opportunities to learn programming during their youth. A common respondent archetype we observed was a STEM professional (e.g., scientist, engineer, technical manager) who learned a bit of programming decades ago in college but did not get a chance to do much programming throughout their career. Now that they are retired or semi-retired, they have much more free time to devote to learning this topic that they have always wanted to learn since they were young. For example, a 67-year-old retired CIO (Chief Information Officer) wrote:

> *"My motivation is two fold. First, I did a little programming in school when I was in school (PL1, Cobol), and when I first started working. However, I got "kicked upstairs" [into management] quite quickly, and was never able to program professionally. I love computer applications and what they can do, and I love great engineering talent. I always wanted to be able to create programs but between work and family, never took the time. Now that I am retired, I am trying to fulfill the dream and learn. Secondarily, you need passion and focus when your retire and learning to program provides me that."*

A 60-year-old metal fabrication technician wrote about how programming is much more motivating and accessible to newcomers today than when he was young:

> *"I liked programming as a student. However, at the time the neon lit cavernous rooms with rows upon rows of green blinking terminals freed up for homework sometimes after midnight had the same appeal to me as the coal mines. Not only have the conditions changed a lot since then. What caught my attention was the capabilities. Self-driving car. Not that I like cars but the possibility of making it autonomous. The AI allowing the routines to self improve. How fascinating. How can you not be motivated?"*

19% of respondents wanted to use programming to keep challenging their brains while in retirement. For instance, a 62-year old recently-retired surgeon wrote, *"The only way to keep young in mind and body is to learn forever and excersize [sic]."* A 73-year-old retired computer repair technician wrote, *"I never had to learn programming to do my job but I was always interested in it. The medical experts say that learning something new will keep a mind sharper longer."* A 68-year-old retired county planner mentioned how he preferred programming over playing "brain training games" such as Lumosity [2] that are now popular with older adults: *"Also*

⋆Making up for missed opportunities during youth	22%
⋆Keeping one's brain challenged, fresh, and sharp	19%
To implement a specific hobby project idea	19%
For fun and entertainment	15%
Continuing education for one's job duties	14%
Vague general interest (e.g., like to learn new things)	10%
To improve one's future job prospects	9%
To be able to teach others	8%
Computational literacy; understand modern computing	8%
⋆Directly motivated by children/grandchildren/relatives	5%

Table 2. Kinds of reasons mentioned by respondents (N=494) for why they are learning computer programming. ⋆denotes responses that were often mentioned along with age-related reasons. The total percentage adds up to more than 100% since 13% of respondents listed reasons that fit into more than one category.

feel [programming] is a better mind strengthening tool than Luminosity [sic], etc." Regardless of whether computer programming indeed keeps an aging brain sharp over time (we do not know of any studies), many respondents at least perceived it to do so and thus were motivated by that reason.

Finally, 5% of respondents were directly motivated by their younger relatives, such as children, grandchildren, nephews, and nieces. A 69-year-old architect wrote, *"[I want to learn] for knowledge and to see what its all about. My daughter enrolled in an online course on Coursera and it sounded like fun so I did the same. Have taken several since."* A 63-year-old business executive wrote, *"I like to learn new things. My daughter is very interested in programming so she is the one who teaches me and motivates me."* For some, programming was a shared hobby that could help them connect better with their younger relatives. A 65-year-old retired accountant wanted to use programming to *"get closer with my grandkids."* And a 72-year-old retired woodworker wrote, *"I'm trying to learn programming in order to help my grandson who's currently learning the basics of how to start programming."*

Enrichment-Related Motivations

Another major class of motivations was related to personal enrichment: 19% wanted to learn programming to implement a specific hobby project idea, 15% for fun and entertainment, and 10% out of vague general interest in learning new things. Although these reasons are certainly not age-specific, retired and semi-retired people tend to have more free time to spend on hobbies and enrichment [57].

Some respondents noted that programming is a good hobby because it can be done in the privacy, comfort, and safety of one's own home without requiring many external resources. For instance, a 75-year-old retired real estate broker from Uruguay observed how it can be a pragmatic and sustainable hobby for people in his age group: *"Not many entertainment choices for older people. Not a very good idea to spend too much time on the streets. Strong age discrimination. On the positive side a fantastic fiber glass connection to internet. Spend most time between home and the gym 3 blocks away."*

Some respondents were specifically motivated to create hobby projects in domains of interest to older adults, including digital family genealogy and healthcare analytics. One person wrote, *"I think about tech ideas for older people frequently and think learning not just programming but tech generally provides a good base for moving ideas forward."* And this detailed response by a 64-year-old retired network engineer captured both the keeping-one's-brain-active motivation as well as the desire to create software projects for seniors:

> *"My motivations are twofold. First, by endlessly learning new things, I hope to delay or reduce the effects of senility on my brain. Keep those new synapses forming everyday. Second, to take advantage of data produced by the many health related, sensor based monitors (FitBits, Blood Sugar Monitors, Heart Monitors, Motion Monitors, etc) I want to help myself and other senior citizens maintain an independent living lifestyle that is affordable by the masses not just seven figure 401K owners."*

Job-Related Motivations

The final class of motivations was vocational in nature, similar to what all working people – regardless of age – need to do to keep up-to-date in technology-related jobs. However, the challenges of layoffs, age discrimination, and troubles finding new jobs are more likely to affect older workers [11, 31].

14% wanted to learn programming as continuing education that is directly relevant to their current job. A 68-year-old website designer wrote, *"I wanted a refresher (learned VB about 15 years ago, but have not used it much, especially recently). My job allows/encourages continuing education, and I want to keep challenging myself. And, I have a particular task at work I'd love to be able to automate."*

Another 9% wanted to learn programming to improve their future job prospects, either because they were unemployed, looking to switch careers, or semi-retired and seeking part-time income. A 61-year-old unemployed programmer wrote:

> *"Before being laid off due to a company reorganization, we had started utilizing automated testing. Test automation requires some level of programming knowledge so I enrolled in a introductory Java course which I completed after the lay off. Currently, I'm learning Python to add it to my programming skills arsenal for possible use in some paid and/or volunteer capacity."*

Other older technology-sector employees, such as this 62-year-old tech support worker, had similar stories of layoffs:

> *"My career ended in 2001 during the technology 'crash.' My employer went out of business. I, along with 30 others in the engineering department were laid off. Career level jobs were very scarce. Younger people were cheaper to employ. [...] I am trying to regain some fresh technical competencies to enable transition into a better position [as an employee or independent contractor]"*

8% of respondents wanted to understand what modern computing is about (i.e., computational literacy) because they felt like it could enrich their careers even if they do not need to do any programming themselves. A 69-year-old research scientist wanted to learn *"to be conversant with programmers I work with,"* fitting the persona of Chilana et al.'s *conversational programmer* [12]. A 67-year-old retired business systems analyst wanted to learn *"as a means of keeping my brain active. In addition, given the world today, I believe everyone should know how to program as well as learn how to think and solve problems computationally."* And a 62-year-old CEO mentioned the importance of computational literacy for modern businesspeople and other leaders:

> *"I run a large corporation in Colombia. I strongly believe that if I do not change my analog thinking I will not be able to conduct the business I run in an exponential world. I do not want to be a programmer, I do want to understand the logic of the new world we are living."*

Finally, 8% wanted to learn to teach others. Although some were full-time teachers, others wanted to transition into part-time adjunct teaching as they phased out of their primary careers (either voluntarily or involuntarily due to layoffs).

Massive Open Online Courses (MOOCs)	44%
Blogs, web tutorials, language/library documentation	39%
Paper trade books	28%
Free non-MOOC online courses (e.g., Codecademy)	17%
Electronic trade books	9%
YouTube and other video-based tutorials	8%
Paper textbooks	8%
Paid online courses (e.g., Udemy, Lynda)	6%
Online discussion forums (e.g., Stack Overflow)	3%
Electronic textbooks	3%
In-person formal class, bootcamp, or seminar	3%
In-person interactions with peers, MOOC meetups	2%
Resources at the library	2%

Table 3. The resources that survey respondents are using to learn computer programming (N=435). Total adds up to more than 100% since 26% of respondents listed resources that fit more than one category.

HOW ARE OLDER ADULTS LEARNING PROGRAMMING?

Table 3 summarizes the 435 open-ended responses to the question: *"What resources are you currently using to learn programming?"* 26% wrote responses that we classified into more than one category, so totals add up to more than 100%.

The majority of learners used free online resources such as MOOCs, blogs, YouTube, and discussion forums. This finding is unsurprising since our survey was deployed to a free educational website that was often linked to by other free resources. Still, it indicates that older adults find and use similar kinds of online resources that learners of all ages use.

67% of respondents took some sort of online course. The most popular (used by 44%) were MOOCs from all three major providers: Coursera, edX, and Udacity. People took introductory programming MOOCs in languages such as Java and Python, as well as domain-specific ones such as data science (e.g., in R and MATLAB) and web development (JavaScript). In addition, 17% took free non-MOOC online courses such as those from Codecademy and Khan Academy, while 6% paid for online courses such as those from Udemy and Lynda.

For online content that was not organized into courses, 39% of respondents used free web-based materials such as blogs, tutorials, and documentation for programming languages or libraries; 8% used YouTube and other video-based tutorials; and 3% used discussion forums such as Stack Overflow.

Almost half of all respondents (48%) used some kind of book. Trade books (often written by industry practitioners) were more popular than textbooks (often written by academics). Far more people used traditional paper versions of books than electronic versions. And 2% of respondents mentioned checking out those books at their local library.

Only 5% of respondents mentioned in-person learning, either in formal classes (3%) or informally from peers (2%). Although this is a consequence of deploying our survey to a website, the small percentage of learners who have had in-person pedagogical interactions is still noteworthy. One possible explanation is that older adults (unlike, say, college students) have relatively fewer opportunities to take classes, to attend social meetups (e.g., hackathons, on-campus career fairs, tech talks), or to learn informally from peers, but they can still find online resources just as easily as people of all ages can. We know of no prior research or educational initiatives on formal programming classes that target older adults, so our hunch is that online self-directed learning avenues are currently the most widely-used amongst this age group.

Bad pedagogy (e.g., jargon, lack of scaffolding)	21%
⋆Cognitive impairments (e.g., forgetfulness)	12%
Syntax errors	12%
⋆Lack of free time (e.g., job or home caretaker duties)	11%
⋆No human contact with tutors or peers	10%
⋆Software choices are too vast and fast-changing	8%
Software installation and configuration problems	8%
Programming language-specific features	7%
Run-time errors	7%
Poor online documentation	6%
⋆Affective barriers (e.g., fear, self-doubt)	6%
Programming lessons lack real-world relevance	6%
Lack of algorithmic thinking skills	6%
Lack of formal mathematical background	3%
⋆Lack of job opportunities for older people	1%

Table 4. Kinds of learning frustrations mentioned by respondents (N=414). ⋆denotes responses that were often mentioned along with age-related reasons. Total adds up to more than 100% since 15% of respondents listed frustrations that fit into more than one category.

WHAT FRUSTRATIONS DO LEARNERS EXPERIENCE?

Table 4 summarizes the 414 responses to the question: *"What has been the most frustrating thing so far about learning programming?"* 15% wrote responses that we classified into more than one category, so totals add up to more than 100%.

Age-Related Frustrations

The following frustrations (marked with ⋆ in Table 4) are not exclusive to older adults, but our survey respondents frequently mentioned age as a contributing factor in responses.

12% of respondents mentioned frustrating cognitive limitations such as bad memory, forgetfulness, slow speed of comprehension, lack of mental clarity, and difficulty in concentrating. A 68-year-old retired programmer summarized:

> *"I think I know what the principal difficulties are: a.) ingrained habits [...] b.) age: Maybe not everybody does, but I forget. This is the basic reason that I retired, when I noticed that jobs that used to take 10 minutes now take an hour. Again, I may be overgeneralizing, but I think that programming is done by keeping actively in mind everything related to the problem and the code, which takes some warming up time, and is easily broken by interruption, and cannot work without memory."*

A 62-year-old electrical engineer humorously reported his frustrations when learning programming, in spite of his extensive technical background and career spent in a related field:

> *"Given that I was a VERY early adopter of microprocessor / microcontroller technology, I have NO fear of the equipment or the concepts. But things that were almost "automatic" a few years back seem to take a*

lot more time and effort to digest and store than they used to. Early onset Alzheimer's? Probably not. ACS? (Advanced curmudgeon syndrome) - Probably some of that. There is definitely something going on with the malleability of my neurons as I get older - LOL"

A few respondents, including this 71-year-old retired I.T. technician, mentioned health problems contributing to impaired cognitive abilities:

"I have to take medications for Back Pain and Nerve-memory that causes pain and cramping in my right leg. The medications interfere with my short-term memory, which in turn interferes with committing things to long-term memory. These are the same symptoms as premature Alzheimer's Disease..."

Even though we initially thought that older adults would have more free time, 11% of respondents cited lack of free time as a frustration. While the majority of those people were still working, even retirees mentioned lack of time since they had other life duties to take care of. For instance, a 69-year-old retired electronic power systems manager mentioned his caretaker role at home: *"I am a full time (24/7) carer for my wife, so life is full of interruptions: it is difficult to concentrate."*

10% mentioned lack of human contact with tutors or peers as frustrating due to feelings of social isolation, lack of emotional support, and lack of real-time technical guidance. A 67-year-old retired business analyst said, *"I don't personally know anyone (in or out of my age group) who has the same level of interest that I have regarding programming or computing in general. As a result I'm more or less on my own."*

Lack of human contact is a challenge that all online learners face, but this 60-year-old I.T. salesperson emphasized that it may be an especially salient problem for older adults:

"This is by far the most important question you are asking, because i have to imagine that many 60+ people are facing this same challenge as I. Programming is not something that should be done as a lone person even with the resources that i mention above. Learning a programming language is much like learning a foreign language. Yes, one could teach them selves French by reading a bunch of books, but it sure would be faster if one practiced speaking with French people. Programming languages are the same. You learn a lot faster in collaboration with others that are learning at the same time. [...] Net of it all: efficient collaboration is the key for the 60+ crowd to learn programming."

8% of respondents were frustrated by the sheer multitude of choices in software packages and the rapid rate of change of what technologies are currently in vogue. One wrote, *"Plus, the field is always evolving, so you may be learning stuff that is on its way to extinction."* Although this challenge is certainly not limited to this age group, some respondents expressed how they pined for the "good ole' days" when technology choices were more limited and slower-changing. An engineer who has been programming professionally since 1983 wrote: *"I am basically an assembler programmer. There are about 100 instructions, you know precisely, to the bit level, what each one does, and with a good debugger you can actually see the bit changes, with complete control. With high level languages, all that goes to hell. Java has some 3000 built in classes, each of which wants strictly typed arguments, and has a name longer than Mahabharata."*

6% of respondents cited affective barriers such as fear, self-doubt, lack of self-efficacy, and being afraid of falling behind when taking MOOCs. Although these issues affect novices of all ages, some older adults are more anxious about learning to use computers [57], so those feelings may also transfer to learning programming. Finally, a few unemployed respondents (1%) were frustrated by their perceptions of age discrimination: e.g., *"No matter what I learn, I know that at my age, no one will hire me."* Another wrote: *"I'm not getting a job! No one wants to hire people over 40 for programming."*

Pedagogy-Related Frustrations

The most common kind of frustration (mentioned by 21% of respondents) was regarding bad pedagogy. Respondents complained about a lack of instructional scaffolding, sudden spikes in difficulty levels of lessons, not enough example problems, and too many low-level jargon-filled technical explanations that focus on the "how" and not the "why."

A 74-year-old retired physician observed that many free programming tutorials are created by practitioners who are not trained as teachers: *"Most [tutorials] are offered by people who must know how to program but don't seem to have much training in teaching."* And a 71-year-old retired electrical engineer and sales manager used his age to humorously lament the lack of scaffolding: *"Noticed both John Hopkins and Rice [MOOCs] have a tendency to throw us in the water and yell swim!! I don't have the time to waste floundering around. Males in my family die before their late 60's. As the exception in mine, I'm well aware I'm on borrowed time."*

6% of respondents complained about a lack of real-world relevance in what they were learning. They wanted to see more practical problems being solved with code rather than lessons that focus exclusively on abstract concepts.

Finally, 6% were frustrated by poor-quality and poorly-organized online documentation, too much "marketing-speak" in programming materials (both open-source and corporate), and frequent errors and typos in learning resources.

In theory, these pedagogy-related frustrations could be eliminated by taking a well-designed, carefully-vetted, high-quality course. However, in reality people tend to forage online for a variety of piecemeal resources (see Table 3), which are of varying quality. Also, the abundance of free resources surfaced by web searches makes it hard for a novice to hone in on the best ones for their own learning needs.

Technology-Related Frustrations

The final class of frustrations are those that all novices commonly face. Debugging syntax (12%) and run-time errors (7%) is perennially painful for novices [3]. Technical features of specific programming languages accounted for 7% of frustrations. e.g.,: *"Pointers in C is the most frustrating one*

that took me nearly 5 months to understand it properly." Software installation and configuration frustrated 8% of respondents. e.g.,: *"Building and maintaining attendant infrastructure requirements, from operating systems (Linux), utilities (Git) and languages themselves, and their maintenance."*

6% of respondents were frustrated by their inability to think algorithmically to conceptualize, plan, and implement solutions to programming problems. One wrote, *"The language is easy. Solving problems in discrete steps is hard. Developing the heuristic, converting that into an algorithm, and then creating a program from it."* Finally, 3% explicitly called out a lack of mathematical background as a hindrance. e.g.,: *"I didn't take enough math earlier in life to more easily grasp the computer science concepts. So I kick myself over that."*

In theory, these frustrations could be alleviated by having more real-time help from tutors or peers, but another common frustration (from 10% of respondents) was that such human contact was not readily available to people learning online.

DISCUSSION

We reflect on our findings in three ways: We first relate what we discovered to the findings of other HCI studies on aging and technology, then propose a learner-centered design [25] of techniques and tools to motivate more older adults to learn programming, and conclude by speculating on what the future may look like if more older adults learn programming.

Relationship to HCI Research on Aging and Technology

Since programming is a kind of computer-based technology – albeit a sophisticated one – we expect that what motivates and frustrates older adults in learning about this technology will align with how they perceive other digital technologies.

Reflecting on the motivations for learning programming in Table 2, a top motivator for our respondents (19%) was the desire to keep their brains challenged, fresh, and sharp. This mirrors similar motivations for performing writing-related creative work such as blogging [9] and posting to online discussion forums [43], where older adults wanted to feel a sense of meaningful, purposeful engagement rather than passive entertainment such as watching TV. Also, Brewer et al. found that mental stimulation was the highest-rated motivator for older adults to perform microtask crowd work on Amazon Mechanical Turk, even surpassing the obvious benefit of making money from that work [8]. Their study participants also wanted to work on more challenging and sophisticated tasks that could possibly serve as cognitive conditioning.

The desire to create hobby projects was another big motivator (19% of respondents). Some mentioned wanting to volunteer to build software or websites for local organizations or to teach programming on a voluntary basis in local schools. Prior studies have found many benefits of volunteering on the well-being of older adults, including improved health, reduced feelings of social isolation, and longer lifespans [40].

5% of respondents were also motivated by younger family members, both as a way to understand what technologies the younger generations are now using and to foster connections with those family members. Some also wanted to learn to teach programming to their grandchildren. Prior work has shown how older adults value using technology to keep in touch with family, and how they also prefer more substantive long-form communication rather than superficial interactions on social media [30, 38]. Sharing a common learning goal and taking online courses together can be one substantive channel for staying connected with family members.

Job-related motivations such as continuing education and improving one's future job prospects were not often mentioned by prior studies, since those focused more on recreational rather than vocational activities. One exception is Brewer et al.'s study of crowd work, where some older adults were motivated by making side income from performing Mechanical Turk microtasks [8] Computer programming is unique amongst technologies used by older adults because it can be useful both for recreation and to make a sustainable living.

For frustrations reported by respondents (Table 4), concerns about age-related cognitive impairments such as forgetfulness have been well-documented [52]. Some retired people also cited lack of free time due to their role as home caretakers for family members, an activity that has been shown to be both time consuming and emotionally demanding [50]. Respondents were also frustrated by lack of human contact with like-minded peers to learn together, which is consistent with findings on greater social isolation experienced by older adults as a group [45, 56]. Finally, although all novices face syntax and run-time errors due to the exact precision required by text-based programming languages (i.e., any minor typo can lead to frustrating errors), prior studies of typing mistakes amongst older adults [27, 42] could point to why these sorts of errors may be more prevalent amongst this population. An open question here is how to design better programming interfaces to account for these kinds of common input slips faced by older adults and others with motor impairments.

Learner-Centered Design for Older Adult Programmers

Our survey respondents are mostly using general-purpose online learning resources such as MOOCs and technical blogs (see Table 3). Although many age-targeted learn-to-code resources exist for children and college students, along with some for early- to mid-career professionals (see Related Work for details), to our knowledge, none are tailored to older adults. What might such curricula and tools look like if they were designed with this specific learner population in mind?

To generate some initial ideas, we turn to the *learner-centered design* process proposed by Guzdial et al. [25], which posits that we must use the learner's own motivations and frustrations as the starting points for design. In a similar spirit as user-centered design, Guzdial advocates performing targeted studies on specific learner populations and then using empirical findings – rather than instructor intuitions – as the basis for formulating design suggestions. Since this specific population has not yet been studied in prior work, we want to use our study as the basis for grounding design ideas using Guzdial's template. That said, we acknowledge that some of these ideas make sense intuitively even without a study and can apply to other age groups as well. We now present ideas based on how our findings relate to the seven parts of this template:

1. Understand where learners are starting from and what they want to do: Although our respondents are a diverse group – spanning 52 countries, aged 60 to 85, and in dozens of occupations (Table 1) – we found that they shared a sense of intellectual curiosity and desires for mental stimulation and meaningful engagement (Table 2). Based on tone in written responses, some are proud of being older adults who are either tech-literate or have a strong desire to be, so learning materials need to respect that identity. Thus, it can be a good idea for resources to explicitly advertise that they are targeted to this age group so that learners feel like it is meant for people like them, but also to make sure not to appear patronizing. Another idea is to re-frame programming curricula as brain training games [2] that are now popular with older adults.

2. Understand where learners have trouble: Our respondents' most common frustrations were about bad pedagogy and perceived cognitive impairments (Table 4). One solution to both of these issues is to design learning materials that emphasize scaffolding, repetition, and abundant examples to account for potential cognitive impairments commonly faced by this age group, but taking special care not to appear patronizing (see above). Designing custom programming environments and tools for older learners could also alleviate the effect of cognitive impairments and reduce input slips [27] that lead to frustrating syntax and run-time errors. Finally, another common frustration was lack of human contact with tutors or peers, which echoes more general feelings of social isolation amongst this age group [45, 56]. This points toward the importance of organizing in-person courses or online video chat-based workshops targeted to this age group. Note that these ideas could be examples of *universal design* – focusing on improving the learning experience of older adults may result in designs that benefit learners of all ages.

3. Scaffolding: Our respondents were often overwhelmed by the multitude of choices available in modern software development, and the ensuing installation and configuration problems (Table 4). One critical form of scaffolding is to take away the complexities of choice by providing a uniform yet full-featured environment with enough built-in libraries to create authentic, non-toy projects. Starting points include IDEs such as DrRacket [19] and JES for Media Computation [26], and packages such as Anaconda [4] for Python.

4. Use terminology that learners understand: Many of our respondents are scientists, engineers, and other STEM professionals who were first exposed to programming during their youth using technologies from that era such as Fortran, COBOL, assembly language, and digital electronics circuitry. One way to design better curricula for this audience is to leverage their prior knowledge of terminology from these older technologies that they are familiar with and then make bridging analogies to similar concepts in newer platforms (e.g., mobile, web, data science) that they are now learning.

5. Contextualization: Learning resources are more compelling when put into the context of domains that learners personally care about. One way to contextualize programming education for this audience is to enable learners to develop projects of relevance to older adults, which some respondents were already doing on their own. For instance, much like how tools enable children to tell fictional stories by writing code [23, 33], a coding environment for older adults could facilitate storytelling using digital media from their own life and family histories, which is an activity of high interest to this age group [60]. Other project-based learning ideas include writing software to assist caregivers [5, 50], to mediate online interactions on social media [22, 30, 44], and to aggregate and organize information from healthcare websites [37].

6. Learners change as they acquire expertise: As our respondents become more experienced, they may transition from identifying as learners to wanting to be treated as community volunteers, programming teachers, open-source contributors, or in some cases, professional software developers. More advanced learning resources should be catered to how people self-identify once they are no longer novices. (This part is not specific to older adults, but is included for completeness.)

7. Acknowledge differences among learners: Finally, older adults are not a homogeneous population by any means [57], so it is important not to overgeneralize design ideas or to implicitly claim that there is some sort of optimal design for everyone in this age range. For instance, a 60-year-old working engineer is likely to have very different motivations and constraints than an 85-year-old retiree living in a long-term care facility. Thus, these ideas should be viewed only as a starting point.

CONCLUSION

Using survey responses from 504 learners aged 60 to 85, we discovered a variety of motivations and frustrations from older adults learning computer programming. This study is only the first step, though, since we observed an elite population of self-selected "early adopters" – mostly people from technical backgrounds who took the initiative to learn online. What might the future look like if we make explicit efforts to broaden the population of older adults who learn to code? First, more older adults doing programming means that they have an additional tool to improve quality of life as they age: This activity can be done in the comfort of one's own home, provide engaging mental stimulation, give meaning by empowering people to create software that is of value to their community, and foster social connections to both family and peers, all of which are important to this population. Programming skills also enable older adults to maintain gainful part- and full-time employment with flexible jobs where they can work from home, which is significant as people's lifespans continue to increase. Another benefit of greater participation of older adults in the software industry is that there may be more emphasis on building products for this fast-growing population, as opposed to the currently youth-centric product focus of the tech industry [11, 31, 59]. Lastly, this population can potentially become a pipeline of teachers necessary for educating subsequent generations and alleviate the current shortage of qualified computing instructors [16, 17]. These efforts all point toward a future where the spirit of *Computer Science For All* [46] means full inclusion of all age groups.

Acknowledgments: Thanks to Steven Dow, Bill Griswold, Scott Klemmer, Leo Porter, and Beth Simon for feedback.

REFERENCES

1. 2016. About Scratch. https://scratch.mit.edu/about. (2016). Accessed: 2016-09-19.
2. 2016. Lumosity: Brain Games & Brain Training. https://www.lumosity.com/. (2016). Accessed: 2016-09-19.
3. Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 522–527. DOI: http://dx.doi.org/10.1145/2676723.2677258
4. Continuum Analytics. 2016. Anaconda Distribution Open Data Science Core. https://docs.continuum.io/anaconda/. (2016). Accessed: 2016-09-19.
5. Ingrid Arreola, Zan Morris, Matthew Francisco, Kay Connelly, Kelly Caine, and Ginger White. 2014. From Checking on to Checking in: Designing for Low Socio-economic Status Older Adults. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1933–1936. DOI: http://dx.doi.org/10.1145/2556288.2557084
6. Andrew Begel and Beth Simon. 2008. Struggles of New College Graduates in Their First Software Development Job. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, New York, NY, USA, 226–230. DOI: http://dx.doi.org/10.1145/1352135.1352218
7. Lori Breslow, David E Pritchard, Jennifer DeBoer, Glenda S Stump, Andrew D Ho, and Daniel T Seaton. 2013. Studying learning in the worldwide classroom: Research into edX's first MOOC. *Research & Practice in Assessment* 8 (2013).
8. Robin Brewer, Meredith Ringel Morris, and Anne Marie Piper. 2016. "Why Would Anybody Do This?": Understanding Older Adults' Motivations and Challenges in Crowd Work. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2246–2257. DOI: http://dx.doi.org/10.1145/2858036.2858198
9. Robin Brewer and Anne Marie Piper. 2016. "Tell It Like It Really Is": A Case of Online Content Creation and Sharing Among Older Adult Bloggers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 5529–5542. DOI: http://dx.doi.org/10.1145/2858036.2858379
10. Bo Brinkman and Amanda Diekman. 2016. Applying the Communal Goal Congruity Perspective to Enhance Diversity and Inclusion in Undergraduate Computing Degrees. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 102–107. DOI: http://dx.doi.org/10.1145/2839509.2844562
11. Kevin Casey. 2013. Are You Too Old For IT? http://www.informationweek.com/strategic-cio/team-building-and-staffing/are-you-too-old-for-it/d/d-id/1006268. (Nov 2013). Accessed: 2016-09-19.
12. Parmit K. Chilana, Rishabh Singh, and Philip J. Guo. 2016. Understanding Conversational Programmers: A Perspective from the Software Industry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1462–1472. DOI: http://dx.doi.org/10.1145/2858036.2858323
13. Juliet M. Corbin and Anselm L. Strauss. 2008. *Basics of qualitative research: techniques and procedures for developing grounded theory*. SAGE Publications, Inc.
14. Brian Dorn and Mark Guzdial. 2010. Learning on the Job: Characterizing the Programming Knowledge and Learning Strategies of Web Designers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 703–712. DOI: http://dx.doi.org/10.1145/1753326.1753430
15. Stuart Dredge. 2014. Coding at school: a parent's guide to England's new computing curriculum. The Guardian: https://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming. (Sep 2014). Accessed: 2016-09-19.
16. Barbara Ericson and Mark Guzdial. 2014. Measuring Demographics and Performance in Computer Science Education at a Nationwide Scale Using AP CS Data. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 217–222. DOI: http://dx.doi.org/10.1145/2538862.2538918
17. Barbara J. Ericson, Kantwon Rogers, Miranda Parker, Briana Morrison, and Mark Guzdial. 2016. Identifying Design Principles for CS Teacher Ebooks Through Design-Based Research. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 191–200. DOI: http://dx.doi.org/10.1145/2960310.2960335
18. Deborah A. Fields, Michael Giang, and Yasmin Kafai. 2014. Programming in the Wild: Trends in Youth Computational Participation in the Online Scratch Community. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*. ACM, New York, NY, USA, 2–11. DOI: http://dx.doi.org/10.1145/2670757.2670768

19. Robert Bruce Findler. 2013. DrRacket: The Racket Programming Environment. http://mirror.racket-lang.org/releases/6.4/pdf-doc/drracket.pdf. (2013). Accessed: 2016-09-19.
20. Louise P. Flannery, Brian Silverman, Elizabeth R. Kazakoff, Marina Umaschi Bers, Paula Bontá, and Mitchel Resnick. 2013. Designing ScratchJr: Support for Early Childhood Learning Through Computer Programming. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*. ACM, New York, NY, USA, 1–10. DOI: http://dx.doi.org/10.1145/2485760.2485785
21. National Science Foundation. 2016. Broadening Participation in Computing (BPC). http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=13510&org=CISE&from=fund. (2016). Accessed: 2016-09-19.
22. Lorna Gibson, Wendy Moncur, Paula Forbes, John Arnott, Christopher Martin, and Amritpal S. Bhachu. 2010. Designing Social Networking Sites for Older Adults. In *Proceedings of the 24th BCS Interaction Specialist Group Conference (BCS '10)*. British Computer Society, Swinton, UK, UK, 186–194. http://dl.acm.org/citation.cfm?id=2146303.2146331
23. Paul A. Gross, Micah S. Herstand, Jordana W. Hodges, and Caitlin L. Kelleher. 2010. A Code Reuse Interface for Non-programmer Middle School Students. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*. ACM, New York, NY, USA, 219–228. DOI: http://dx.doi.org/10.1145/1719970.1720001
24. Philip J. Guo. 2013. Online Python Tutor: Embeddable Web-based Program Visualization for CS Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 579–584. DOI: http://dx.doi.org/10.1145/2445196.2445368
25. Mark Guzdial. 2015. Learner-Centered Design of Computing Education: Research on Computing for Everyone. *Synthesis Lectures on Human-Centered Informatics* 8, 6 (2015), 1–165.
26. Mark Guzdial and Andrea Forte. 2005. Design Process for a Non-majors Computing Course. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, USA, 361–365. DOI: http://dx.doi.org/10.1145/1047344.1047468
27. Toshiyuki Hagiya, Toshiharu Horiuchi, and Tomonori Yazaki. 2016. Typing Tutor: Individualized Tutoring in Text Entry for Older Adults Based on Input Stumble Detection. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 733–744. DOI: http://dx.doi.org/10.1145/2858036.2858455
28. Joseph Henrich, Steven J Heine, and Ara Norenzayan. 2010. The weirdest people in the world? *Behavioral and brain sciences* 33, 2-3 (2010), 61–83.
29. Andrew Dean Ho, Isaac Chuang, Justin Reich, Cody Austun Coleman, Jacob Whitehill, Curtis G Northcutt, Joseph Jay Williams, John D Hansen, Glenn Lopez, and Rebecca Petersen. 2015. HarvardX and MITx: Two years of open online courses fall 2012-summer 2014. *Available at SSRN 2586847* (2015).
30. Alexis Hope, Ted Schwaba, and Anne Marie Piper. 2014. Understanding Digital and Material Social Communications for Older Adults. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3903–3912. DOI: http://dx.doi.org/10.1145/2556288.2557133
31. Carol Hymowitz and Robert Burnson. 2016. It's Tough Being Over 40 in Silicon Valley. Bloomberg Businessweek. (Sep 2016).
32. Jeff Immelt. 2016. Why GE is giving up employee ratings, abandoning annual reviews and rethinking the role of HQ. LinkedIn Pulse interview. (Aug 2016). Accessed: 2016-09-19.
33. Caitlin Kelleher, Randy Pausch, and Sara Kiesler. 2007. Storytelling Alice Motivates Middle School Girls to Learn Computer Programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 1455–1464. DOI: http://dx.doi.org/10.1145/1240624.1240844
34. Eric Klopfer, Hal Scheintaub, Wendy Huang, and Daniel Wendel. 2009. StarLogo TNG. In *Artificial Life Models in Software*. Springer, 151–182.
35. Celine Latulipe, Amy Gatto, Ha T. Nguyen, David P. Miller, Sara A. Quandt, Alain G. Bertoni, Alden Smith, and Thomas A. Arcury. 2015. Design Considerations for Patient Portal Adoption by Low-Income, Older Adults. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3859–3868. DOI: http://dx.doi.org/10.1145/2702123.2702392
36. Rock Leung, Charlotte Tang, Shathel Haddad, Joanna Mcgrenere, Peter Graf, and Vilia Ingriany. 2012. How Older Adults Learn to Use Mobile Devices: Survey and Field Investigations. *ACM Trans. Access. Comput.* 4, 3, Article 11 (Dec. 2012), 33 pages. DOI: http://dx.doi.org/10.1145/2399193.2399195
37. Q. Vera Liao and Wai-Tat Fu. 2014. Age Differences in Credibility Judgments of Online Health Information. *ACM Trans. Comput.-Hum. Interact.* 21, 1, Article 2 (Feb. 2014), 23 pages. DOI: http://dx.doi.org/10.1145/2534410
38. Siân E. Lindley, Richard Harper, and Abigail Sellen. 2009. Desiring to Be in Touch in a Changing Communications Landscape: Attitudes of Older Adults. In *Proceedings of the SIGCHI Conference on Human*

Factors in Computing Systems (CHI '09). ACM, New York, NY, USA, 1693–1702. DOI: http://dx.doi.org/10.1145/1518701.1518962

39. Johanna Meurer, Martin Stein, David Randall, Markus Rohde, and Volker Wulf. 2014. Social Dependency and Mobile Autonomy: Supporting Older Adults' Mobility with Ridesharing ICT. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1923–1932. DOI: http://dx.doi.org/10.1145/2556288.2557300
40. Nancy Morrow-Howell, Jim Hinterlong, Philip A Rozario, and Fengyan Tang. 2003. Effects of volunteering on the well-being of older adults. *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences* 58, 3 (2003), S137–S145.
41. United Nations. 2015. World Population Ageing Report. http://www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2015_Report.pdf. (2015). Accessed: 2016-09-19.
42. Hugo Nicolau and Joaquim Jorge. 2012. Elderly Text-entry Performance on Touchscreens. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12)*. ACM, New York, NY, USA, 127–134. DOI: http://dx.doi.org/10.1145/2384916.2384939
43. Galit Nimrod. 2011. The fun culture in seniors' online communities. *The Gerontologist* 51, 2 (2011), 226–237.
44. Chris Norval, John L. Arnott, and Vicki L. Hanson. 2014. What's on Your Mind?: Investigating Recommendations for Inclusive Social Networking and Older Adults. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3923–3932. DOI: http://dx.doi.org/10.1145/2556288.2556992
45. British Columbia Ministry of Health. 2004. Social Isolation Among Seniors: An Emerging Issue. http://www.health.gov.bc.ca/library/publications/year/2004/Social_Isolation_Among_Seniors.pdf. (Mar 2004). Accessed: 2016-09-19.
46. The White House: Office of the Press Secretary. 2016. FACT SHEET: President Obama Announces Computer Science For All Initiative. (Jan 2016).
47. International Labour Organization. 2008. International Standard Classification of Occupations. http://www.ilo.org/public/english/bureau/stat/isco/isco08/. (2008). Accessed: 2016-09-19.
48. Seymour Papert. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.
49. Annie Murphy Paul. 2014. Bill Gates Is an Autodidact. You're Probably Not. Ed tech promoters need to understand how most of us learn. *Slate* (July 2014).
50. Anne Marie Piper, Raymundo Cornejo, Lisa Hurwitz, and Caitlin Unumb. 2016. Technological Caregiving: Supporting Online Activity for Adults with Cognitive Impairments. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 5311–5323. DOI: http://dx.doi.org/10.1145/2858036.2858260
51. Leo Porter, Dennis Bouvier, Quintin Cutts, Scott Grissom, Cynthia Lee, Robert McCartney, Daniel Zingaro, and Beth Simon. 2016. A Multi-institutional Study of Peer Instruction in Introductory Computing. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 358–363. DOI: http://dx.doi.org/10.1145/2839509.2844642
52. Laura Ramos, Elise van den Hoven, and Laurie Miller. 2016. Designing for the Other 'Hereafter': When Older Adults Remember About Forgetting. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 721–732. DOI: http://dx.doi.org/10.1145/2858036.2858162
53. Yvonne Rogers, Jeni Paay, Margot Brereton, Kate L. Vaisutis, Gary Marsden, and Frank Vetere. 2014. Never Too Old: Engaging Retired People Inventing the Future with MaKey MaKey. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3913–3922. DOI: http://dx.doi.org/10.1145/2556288.2557184
54. Jason Scott. 2014. Aging Australians Balk at World's Oldest Retirement Age. Bloomberg News. (Jun 2014).
55. The Yomiuri Shimbun. 2016. Plan to make programming mandatory at schools a step to foster creativity. http://the-japan-news.com/news/article/0002951918. (May 2016). Accessed: 2016-09-19.
56. Sarah Stevenson. 2014. 20 Facts about Senior Isolation That Will Stun You. http://www.aplaceformom.com/blog/10-17-14-facts-about-senior-isolation/. (Oct 2014). Accessed: 2016-09-19.
57. John Vines, Gary Pritchard, Peter Wright, Patrick Olivier, and Katie Brittain. 2015. An Age-Old Problem: Examining the Discourses of Ageing in HCI and Strategies for Future Research. *ACM Trans. Comput.-Hum. Interact.* 22, 1, Article 2 (Feb. 2015), 27 pages. DOI: http://dx.doi.org/10.1145/2696867
58. Amy Voida, Sheelagh Carpendale, and Saul Greenberg. 2010. The Individual and the Group in Console Gaming. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW '10)*. ACM, New York, NY, USA, 371–380. DOI: http://dx.doi.org/10.1145/1718918.1718983
59. Todd Wasserman. 2014. Old Coders: When Programming Is a Second Career. Mashable Business: http://mashable.com/2014/08/21/programming-as-a-second-career/. (Aug 2014). Accessed: 2016-09-19.

60. Jenny Waycott, Frank Vetere, Sonja Pedell, Lars Kulik, Elizabeth Ozanne, Alan Gruner, and John Downs. 2013. Older Adults As Digital Content Producers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 39–48. DOI: http://dx.doi.org/10.1145/2470654.2470662

61. Uri Wilensky, Corey E. Brady, and Michael S. Horn. 2014. Fostering Computational Literacy in Science Classrooms. *Commun. ACM* 57, 8 (Aug. 2014), 24–28. DOI: http://dx.doi.org/10.1145/2633031

62. David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney. 2011. *App Inventor*. " O'Reilly Media, Inc.".