Access

Multiple Means of **Engagement**

Design Options for **Welcoming Interests & Identities**

- Give students choices (choose project, software, topic)
- Encourage students to make projects relevant to culture and interests
- Allow for differences in pacing and length of work sessions
- Provide options for managing background noise and visual stimulation, both in the general classroom environment (e.g., listening to music with headphones or using noise-canceling headphones) and within CS online and physical tools
- Provide opportunities for exploring creative uses of coding (e.g., stories or games)
- Ensure the CS learning environment is welcoming and inclusive of all learners (e.g., proactively attending to approaches to promote active engagement by all learners, ensuring physical environment is safe and accessible, establishing class norms for respectful sharing)

Multiple Means of **Representation**

Design Options for **Perception**

- Present information through multiple modalities (visual, auditory, and tactile)
- Model computing tasks and activities using both physical and visual representations (e.g., interactive whiteboards, videos, manipulatives)
- Provide access to modeled code while students work independently
- Provide access to video tutorials of computing tasks with closed captioning
- Use accessible CS tools, applications, and websites that allow the students to adjust visual and auditory settings (e.g., font size & contrast)
- Introduce diverse role models and contributions in CS (e.g., computer scientists who are women, people with disabilities, and people from different cultural and language backgrounds)
- Highlight the contributions of individuals from underrepresented groups in CS

Multiple Means of **Action & Expression**

Design Options for **Interaction**

- Include hands-on, physical computing activities (e.g., CS Unplugged, using microcontrollers) that allow students to demonstrate abstract concepts through movement and tangible interaction
- Ensure appropriate use of assistive technologies like larger or smaller mice, alternative keyboards, or touch-screen devices to optimize access to CS tools and curricula to comply with students' Individualized Education Programs or 504 Plans
- Offer students the choice to code using different formats, such as block-based programming, text-based coding, or voice-activated commands, and/or provide options for students to switch modalities
- Incorporate a variety of accessible CS tools that include a range of options to increase learner interaction for students with a range of functional needs

UF | CSEveryone Center *for* Computer Science Education

Support

Multiple Means of **Engagement**

Design Options for
Sustaining Effort & Persistence

- Regularly introduce and refer back to computing and content goals in learner-friendly language such as "I can" statements
- Provide differentiated tasks with various levels of challenge, offering extensions or additional support when appropriate such as using multiple entry point activities (e.g., worked examples, buggy projects, exploded code/Parsons Problems, and extensions)
- Explicitly teach and encourage peer collaboration
- Implement pair programming and group work with clearly defined roles
- Provide real-world CS examples where persistence leads to innovative solutions
- Recognize students for demonstrating perseverance and problem solving in the classroom
- Create a classroom culture that celebrates diverse perspectives and approaches in CS
- Offer timely, actionable feedback that focuses on specific improvements and next steps

Multiple Means of **Representation**

Design Options for
Language & Symbols

- Teach and review content-specific and CS-related vocabulary (e.g., conditional statement, variables, nested loops) by anchoring new concepts to known knowledge and by using multiple approaches (e.g., word walls, reinforcements through modeled activities)
- Post anchor charts and provide reference sheets with images of blocks or with common syntax when using text
- Allow students to use coding languages and tools in their native language or offer dual-language resources to support multilingual learners
- Ensure that instructional materials are devoid of biased or exclusionary language or perspectives (e.g., visuals only showing men as programmers)
- Ensure that programming examples provide diverse representations in terms of gender, race, culture, and ability
- Use a variety of media (e.g., videos, animations, diagrams) to explain key CS concepts

Multiple Means of **Action & Expression**

Design Options for
Expression & Communication

- Allow students to communicate their understanding and creativity through various media (e.g., pseudocode, video explanations, digital presentations, or physical demonstrations)
- Give opportunities to practice computing skills and content through projects that build on prior lessons
- Provide sentence starters or checklists for communicating with others, explaining work, and offering feedback
- Provide unplugged activities, physical manipulatives (e.g., command cards or block-based coding pieces), and/or robotics for hands-on, tangible engagement in computing
- Use scaffolding techniques, such as starter code or guided tutorials, to support students in mastering complex CS tasks gradually
- Guide students in reflection of how certain modes of communication in CS might be more or less accessible to themselves and to others (e.g., use of only visual or auditory outputs may be inaccessible to some peers)

UF | CSEveryone Center *for* Computer Science Education

Executive Function

Multiple Means of **Engagement**

Design Options for
Emotional Capacity

- Clearly communicate expectations for computing tasks, peer collaboration, and help-seeking in multiple ways (e.g., verbally, written, signage)
- Break up activities with reflective practices where students assess their own progress on their computational activities and consider how their work impacts group collaboration (e.g., journaling, classroom discussion, "turn and talks")
- Use formative and summative assessments that evaluate both content and process (e.g., rubrics, exit slips)
- Acknowledge difficulty and frustration, and model strategies for dealing with frustration or conflict
- Guide students to reflect on the real-world implications of their work, prompting them to consider how their programs and designs can be accessible and beneficial to a variety of users

Multiple Means of **Representation**

Design Options for
Building Knowledge

- Activate students' prior knowledge through connections to previous lessons or opportunities to relate their real-world experiences to new CS concepts
- State and reinforce lesson goals and learning objectives (e.g., through " I can" statements)
- Use analogies and other anchoring techniques to make cross-curricular connections (e.g., comparing debugging code to the editing process in writing)
- Provide graphic organizers for students to "translate" programs into pseudocode
- Provide multiple ways for understanding CS through diverse methods (e.g., unplugged activities, visual tutorials, or narrative-driven projects)
- Explore how students' unique cultural, linguistic, or personal backgrounds influence their approach to learning CS
- Provide opportunities for students to apply CS concepts learned in class to new contexts

Multiple Means of **Action & Expression**

Design Options for
Strategy Development

- Guide students in setting clear, achievable goals for short- and long-term CS projects (e.g., implement planned checkpoints and opportunities for self-assessment)
- Provide exemplars or worked examples to scaffold project planning
- Embed prompts throughout lessons to encourage students to anticipate potential issues in their projects (e.g., stop and plan, test, debug)
- Provide graphic organizers to facilitate planning, goal-setting, and debugging
- Provide strategies for working independently through challenging computational activities (e.g., rubber duck debugging, Debugging Detective, think-alouds)
- Encourage students to ask questions as comprehension checkpoints
- Empower students to recognize and challenge inequitable or unjust CS practices

UF | CSEveryone Center *for* Computer Science Education