Great Public Schools for Every Student

Applied Programming

Educator designs integrated learning experiences that include computational thinking, algorithm development, and coding as tools to solve problems.

Key Method

The educator demonstrates the use of programming to solve a problem within a community, guides students in the design of a flowchart or pseudocode to illustrate the algorithm design process, and collaborates with students to proofread to trace code and/or debug programs to solve logic and syntax errors.

Method Components

Computer science moves beyond using technology tools toward an understanding of how they work and ultimately designing new solutions to enduring human problems. Despite common misperceptions, computer science is not simply programming. Like any scientific discipline, computer science consists of a body of knowledge that informs how people understand and perceive the world around them, as well as practices for exploration, creation, and experimentation. Programming, defined as giving computers instructions to follow, is a practice used in computer science. The field itself is much broader, much as biology is not simply conducting lab experiments.

- Over 70% of jobs in STEM are actually computing jobs, and most of the others use computer science as a core part of the job.
- Many future jobs and opportunities will require knowledge and skills in the area of computer science. Therefore, students need multiple opportunities to use computer science to help them explore and understand the world.
- Even a student who does not end up programming in their job will still need to understand the central principles of how data, networking, the Internet, and cybersecurity impact the lives of people in their families and communities.
- Students need to know that when they use a free social media platform, their data can be shared with anyone.
- All of the strands of computer science have drastic impacts on how we live our lives.
- Understanding the basic principles of computer science influences how students will interact with the world around them.

What is Programming?

Programming is the art of encoding algorithms into a programming language. The programming language will provide instructions for a computing device to perform various tasks. Flowcharts and algorithms are the graphical and text representations of the procedures in the computer program. Both flowcharts and pseudocode are "language agnostic," which means that the process is emphasized, rather than the syntax of a specific language.

Flowcharts and Pseudocode

Flowcharts are graphical. There are standardized boxes that represent start and endpoints, inputs, decision statements,

processing, and other events within a program.

Pseudocode is a detailed list of the steps used in a program. Rather than using the syntax of a specific coding language, pseudocode is written in natural language. Using pseudocode allows the designers to create the process without focusing on the syntax of the coding language. Once a program is written in pseudocode, it can then be coded using the computer language.

Programming Concepts

- Flowcharts or Pseudocode
- Variables (declaring, initializing, data types, and/or data structures*)
- Variable Scope
- Operators/Operands
- Logical Operators
- Methods/Functions
- Arguments/Parameters

Program Design

- User Input, Reading User Input, and/or File Input/Output
- Conditional Statements (If, If-Else, Switch statements)
- Repetitive Statements (While, Do-While, For Loops, may include Recursion)
- Nested Structures
- Data Structures

Best Practices

- Pair Programming
- Use of Comments
- Programming Efficiency
- Intellectual Property Rights

Supporting Research

Burgstahler, Sheryl. "Differentiating for Diversity: Using Universal Design for Learning in Elementary Computer Science Education." Universal Design: Implications for Computing Education, ACM Transactions on Computing Education, Oct. 2011, https://staff.washington.edu/sherylb/ud_computing.html

Honey, Margaret, et al. "STEM Integration in K–12 Education: Status, Prospects, and an Agenda for Research." The National Academies Press, National Academy of Engineering and National Research Council of The National Academies, 7 Feb. 2014, http://www.nap.edu/catalog/18612/stem-integration-in-k-12-education-status-prospects-and-an

Lewis, Colleen, and Niral Shah. "How Equity and Inequity Can Emerge in Pair Programming." Association for Computing Machinery, ICER '15 Proceedings of the Eleventh Annual International Conference on International Computing Education Research, July 2015, http://blogs.hmc.edu/lewis/wp-content/uploads/sites/2/2013/07/LewisShah2015_EquitySpeed.pdf

Lewis, Colleen M. "Good (and Bad) Reasons to Teach All Students Computer Science." SpringerLink, Springer, Cham, 1 Jan. 2017, https://docs.google.com/document/d/1R57koI5El5B6jZQyZkmG4NY9MM4wwfJ13V13Yx4gWzw/edit#heading=h.gjdgxs

Resources

Understanding Flowchart/Pseudocode

Algorithm and Flowchart Manual for Students

http://www.yspuniversity.ac.in/cic/algorithm-manual.pdf

Flowchart Lesson

http://accioneduca.org/admin/archivos/clases/material/flowchart_1563990484.pdf

How to Create a Basic Flowchart

https://support.office.com/en-us/article/create-a-basic-flowchart-e207d975-4a51-4bfa-a356-eeec314bd276

Introduction to Pseudocode

http://bioinformaticsalgorithms.com/excerpt/Pseudocode.pdf

Pseudocode Standard

https://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html

Twine Is an Open-Source Tool for Telling Interactive, Nonlinear Stories.

https://twinery.org/

Google Drawing

https://docs.google.com/drawings

Draw.io

https://www.draw.io/

Canva

https://www.canva.com/graphs/flowcharts/

Programming Concepts

Codecademy – Programming Online Modules

https://www.codecademy.com/catalog/subject/programming

EdX Introduction to Python

https://www.edx.org/professional-certificate/introduction-to-python-programming

Khan Academy – Computer Programming

https://www.khanacademy.org/computing/computer-programming

Coursera – Learn Python

https://www.coursera.org/learn/python

Coursera – Learn Java

https://www.coursera.org/specializations/java-programming

Aaronson, Leslie, and Jake Baskin. "Guide to Inclusive Computer Science Education: How Educators Can Encourage and Engage All Students in Computer Science." National Center for Women & Information Technology, 22 May 2019, http://www.ncwit.org/resources/guide-inclusive-computer-science-education-how-educators-can-encourage-and-engage-all.

Submission Guidelines & Evaluation Criteria

To earn the micro-credential, you must receive a passing score in Parts 1 and 3 and receive a proficient for all components in Part 2.

Part 1. Overview Questions

250–500 words

1. Identify your current position and title, number of years working in education, and description of your school community climate (demographic information).
2. Describe your current knowledge level with programming, including any formal or informal training you have had.
3. What impact on your students do you hope completing this micro-credential will have?
4. In the field of computer science, women and minorities are underrepresented. How will you intentionally differentiate instruction to

- **Passing:** Passing: Response provides accurate information that justifies the reason for choosing this micro-credential to address specific needs of both the teacher and the student. Educator includes a learning goal that describes what they hope to gain from earning this micro-credential. Specific details about how you will engage and inspire underrepresented minorities and girls are included.

Part 2. Work Examples / Artifacts

To earn this micro-credential, please submit the following **four artifacts** as evidence of your learning. *Please do not include any information that will make you or your students identifiable to your reviewers*.

Artifact 1: Lesson Plan or Unit of Study

Design an integrated lesson or unit of study that incorporates:

- Computer Science Teacher Association and/or State Computer Science Standards
- Rationale for using programming as a tool to solve an identified problem
- Explanation of how your lesson plan integrates computational thinking strategies: decomposition, pattern recognition, abstraction, and algorithm design
- Action steps for guiding students in tracing code or debugging errors
- Strategies for assessment
- Strategies to engage and inspire underrepresented minorities and girls through the design of your unit
- Integration with a Content Standard

Artifact 2: Teacher Programming File

Develop a program using a programming tool/language of your choice. The program should be designed to solve a pre-identified problem (this can be the solution to Artifact 1). Submit the Teacher Programming File via a pdf, screenshot, or text file.

Artifact 3: Student Programming File

Develop a program using a programming tool/language of your choice. The program should be designed to solve a pre-identified problem. Submit the Student Programming File as a pdf, screenshot, or text file.

Artifact 4: Video of Program in Action

Create a short video that includes teacher or student narration or written caption of the program submitted for Artifact 2 or Artifact 3. Video should illustrate the program in action (executed/running), completing the task. Adhere to district policies on media recording in the classroom.

	Basic	Developing

	Proficient		
Artifact 1: Lesson Plan or Unit of Study	The lesson plan includes all of the following: -Computer Science Teacher Association and/or State Computer Science Standards -Rationale for using programming as a tool to solve an identified problem -Explanation of how your lesson plan integrates computational thinking strategies: decomposition, pattern recognition, abstraction, and algorithm design -Action steps for guiding students in tracing code or debugging errors -Strategies for assessment -Strategies to engage and inspire underrepresented minorities and girls through the design of your unit - Integration with a Content Standard	The lesson plan includes most of the following: -Computer Science Teacher Association and/or State Computer Science Standards -Rationale for using programming as a tool to solve an identified problem -Explanation of how your lesson plan integrates computational thinking strategies: decomposition, pattern recognition, abstraction, and algorithm design -Action steps for guiding students in tracing code or debugging errors -Strategies for assessment -Strategies to engage and inspire underrepresented minorities and girls through the design of your unit -Integration with a Content Standard	The lesson plan includes only a few of the following: -Computer Science Teacher Association and/or State Computer Science Standards -Rationale for using programming as a tool to solve an identified problem -Explanation of how your lesson plan integrates computational thinking strategies: decomposition, pattern recognition, abstraction, and algorithm design -Action steps for guiding students in tracing code or debugging errors -Strategies for assessment -Strategies to engage and inspire underrepresented minorities and girls through the design of your unit - Integration with a Content Standard
Artifact 2: Teacher Programming File	Program is easy to read and includes the use of comments within the code that illustrate the use of: Minimum of five programming concepts AND Minimum of three programing design elements	Program is easy to read and includes the use of comments within the code that illustrate the use of: Only two to four programming concepts AND Only two programing design elements	Program is easy to read and includes the use of comments within the code that illustrate the use of: Only one programming concept AND Only one programing design element

Artifact 3: Two Student Programming Files	Program is easy to read and includes the use of comments within the code that illustrate the use of: Minimum of five programming concepts AND Minimum of three programing design elements	Program is easy to read and includes the use of comments within the code that illustrate the use of: Only two programming concepts AND Only two programing design elements	Program is easy to read and includes the use of comments within the code that illustrate the use of: Only one programming concept AND Only one programing design element
Artifact 4: Video File	Video must have all of the elements listed below: -Teacher or student narration or written caption of the program -Demonstration of the program in action (executed/running), completing the task -Adheres to district policies on media recording in the classroom	Video has only some of the elements listed below: -Teacher or student narration or written caption of the program -Demonstration of the program in action (executed/running), completing the task -Adheres to district policies on media recording in the classroom	Video is missing most of the elements listed below: -Teacher or student narration or written caption of the program -Demonstration of the program in action (executed/running), completing the task -Adheres to district policies on media recording in the classroom

Part 3 Reflection

250-500

1. How did completing this micro-credential impact your understanding of teaching programming?
2. What factors influenced your choice of programming tool/programming language?
3. What was the change in your students' belief in themselves as a computer programmer (note if there are any students in underrepresented groups that were impacted)?
4. How did you and your students handle the challenges encountered during the development process?

- **Passing:** Passing: Reflection provides evidence that this activity has had a positive impact on both educator practice and student success. Specific examples are cited directly from personal or work-related experiences to support claims. Also included are specific actionable steps that demonstrate how new learning will be integrated into future practice.