CRITICAL THINKING

14 week homework

QUESTION:

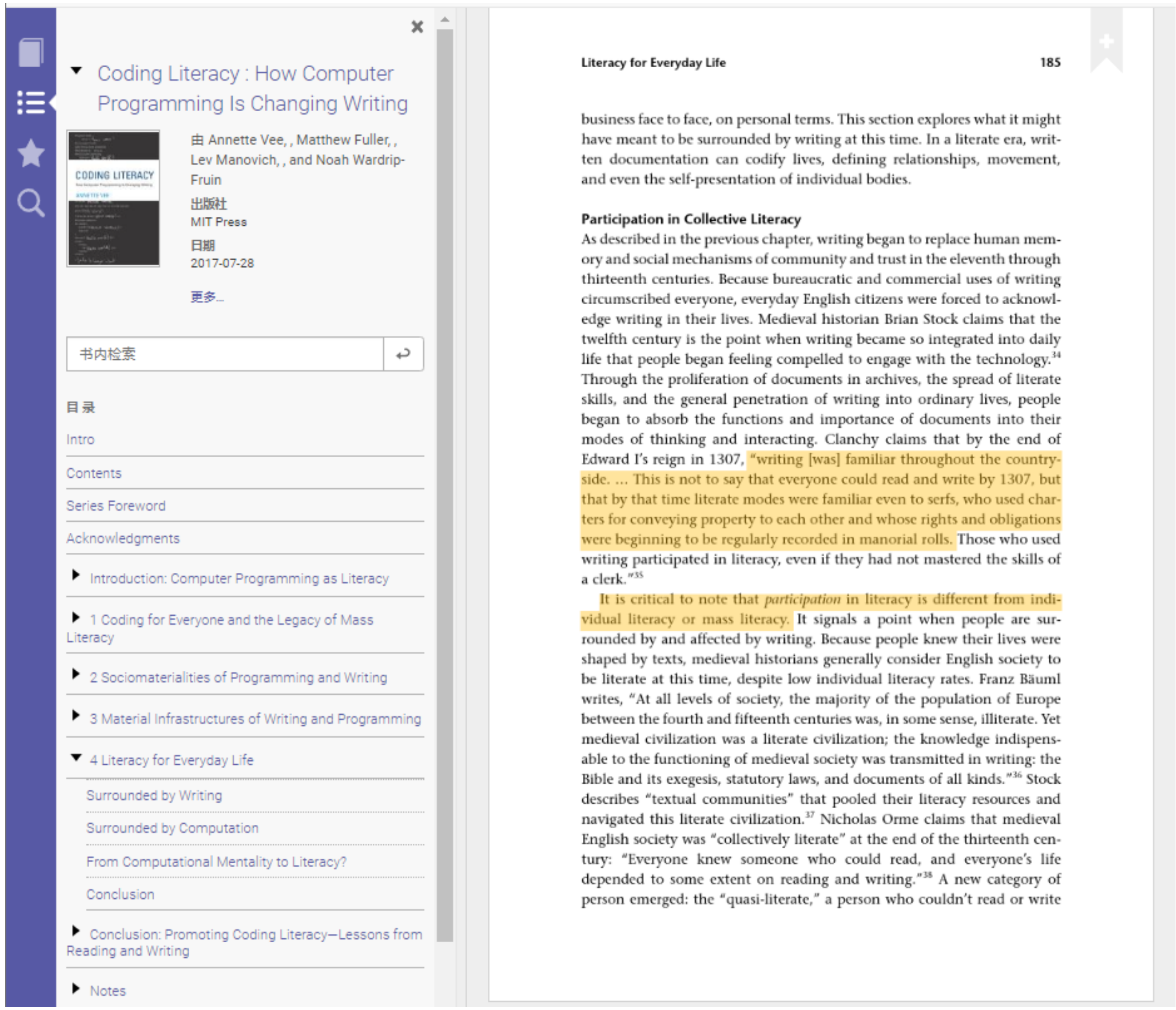
Just simply take a moment to write/reflect about “What is literacy and what does it mean to call computer programming a literacy?”

My Answer:

Reading and writing ability should include reading ability and writing ability. A person is referred to as having strong reading and writing ability, often his understanding is also strong, and at the same time he can also think critically. I think computer programming is similar to reading and writing ability in terms of thinking and logic. We need to analyze codes, think about operations, and finally come up with solutions.

READING TEXTUAL

1.Vee, Annette. Coding Literacy. MIT Press, 2017. (Introduction chapter pp. 1-38)



Coding Literacy : How Computer Programming Is Changing Writing

由 Annette Vee, , Matthew Fuller, , Lev Manovich, , and Noah Wardrip-Fruin

出版社
MIT Press

日期
2017-07-28

更多...

书内检索

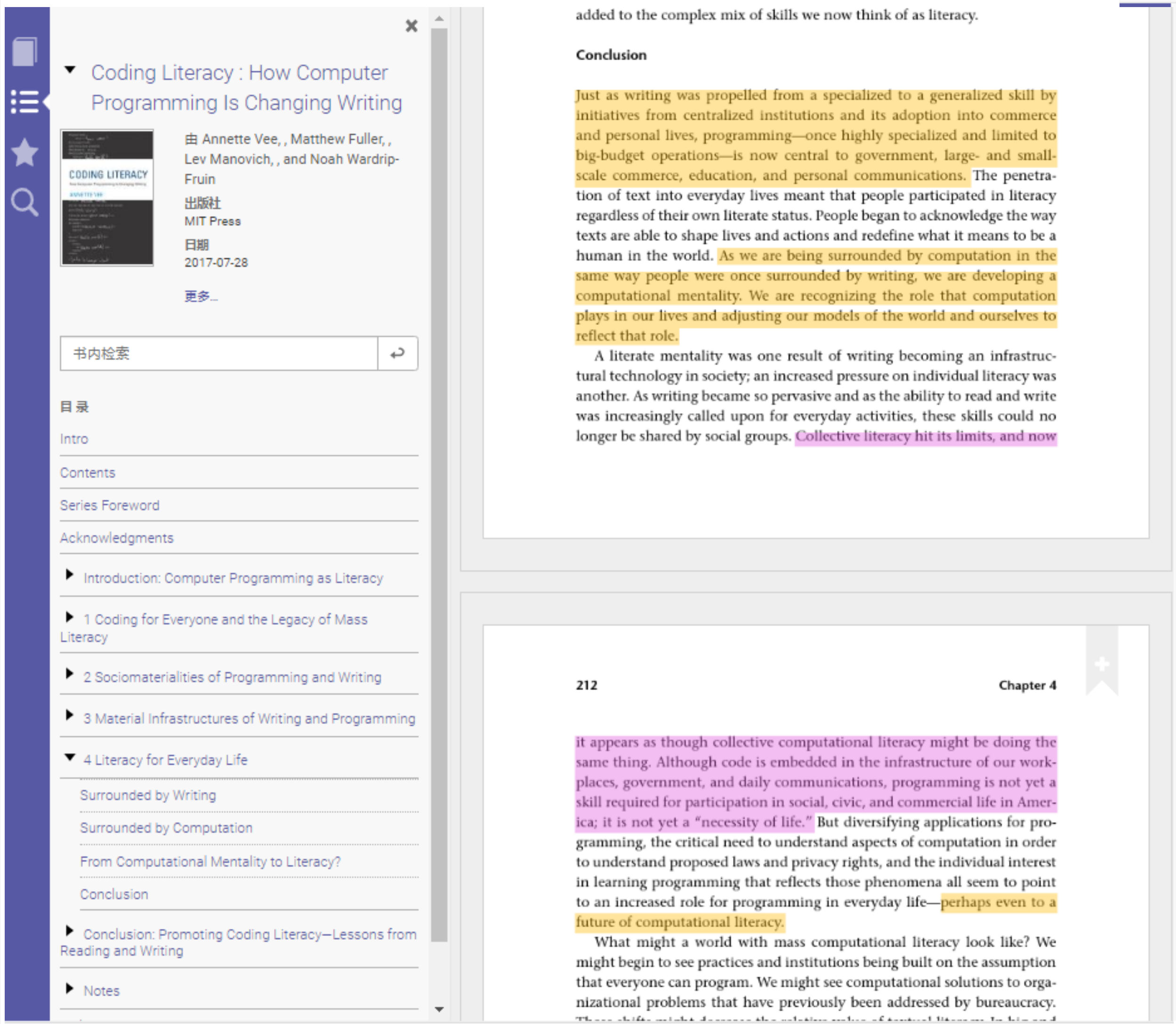
目 录

business face to face, on personal terms. This section explores what it might have meant to be surrounded by writing at this time. In a literate era, written documentation can codify lives, defining relationships, movement, and even the self-presentation of individual bodies.

Participation in Collective Literacy

As described in the previous chapter, writing began to replace human memory and social mechanisms of community and trust in the eleventh through thirteenth centuries. Because bureaucratic and commercial uses of writing circumscribed everyone, everyday English citizens were forced to acknowledge writing in their lives. Medieval historian Brian Stock claims that the twelfth century is the point when writing became so integrated into daily life that people began feeling compelled to engage with the technology.[34] Through the proliferation of documents in archives, the spread of literate skills, and the general penetration of writing into ordinary lives, people began to absorb the functions and importance of documents into their modes of thinking and interacting. Clanchy claims that by the end of Edward I's reign in 1307, "writing [was] familiar throughout the countryside. … This is not to say that everyone could read and write by 1307, but that by that time literate modes were familiar even to serfs, who used charters for conveying property to each other and whose rights and obligations were beginning to be regularly recorded in manorial rolls. Those who used writing participated in literacy, even if they had not mastered the skills of a clerk."[35]

It is critical to note that *participation* in literacy is different from individual literacy or mass literacy. It signals a point when people are surrounded by and affected by writing. Because people knew their lives were shaped by texts, medieval historians generally consider English society to be literate at this time, despite low individual literacy rates. Franz Bäuml writes, "At all levels of society, the majority of the population of Europe between the fourth and fifteenth centuries was, in some sense, illiterate. Yet medieval civilization was a literate civilization; the knowledge indispensable to the functioning of medieval society was transmitted in writing: the Bible and its exegesis, statutory laws, and documents of all kinds."[36] Stock describes "textual communities" that pooled their literacy resources and navigated this literate civilization.[37] Nicholas Orme claims that medieval English society was "collectively literate" at the end of the thirteenth century: "Everyone knew someone who could read, and everyone's life depended to some extent on reading and writing."[38] A new category of person emerged: the "quasi-literate," a person who couldn't read or write



Coding Literacy : How Computer Programming Is Changing Writing

由 Annette Vee, , Matthew Fuller, , Lev Manovich, , and Noah Wardrip-Fruin

出版社
MIT Press

日期
2017-07-28

更多...

书内检索

目录

added to the complex mix of skills we now think of as literacy.

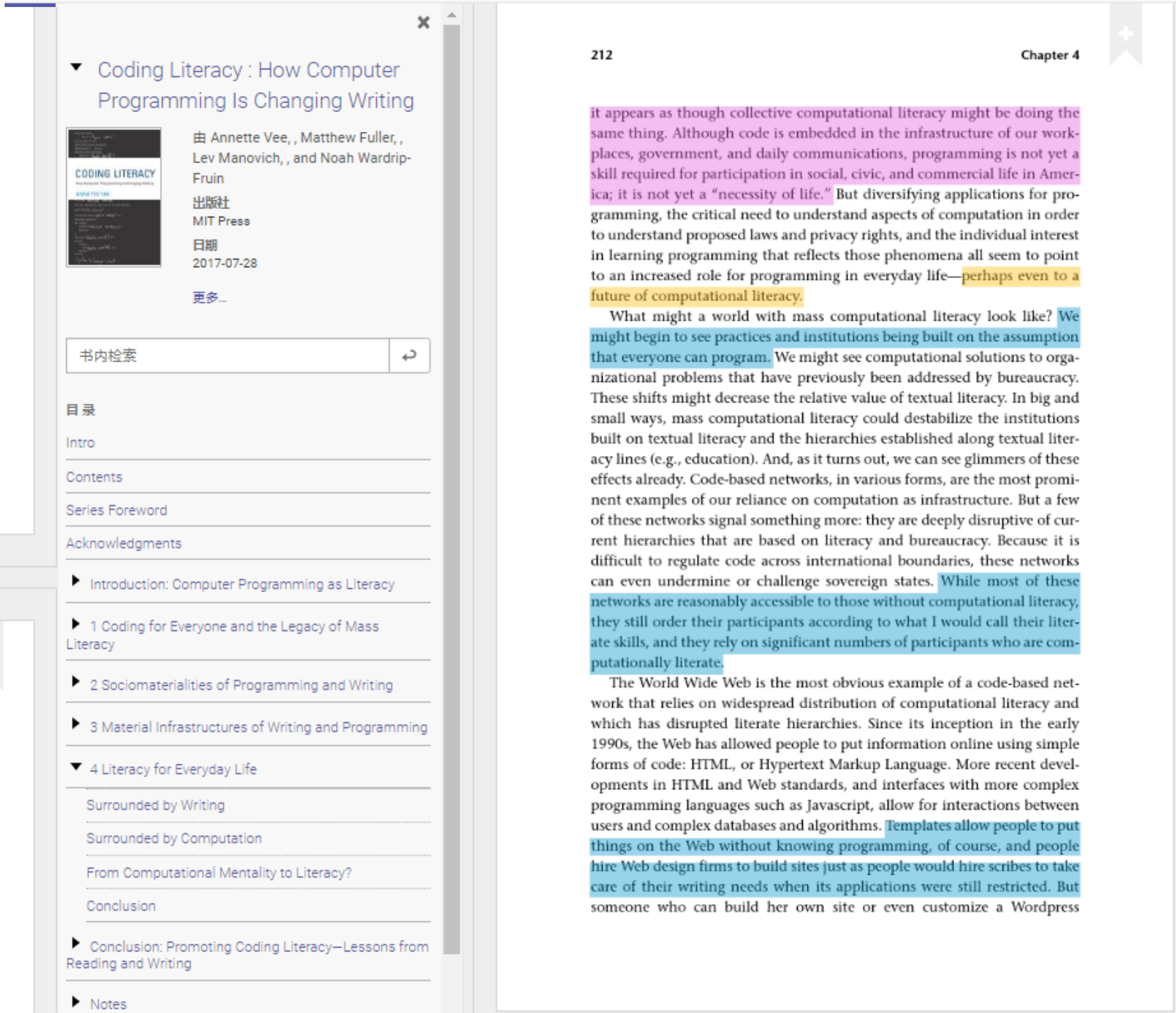
Conclusion

Just as writing was propelled from a specialized to a generalized skill by initiatives from centralized institutions and its adoption into commerce and personal lives, programming—once highly specialized and limited to big-budget operations—is now central to government, large- and small-scale commerce, education, and personal communications. The penetration of text into everyday lives meant that people participated in literacy regardless of their own literate status. People began to acknowledge the way texts are able to shape lives and actions and redefine what it means to be a human in the world. As we are being surrounded by computation in the same way people were once surrounded by writing, we are developing a computational mentality. We are recognizing the role that computation plays in our lives and adjusting our models of the world and ourselves to reflect that role.

A literate mentality was one result of writing becoming an infrastructural technology in society; an increased pressure on individual literacy was another. As writing became so pervasive and as the ability to read and write was increasingly called upon for everyday activities, these skills could no longer be shared by social groups. Collective literacy hit its limits, and now

it appears as though collective computational literacy might be doing the same thing. Although code is embedded in the infrastructure of our workplaces, government, and daily communications, programming is not yet a skill required for participation in social, civic, and commercial life in America; it is not yet a "necessity of life." But diversifying applications for programming, the critical need to understand aspects of computation in order to understand proposed laws and privacy rights, and the individual interest in learning programming that reflects those phenomena all seem to point to an increased role for programming in everyday life—perhaps even to a future of computational literacy.

What might a world with mass computational literacy look like? We might begin to see practices and institutions being built on the assumption that everyone can program. We might see computational solutions to organizational problems that have previously been addressed by bureaucracy. These shifts might decrease the relative value of textual literacy. In big and



Coding Literacy : How Computer Programming Is Changing Writing

由 Annette Vee, , Matthew Fuller, , Lev Manovich, , and Noah Wardrip-Fruin

出版社
MIT Press

日期
2017-07-28

更多...

书内检索

目录

Intro
Contents
Series Foreword
Acknowledgments
Introduction: Computer Programming as Literacy
1 Coding for Everyone and the Legacy of Mass Literacy
2 Sociomaterialities of Programming and Writing
3 Material Infrastructures of Writing and Programming
4 Literacy for Everyday Life
Surrounded by Writing
Surrounded by Computation
From Computational Mentality to Literacy?
Conclusion
Conclusion: Promoting Coding Literacy—Lessons from Reading and Writing
Notes

it appears as though collective computational literacy might be doing the same thing. Although code is embedded in the infrastructure of our workplaces, government, and daily communications, programming is not yet a skill required for participation in social, civic, and commercial life in America; it is not yet a "necessity of life." But diversifying applications for programming, the critical need to understand aspects of computation in order to understand proposed laws and privacy rights, and the individual interest in learning programming that reflects those phenomena all seem to point to an increased role for programming in everyday life—perhaps even to a future of computational literacy.

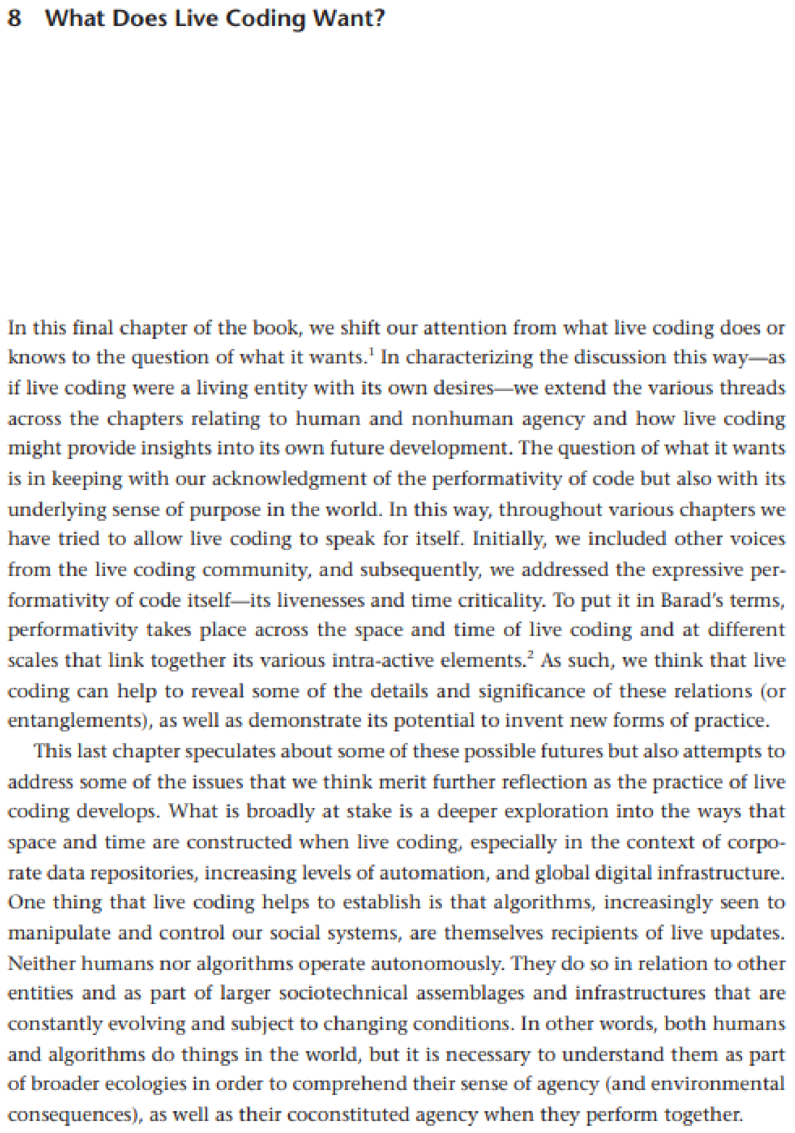
What might a world with mass computational literacy look like? We might begin to see practices and institutions being built on the assumption that everyone can program. We might see computational solutions to organizational problems that have previously been addressed by bureaucracy. These shifts might decrease the relative value of textual literacy. In big and

small ways, mass computational literacy could destabilize the institutions built on textual literacy and the hierarchies established along textual literacy lines (e.g., education). And, as it turns out, we can see glimmers of these effects already. Code-based networks, in various forms, are the most prominent examples of our reliance on computation as infrastructure. But a few of these networks signal something more: they are deeply disruptive of current hierarchies that are based on literacy and bureaucracy. Because it is difficult to regulate code across international boundaries, these networks can even undermine or challenge sovereign states. While most of these networks are reasonably accessible to those without computational literacy, they still order their participants according to what I would call their literate skills, and they rely on significant numbers of participants who are computationally literate.

The World Wide Web is the most obvious example of a code-based network that relies on widespread distribution of computational literacy and which has disrupted literate hierarchies. Since its inception in the early 1990s, the Web has allowed people to put information online using simple forms of code: HTML, or Hypertext Markup Language. More recent developments in HTML and Web standards, and interfaces with more complex programming languages such as Javascript, allow for interactions between users and complex databases and algorithms. Templates allow people to put things on the Web without knowing programming, of course, and people hire Web design firms to build sites just as people would hire scribes to take care of their writing needs when its applications were still restricted. But someone who can build her own site or even customize a Wordpress

READING TEXTUAL

2.Blackwell, Alan F., Emma Cocker, Geoff Cox, Alex McLean, and Thor Magnusson. Live coding: a user's manual. MIT Press, 2022.



8 What Does Live Coding Want?

In this final chapter of the book, we shift our attention from what live coding does or knows to the question of what it wants.[1] In characterizing the discussion this way—as if live coding were a living entity with its own desires—we extend the various threads across the chapters relating to human and nonhuman agency and how live coding might provide insights into its own future development. The question of what it wants is in keeping with our acknowledgment of the performativity of code but also with its underlying sense of purpose in the world. In this way, throughout various chapters we have tried to allow live coding to speak for itself. Initially, we included other voices from the live coding community, and subsequently, we addressed the expressive performativity of code itself—its livenesses and time criticality. To put it in Barad's terms, performativity takes place across the space and time of live coding and at different scales that link together its various intra-active elements.[2] As such, we think that live coding can help to reveal some of the details and significance of these relations (or entanglements), as well as demonstrate its potential to invent new forms of practice.

This last chapter speculates about some of these possible futures but also attempts to address some of the issues that we think merit further reflection as the practice of live coding develops. What is broadly at stake is a deeper exploration into the ways that space and time are constructed when live coding, especially in the context of corporate data repositories, increasing levels of automation, and global digital infrastructure. One thing that live coding helps to establish is that algorithms, increasingly seen to manipulate and control our social systems, are themselves recipients of live updates. Neither humans nor algorithms operate autonomously. They do so in relation to other entities and as part of larger sociotechnical assemblages and infrastructures that are constantly evolving and subject to changing conditions. In other words, both humans and algorithms do things in the world, but it is necessary to understand them as part of broader ecologies in order to comprehend their sense of agency (and environmental consequences), as well as their coconstituted agency when they perform together.

MY CONCLUSIONS

These two articles put forward the view that programming ability should be regarded

Firstly, like reading and writing, programming is a way of expression and communication. Programming language is a language that can help people describe and solve problems. Mastering programming language is like mastering a language, which can enable people to communicate and express themselves more effectively.

Secondly, programming ability, like reading and writing, is a combination of skills and abilities. Programming requires mastering specific syntax and rules, but also requires understanding how to apply these rules to specific problems. This combination of skills and abilities can cultivate creative thinking and problem-solving skills.

Thirdly, programming ability is one of the cores of digital literacy. In modern society, digital technology has penetrated into almost all fields. Mastering programming ability means being able to better understand digital technology and use it to solve problems and create value. This digital literacy is crucial for future work and life.

Overall, these two articles believe that it makes sense to call programming ability a literacy skill. Programming is a way of expression and communication, requires a combination of skills and abilities, and is also one of the cores of digital literacy. Mastering programming ability can help people better participate in the digital world and gain more opportunities and achievements in future work and life.