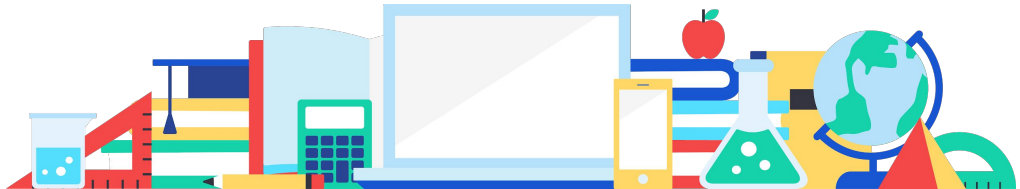
AP Computer Science Principles: Create Task

Updated Date March 7, 2025



Training Overview

By the end of this guide you will

➔ **Understand the Create Task Requirements**

- Learn what students must include in their program (sequencing, selection, iteration, input/output).
- Understand how these components align with the AP CSP rubric.

➔ **Identify Appropriate Tutoring Boundaries**

- Distinguish between acceptable guidance (clarifying general concepts, teaching general debugging techniques) and inappropriate help (writing code, providing solutions).
- **Please note that tutors are not allowed to provide any help on any element of the student's actual Create Task, always ask students if they are working on the task or simply preparing for it.**

➔ **Address Common Challenges**

- Support students in:
  - Debugging their code systematically.
  - Planning and implementing project ideas effectively.
  - Meeting key task requirements like abstraction and algorithm development.

AP Computer Science Principles Course and Exam Description

Create Task Requirements

Create Task 101

AP CSP Student Task Directions

The Basics

- **Task Objectives**: Students must develop a program (independently) that demonstrates their skills in problem-solving, creativity, and programming concepts.
- **Components**: The Create Task requires students to...
  - Develop a **program** with at least one input, output, and an algorithm that includes sequencing, selection, and iteration.
  - Submit a **video** (up to 1 minute) demonstrating their program's functionality.
  - Create a **Personalized Project Reference**, a student-authored document containing screen captures of their program's lists and procedures. (Tip sheet)
  - Write a **written response** explaining their code during the AP exam.
- **Scoring Criteria**: The **AP CSP scoring rubric** evaluates the task based on criteria like:
  - Program Functionality and Purpose
  - Algorithm Implementation
  - Abstraction
  - Written Responses

What You Need to Know

Be familiar with

- **Content**
  - **Algorithms:** Explaining how sequencing, selection (if/else statements), and iteration (loops) work in their code.
  - **Abstraction:** Identifying parts of their program that simplify functionality or make it reusable.
  - **Inputs/Outputs:** Demonstrating how user input is used to produce an output.
  - **Scoring Rubric**: AP Computer Science Principles Course and Exam Description
  - **Student Guidance:** AP CSP Student Task Directions
    - Specific instructions may seem odd for experienced programs (limited in code commenting, etc.) so please review
  - **Example Task:** CodeHS

- **Tools & Programming Languages**
  - Be familiar with block-based (Scratch, App Lab) and text-based programming languages (Python, JavaScript, etc).*
  - Help students choose a language or tool they feel comfortable using and that meets the task requirements.

*Note: It's okay to tell a student to make a new request if you aren't familiar with the language they're using. Another coach might be able to help. There is no standard language for the Create Task so we don't guarantee each coach will have specific language knowledge.

Tutoring the Create Task

Do’s and don’ts

Know What You Can and Cannot Do

- **Tutors can:**
  - Help students understand the task directions and scoring rubric.
  - Explain general programming concepts without applying them to the student’s project.
  - Guide students in understanding algorithms, abstraction, sequencing, selection, and iteration–but not in direct relation to their project.
  - Help students learn tools and platforms (e.g., Scratch, Python, JavaScript).
  - Discuss common debugging techniques in a general sense (e.g., using print statements) before the Create Task begins.
  - Ensure students understand academic integrity and plagiarism policies.
- **Tutors cannot:**
  - Contribute directly to the student's code, video, personalized project reference, or any other task element.
  - Provide specific ideas or solutions for their program.
  - Edit or rewrite the student’s written responses or code.
  - Debug, fix, or suggest edits to student code once the Create Task has officially started
  - Suggest or modify project ideas or algorithms for students.
  - Review or revise a student’s written responses.

Helping students write code, provide ideas, or edit their work directly could result in students failing the course or exam. It’s very important you don’t do these things–even if a student asks you nicely!

Only after May 1st

After students upload their Create Task

- **Tutors can:**
  - Look at student code and PPR
  - Help students form written responses to sample prompts related to their PPR (in preparation for the exams)
- **Tutors cannot:**
  - Write example responses
  - Give direct answers
  - Provide improvements to the code or PPR

Do not help students on their tasks before they submit them, always check if a student is still working on their Task.

Addressing Common Challenges

3 Top Challenges

How to help

- **Debugging Code:** Students may struggle to debug their programs.
  - Teach students to use debugging tools or print statements on code that is not part of the project.
  - Demonstrate systematic testing to isolate issues on code that is not part of the project.
- **Meeting Requirements:** Students often forget to include all required elements.
  - Encourage students to check their work against the rubric and student handout
  - Remind them to include input, output, sequencing, selection, iteration, procedure with parameter(s), and a list.
- **Helping with Planning & Algorithm Development (Before the Task Starts):** Students may find it difficult to come up with an idea or figure out how to implement their project.
  - Discuss problem-solving strategies in general terms.
  - Provide practice exercises unrelated to the student's Create Task.
  - Explain general approaches to structuring a program.

Coaching Examples

Clarifying Task Requirements

Explain general concepts

Do

Student: “I don’t understand selection and iteration.”

Tutor: “Selection means making decisions with if-statements. Iteration is using loops. Your program must demonstrate both. Do you want me to explain them in more detail?”

✅ This explains general concepts without providing specific solutions.

Don’t

Student: “I don’t understand selection and iteration.”

Tutor: “Yes, and here’s an example for each: sequencing is like writing print('Hello'), selection is using if x > 5, and iteration is a for loop like for i in range(10). Just use something like that in your code.”

❌ This provides specific examples or solutions that the student might copy directly, which goes beyond clarifying concepts and violates the principle of ensuring the student completes their own work.

Encouraging Debugging Strategies (Before the Task Starts)

Help students troubleshoot

Do

Student: “My practice code doesn’t work, but I can’t figure out why.”

Tutor: “Sounds like you haven’t started the Create Task yet right?

Student: “No I’m preparing for it.”

Tutor: “Great, then I can help: Have you tried adding print statements or console logs to see the value of your variables at different points in the program?”

✅ This helps students troubleshoot their program without fixing it for them.

Don’t

Student: “I can’t figure out how to loop through this list.”

Tutor: “Here’s how you can do it: for item in list: print(item). Just copy and paste that into your program.”

❌ This directly provides the solution, which is against the rules.

Supporting Algorithm Development (through alternative ideas & general content)

Teach computational thinking

Do

Student: “I want my program to calculate a user’s BMI based on height and weight. How should I start?”

Tutor: “That’s a great idea! I can’t help you on your Create Task but we work through a different problem together!”

✅ This encourages planning and critical thinking without working on the Task.

Don’t

Student: “I want my program to calculate a user’s BMI based on height and weight. How should I start?”

Tutor: “Use weight / (height * height) and store it in a variable.”

❌ This is NOT OK because it provides the actual code and helps the student on their Create Task directly.

Encourage Utilizing Approved Resources

If a student has started their Create Task

Do

Student: "I'm stuck on my loop. It keeps running infinitely, and I don't know how to fix it. Can you help me?"

Tutor: "Since you're already working on the Create Task, I can't debug your code for you. However, you could try talking to a classmate or use AI tools to learn about common debugging strategies, but make sure you fully understand any code suggestions before using them."

✅ This encourages use of approved collaboration

Don't

Student: "I'm stuck on my loop. It keeps running infinitely, and I don't know how to fix it. Can you help me?"

Tutor: "I see the problem, here's how to fix it..."

❌ This is NOT OK because it provides the actual code and helps the student on their Create Task directly.

Resources

- AP CSP Student Task Directions
- AP CSP Create Task Rubric (2025)
- Course Overview
- Student Samples and Scoring Commentary (note major changes occurred in 2025)
- Code.org Unit 9 CSP (create a teacher account)