Computational Thinking

What is Computational Thinking (CT)?

A problem-solving process involving skills and techniques to understand and solve complex problems, applicable across various disciplines.

Why use Computational Thinking (CT)?

With the increased focus on AI and large language models in particular, being able to systematically explain your work process is going to be very important.

Thinking through ideas and developing explainable frameworks for their individual parts will become important when solving them using AI.

What are the core concepts of CT?

Decomposition

Breaking down complex problems into smaller, more manageable parts.

Pattern Recognition

Identifying similarities or patterns to make predictions or solve problems more efficiently.

Abstraction

Simplifying complex problems by focusing on the important information only, ignoring irrelevant details.

Algorithm Design

Creating step-by-step instructions or rules to solve problems or perform tasks.

Decomposition

Example: In software development, decomposition involves breaking down a large project into smaller, manageable tasks.

For instance, developing a new application might be divided into tasks such as designing the user interface, setting up the database, creating user authentication, and testing the application.

Decomposition - question

How might breaking down a complex problem into smaller parts change your approach to problem-solving in your current projects?

Pattern Recognition

Example: In computational linguistics, pattern recognition is used to identify common syntactic structures or semantic patterns in large text corpora.

By recognizing these patterns, researchers can develop algorithms to automate tasks such as language translation or sentiment analysis.

Pattern Recognition - question

Can you think of a research project where identifying patterns in your data led to new insights or breakthroughs?

Abstraction

Example: In computer science education, abstraction is used to teach programming concepts by focusing on the core principles of algorithms and data structures, while omitting the complexities of specific programming languages.

This helps students understand the underlying logic and apply it across different contexts.

Abstraction - question

What challenges do you face when trying to simplify complex concepts in your field, and how do you decide which details to focus on?

Algorithmic Design - question

Example: In cooking, algorithm design is akin to creating a recipe.

A recipe provides step-by-step instructions for preparing a dish, ensuring that even novice cooks can achieve consistent results.

Algorithmic Design

How do you determine the priority of tasks when designing algorithms for your academic projects, and what criteria do you use to ensure that the most critical tasks are addressed first?

Key Skills related to CT

Logical Thinking

Using clear, structured reasoning to solve problems.

Critical Thinking

Evaluating and analyzing information to make informed decisions.

Creativity

Thinking outside the box to develop innovative solutions.

Persistence

Continuing to work through challenges and difficulties.

Applications of CT

Cross-disciplinary

Useful in fields like mathematics, science, engineering, and humanities.

Real-world Problem Solving

Applicable in everyday life situations, from planning a trip to managing a project.

Some additional considerations

Do you have some criteria of success or failure?

Do you know when you have successfully completed the task or is the end goal vague?

Do you understand the purpose of the task you are trying to achieve?

This will help you to be able to improvise in cases where some standard ways of doing things become problematic.

Examples: writing academic articles - decomposition

Break Down the Writing Process:

Divide the task into smaller, manageable parts such as conducting literature reviews, developing a thesis statement, outlining the article, writing each section (introduction, methodology, results, discussion), and revising the draft.

This approach helps in organizing the writing process and tackling each component systematically.

Examples: writing academic articles - pattern recognition

Identify Common Structures

Recognize patterns in the structure and style of academic articles within the field.

This can involve understanding the typical flow of arguments, common methodologies, and citation practices, which can guide the writing process and ensure adherence to academic standards.

Examples: writing academic articles - abstraction

Focus on Core Ideas

Simplify complex research findings by focusing on the essential ideas and arguments that need to be communicated, while omitting less relevant details.

This helps in crafting a clear and concise narrative that highlights the key contributions of the research.

Examples: writing academic articles - algorithmic design

Develop a Writing Plan

Create a step-by-step plan or timeline for completing the article, including deadlines for each section, time allocated for revisions, and submission dates.

This structured approach helps in managing time effectively and ensuring that all components of the article are completed on schedule.

Open-ended questions:

1. Conceptually would the strategies provided by computational thinking help you in completing tasks more efficiently?
2. Do you think that the effectiveness of computational thinking depends on the person’s personality type?
3. Would this type of thinking be an option for you to integrate into your current workflow?