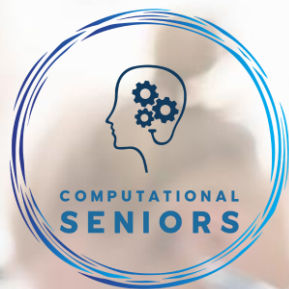




Co-funded by
the European Union



MODULE 1

How do computers think?. Understanding Computational Thinking

Welcome to the COMPutational Seniors course

This course is designed for adult trainers who want to bring Computational Thinking (referred to as CT as well throughout the modules), into their practice, even if they have little or no background in tech.

Computational Thinking isn't just about programming, it's a way of solving problems, thinking logically and adapting to a digital world. Adults, especially those from vulnerable or low-skilled groups, can benefit greatly from it.

In this course, you will:



01

Understand what computational thinking is and why it matters in adult education.



02

Learn the core CT concepts: decomposition, abstraction, pattern recognition and algorithms



03

Explore didactic strategies to make these concepts accessible, engaging and inclusive.



04

Use CT to develop soft skills like problem-solving and collaboration in your learners.



05

Apply CT across different contexts, from everyday tasks to lifelong learning.



06

Create and adapt your own CT-based activities using course examples.

Ready? Let's discover how Computational Thinking can open new doors for adult education.



WELCOME TO MODULE 1

In this module, you will discover how computers think and the principles that make up computational thinking.

We will explore the conceptual foundations of this methodology, focusing on key elements such as decomposition, pattern recognition, abstraction, and algorithm design, and how these processes enable structured and efficient problem-solving. We will look at how this approach has been adopted in compulsory education across many European countries and analyze its potential application in adult education.

Through practical examples, you will learn to identify these principles in everyday life and apply them in educational contexts tailored to adult learners.

Use this module to help your students develop a mindset they can apply beyond the classroom



STRUCTURE OF THE MODULE

Unit 1. What is Computational Thinking?

- Definition and core principles of CT
- Origins and evolution of the concept
- Key characteristics

Unit 2. The value of Computational Thinking in adult learning

- Why is Computational Thinking important?
- Everyday examples of CT in action
- Benefits for adult learners

Unit 3. Computer thinking, human thinking and Computational Thinking

- How computers process information
- How humans think
- Comparing both types of thinking
- Programming vs. Computational Thinking: core differences

Unit 4. Core principles of Computational Thinking

- Decomposition
- Pattern Recognition
- Abstraction
- Algorithms

Unit 5. Computational Thinking in the European educational structure

- Current landscape of CT in Europe
- How different countries are integrating CT

Unit 6. Case studies and activities

- Real-world examples of CT
- Interactive exercises to explore and apply decomposition, patterns, abstraction, and algorithms

At the end of the course, the learner will be able to...

Learning outcomes

Describe what computational thinking is and how it differs from human thought processes.

Determine how the four key principles of computational thinking (decomposition, pattern recognition, abstraction, and algorithms) can be used to solve problems.

Identify examples of decomposition, pattern recognition and abstraction and how these lead to creating algorithms.

Acknowledge that computers and humans approach problem-solving differently

Demonstrate computational thinking through **decomposition, pattern recognition, abstraction, and algorithm** development.

MODULE AIM and OBJECTIVES

AIM: To introduce the concept of Computational Thinking, explain its core components and highlight its relevance in everyday life and lifelong learning, particularly for low-qualified adults.

OBJECTIVES:

1. Define what CT is and why it matters in today's world.
2. Break down the four core techniques of CT: decomposition, pattern recognition, abstraction, and algorithms.
3. Show how CT helps individuals solve problems more effectively and make everyday tasks easier to manage.
4. Set the foundation for understanding how CT can be integrated into adult education.





UNIT 1

What is Computational Thinking?

What is Computational Thinking?: Definition

Computational Thinking can be defined as a method for understanding and solving different types of problems by applying principles from computer science. It is a reasoning process that helps to analyze and resolve challenges independently of the use of computers, relying on logic, organization and step-by-step thinking.

At its core, it involves breaking down complex problems into smaller parts (decomposition), identifying patterns (pattern recognition), focusing on essential information (abstraction) and designing step-by-step strategies to reach a solution (algorithms).

This way of thinking offers a flexible framework that can be applied to everyday situations and common problem-solving scenarios. Beyond its practical value, Computational Thinking also fosters essential 21st-century skills such as critical thinking, adaptability, creativity, and collaboration.

CT is a core skill in the digital age. As technology continues to evolve rapidly, the ability to think computationally will become even more essential for innovation and success in many industries.



What is Computational Thinking?: Origin



Although the concept of Computational Thinking had been mentioned earlier, it was in 2006 that computer scientist Jeannette Wing formally introduced it in an article published in Communications of the ACM. She described it as a way of solving problems and understanding human behavior using fundamental ideas from computer science.

In her 2006 definition, Wing stated:

“Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.”

Later, in 2011, she refined her definition:

“Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.”

We selected these definitions because they lay the groundwork for understanding CT as both a mindset and a problem-solving approach beyond technical skills. Two ideas from this last definition are particularly important for education:

- ☒ Computational Thinking is a thought process not dependent on computers or technology.
- ☒ It is a specific type of problem-solving that involves designing solutions executable by humans, machines or both.

What is Computational Thinking?: Origin



Wing's definition gave us a starting point, but many educators and researchers have expanded on this idea. Let's look at a few more definitions that add depth and nuance to our understanding of CT.

According to Barr & Stephenson:

“Computational Thinking is a problem-solving process that includes (but is not limited to) the following characteristics: logically organizing and analyzing data, representing data through abstractions, and automating solutions through algorithmic thinking.”

According to ISTE (International Society for Technology in Education):

“CT is a set of problem-solving skills and techniques that software engineers use to write programs, but which can also be used to support problem-solving across all disciplines, including the humanities.”

Each of these definitions highlights a different strength of Computational Thinking. As a trainer, understanding these perspectives helps you introduce CT not just as a method, but as a flexible mindset adaptable to diverse learning needs.



What is Computational Thinking?: Origin



Long before the term Computational Thinking was formally introduced, programming already had a place in education. In the 1980s, many students were introduced to programming through a language called Logo, developed in 1968 by Seymour Papert at MIT. Logo allowed learners to explore mathematical and scientific ideas by giving simple commands to a turtle on the screen, offering immediate, visual feedback that made abstract concepts easier to grasp.

Although Logo gradually faded from classrooms, the early 2000s brought renewed interest in educational programming with the development of tools like Scratch, Alice, Kodu, and AppInventor. These platforms, especially Scratch, made programming far more accessible thanks to their visual, block-based structure, allowing users, particularly children, to experiment with logic and creative problem-solving.

While the initial focus was on school-aged learners, the value of Computational Thinking is now extending into adult education, where it offers powerful strategies for understanding problems and making better decisions, regardless of a person's background or profession.

Today, international efforts continue to promote the integration of Computational Thinking across all stages of education. Increasingly, it is recognized as an important skill for lifelong learning, along with reading and writing.



What is Computational Thinking?: Characteristics

As we have just explored, Computational Thinking is a structured and adaptable approach to problem-solving that draws on key concepts from computer science. It is characterized by:

- It is based on four core principles: decomposition, pattern recognition, abstraction and algorithms, which guide the way problems are understood and approached.
- It encourages a logical and organized mindset, helping individuals break down complex challenges into simpler steps.
- It supports the ability to filter information, focus on what's essential and ignore distractions.
- It promotes strategic thinking, allowing learners to plan, test and adjust actions as needed.
- It is transferable across contexts, helping adults apply the same reasoning process in learning or daily life.

Now that we've defined Computational Thinking and explored its core characteristics, let's now explore its importance and how we already use it without realizing.



Are you following along? Try this quick question to reinforce what you've learned



Computational Thinking...

- A) Encourages a logical and organized mindset
- B) Supports the ability to filter information
- C) Promotes strategic thinking
- D) All of the above



UNIT 2

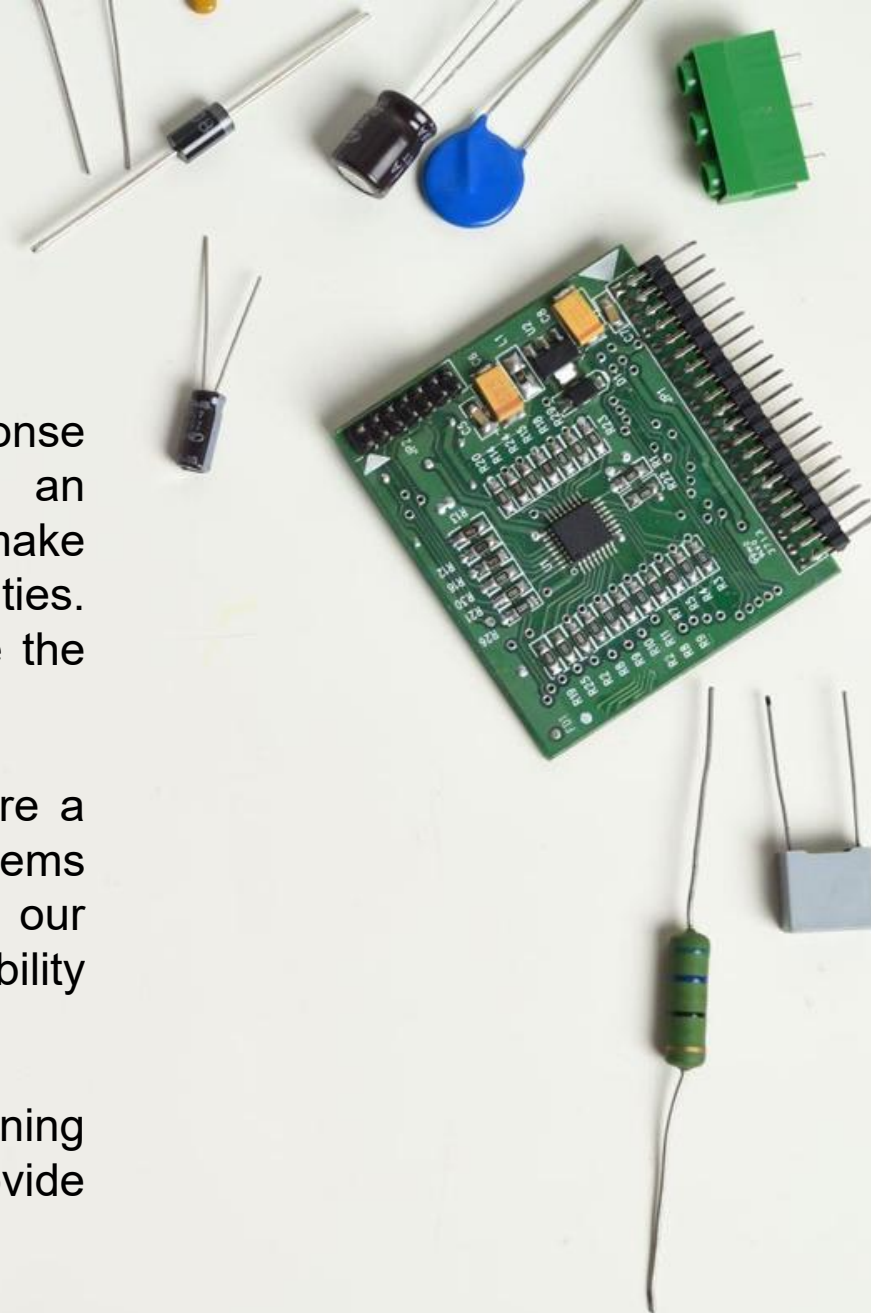
The value of CT in adult learning

Why is Computational Thinking important?

Integrating Computational Thinking into your training it's a necessary response to the world we live in. We are surrounded by technology and an overwhelming amount of information that guides the way we work and make decisions. Teaching CT means giving learners the skills to face these realities. It encourages the development of soft skills to understand and navigate the digital world as users and citizens.

CT integration into adult education It's also about helping people acquire a mindset that prepares them to adapt to change, approach problems strategically and understand the digital world that increasingly defines our everyday lives. In this sense, CT becomes a bridge to inclusion, employability and lifelong learning for adult learners.

Despite growing awareness of its importance, bringing CT into adult learning remains a challenge. As a trainer, you have a unique opportunity to provide adult learners with a mindset that helps them adapt to a fast-paced world.



We use Computational Thinking every day

Computational Thinking may sound technical, but it's something we already do, often without even realizing it. Many everyday tasks involve the same logical steps: breaking things down, identifying patterns, following instructions and solving problems.

Think about these examples:

Following a recipe

You break down a dish into steps (decomposition), follow instructions in order (algorithms), adjust for ingredients or time (abstraction) and recognize patterns in how recipes work.

Doing laundry

You sort clothes by color or fabric (pattern recognition), choose a cycle depending on the load (decision-making) and follow steps in a sequence

Planning a trip

You gather information, choose routes and schedules (algorithmic thinking), and adapt your plan based on budget or time (abstraction and problem-solving).

Explain a concept to students

You simplify it into smaller parts, use examples that match their experience (pattern recognition), and present it in logical steps.

As a **trainer**, recognizing CT in daily activities can help you **make it accessible** for your learners. It's about showing them how to use a way of thinking they already apply and helping them transfer it into **new contexts** like learning, work and digital environments.

What does CT bring to learners?

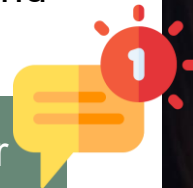
Ultimately, Computational Thinking is not simply about knowing how to program. Through this learning process, learners are equipped with a set of tools that allow them to strengthen their problem-solving abilities, design and create projects, express ideas, improve concentration and make informed decisions.

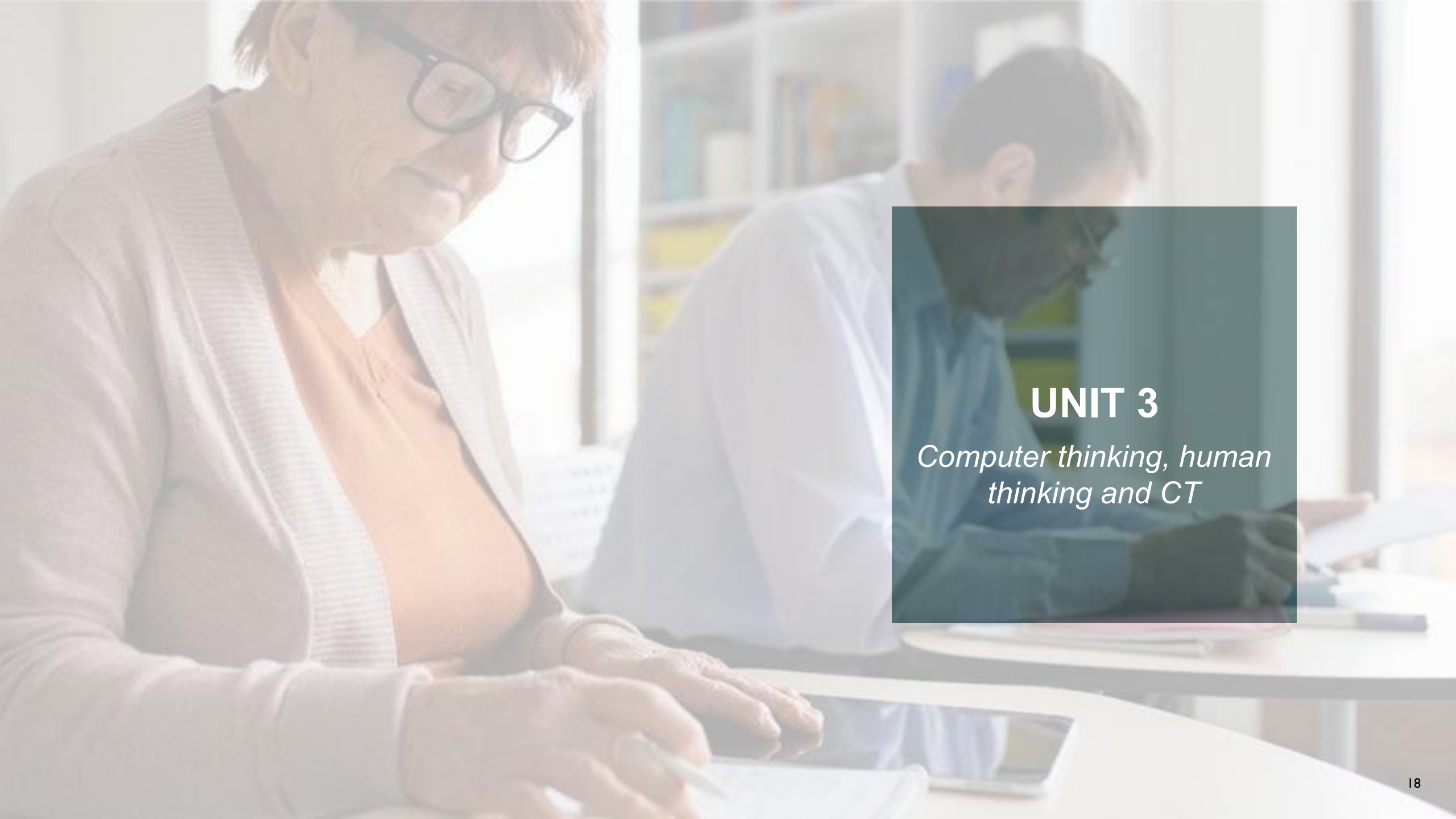
In this sense, adult learners will be able to:

- Enhance their cognitive and socio-emotional skills to solve everyday problems.
- Improve their creativity and imagination to explore alternative solutions to different challenges.

These skills are not only useful in the classroom but also transferable to other areas of everyday life, like employment and collaboration.

As a trainer, you have the chance to bring this mindset into your classroom as a lens to help learners approach problems. What are some daily challenges your learners face that could benefit from a computational thinking approach?





UNIT 3

Computer thinking, human thinking and CT

How do computers “think”?

Now that we’ve defined Computational Thinking, explored its core characteristics and importance, let’s take a closer look at how computers “think”, and how that compares to the way we, as humans, process and solve problems.

Computers don’t think or feel like we do. Instead, they solve problems by following clear, logical instructions. Their power lies in speed, precision, and their ability to repeat actions exactly the same way every time. Understanding how they work helps us better apply the principles of Computational Thinking.

Turning the power on and off

Computers use electricity. At the most basic level, they use the presence (coded as 1) or absence of electricity (coded as 0) in a circuit to form a simple language constituted of two numbers, 0 and 1

Binary code, the language of computers

The assemblage of 0 and 1 is called the binary code. Computers use this code to process and store data. Everything on a computer the user interacts with such as text, images or videos is broken down into 0s and 1s.

Logic gates and decisions

Computers use logic gates (AND, OR, NOT) to make decisions. These gates control the flow of data based on binary input, allowing the computer to execute commands efficiently and perform complex tasks through simple logical operations

Just as computers follow logical sequences, **Computational Thinking helps us solve problems using similar principles**. But what about us, how do we think and process information?



How do humans “think”?

Human thinking is far more flexible and creative than a computer’s logic. Our brains don’t rely on binary code or step-by-step commands. Instead, we learn, feel, adapt and make decisions in ways that are influenced by emotions and experience.

The brain’s structure

The human brain contains around 80 to 100 billion neurons. These cells send signals to each other to help us think, move, feel, and remember.

Neural networks

Neurons build connections called neural networks. The more we use certain pathways, the stronger they become. This is how we learn and develop new habits or skills.

Human cognition

It’s still uncertain how the brain actually takes ideas and combines them in new ways to form newer thoughts.

Understanding how both computers and humans’ approach problem-solving **helps us see the power of Computational Thinking**. Now, let’s compare the two and explore how these differences can guide us in designing better learning experiences.



How do computers and humans think? A comparison

Understanding how computers and humans think helps us appreciate why Computational Thinking is such a powerful tool. While computers rely on logic and structure, human thinking brings flexibility and creativity. Both have strengths and combining them can lead to better problem-solving.

Computers	Humans
Follow strict, step-by-step instructions	Can adapt steps or change approach mid-process
Use binary language (0s and 1s) to process data	Use language, emotion, intuition and experience
Make decisions based on logic gates	Make decisions based on logic and feelings/context
Repeat tasks with precision	Learn from experience and change behavior
Don't "understand" what they do	Reflect, imagine, and create new ideas

Computational Thinking bridges these two worlds: it teaches us to approach problems like a computer (clear, logical, structured) while still using the human ability to adapt, learn, and innovate.

Now that we've explored how computers and humans think differently, it's time to clarify an important distinction that will guide us throughout the module structure: the difference between Computational Thinking and programming.



Programming vs. Computational Thinking: core differences

There is often confusion between CT and programming, but it's important to understand they are not the same. According to the International Society for Technology in Education (ISTE), **both CT and programming use similar cognitive processes and aim to break down and solve problems using algorithmic thinking.** However, the key difference lies in their focus.

Computational Thinking is the cognitive ability to solve problems logically and systematically. It involves analyzing a problem, identifying patterns, abstracting relevant information and designing a step-by-step solution, regardless of whether a computer is involved.

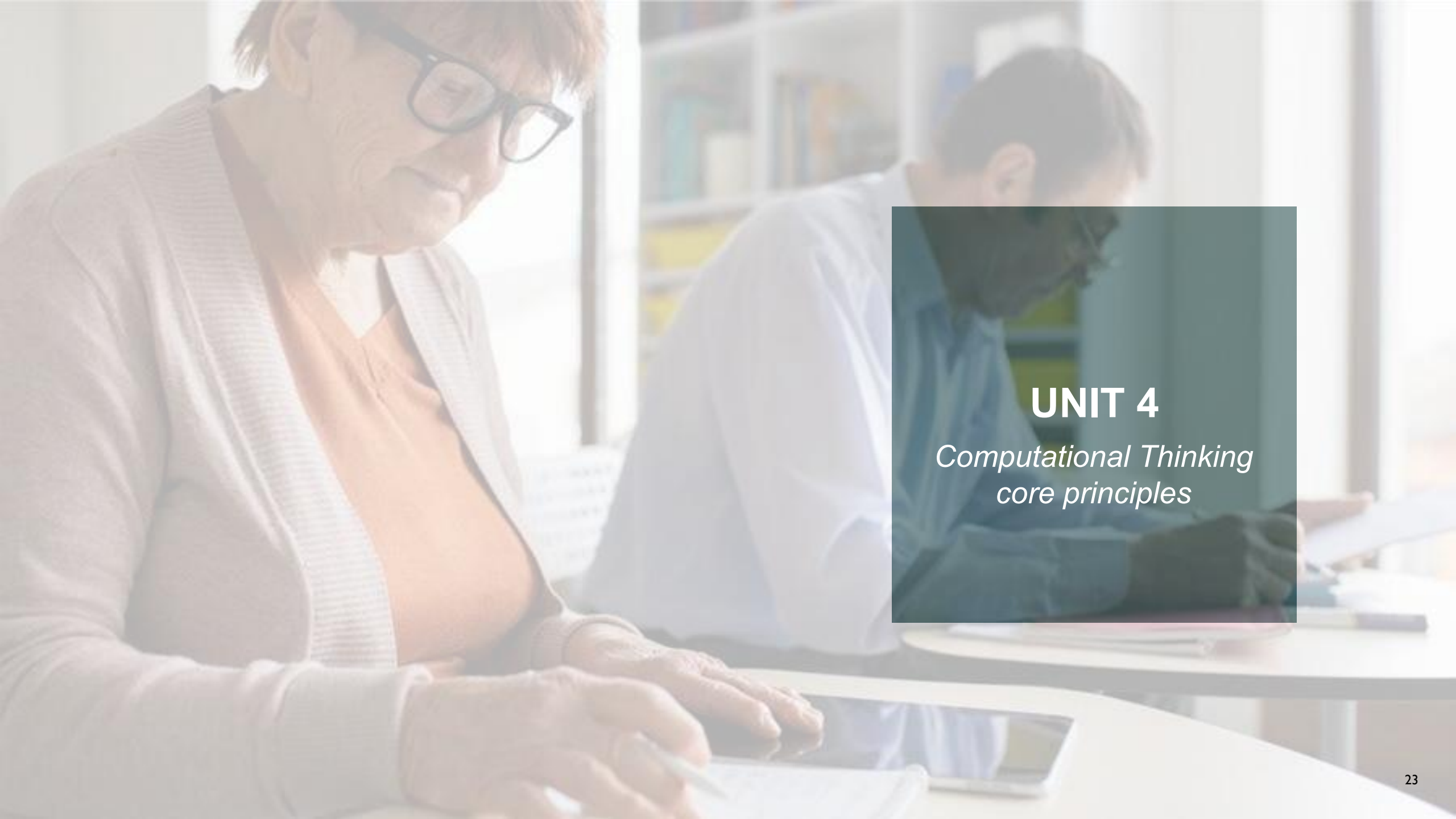
Programming, on the other hand, is the technical skill of implementing those solutions through code. It requires formal learning and familiarity with programming languages to convert ideas into executable instructions for a machine.

To clarify this distinction, the organization Programamos offers a useful definition: we can understand CT as a mental skill, while programming is a practical tool that puts that thinking into action.



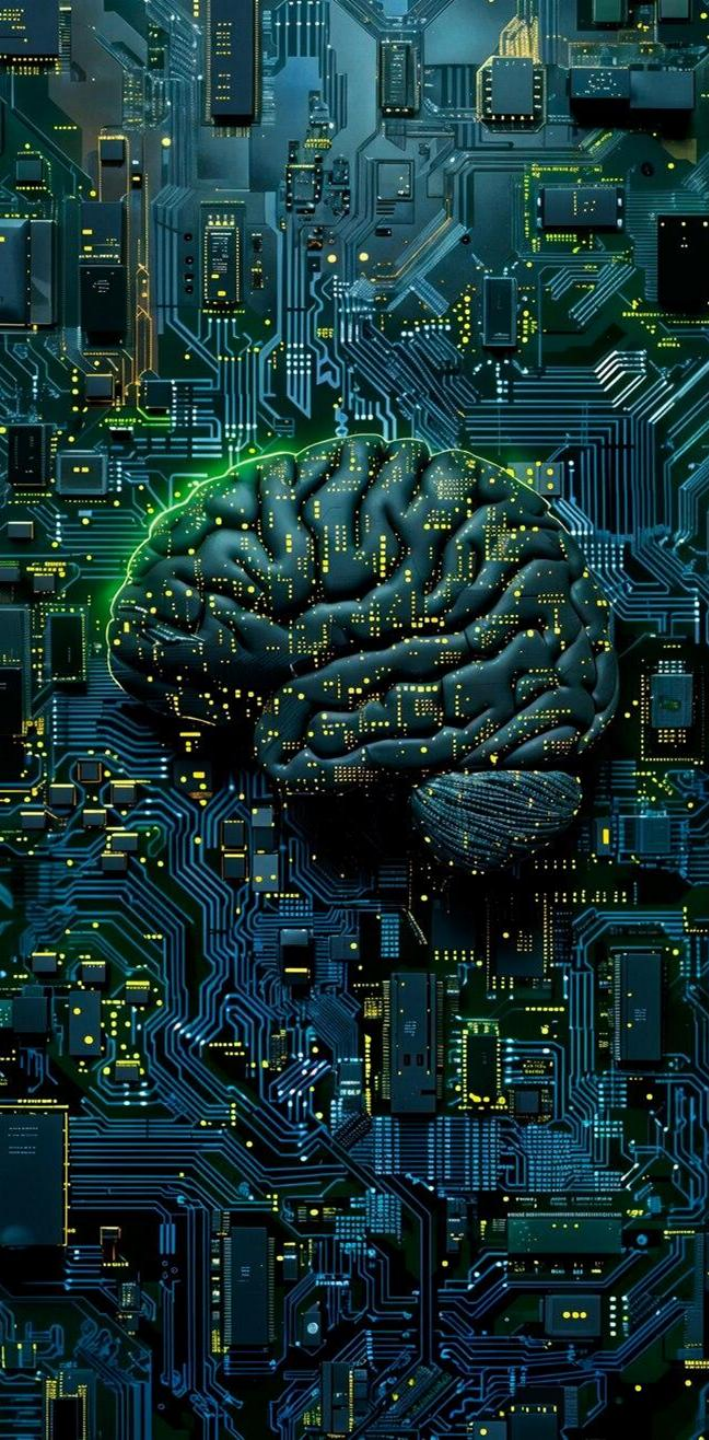
Computational Thinking helps us think clearly.
Programming helps us execute those thoughts.





UNIT 4

*Computational Thinking
core principles*



When we face a challenge or difficulty, two things can happen: Either insecurity takes over and it becomes hard to face the problem, or our understanding of Computational Thinking helps us stay calm and think strategically to find a way through it.

Having this mindset allows us to build confidence and identify the steps we need to tackle challenges more effectively.

Computational Thinking teaches us a way of thinking, a structured approach to facing different problems. But...

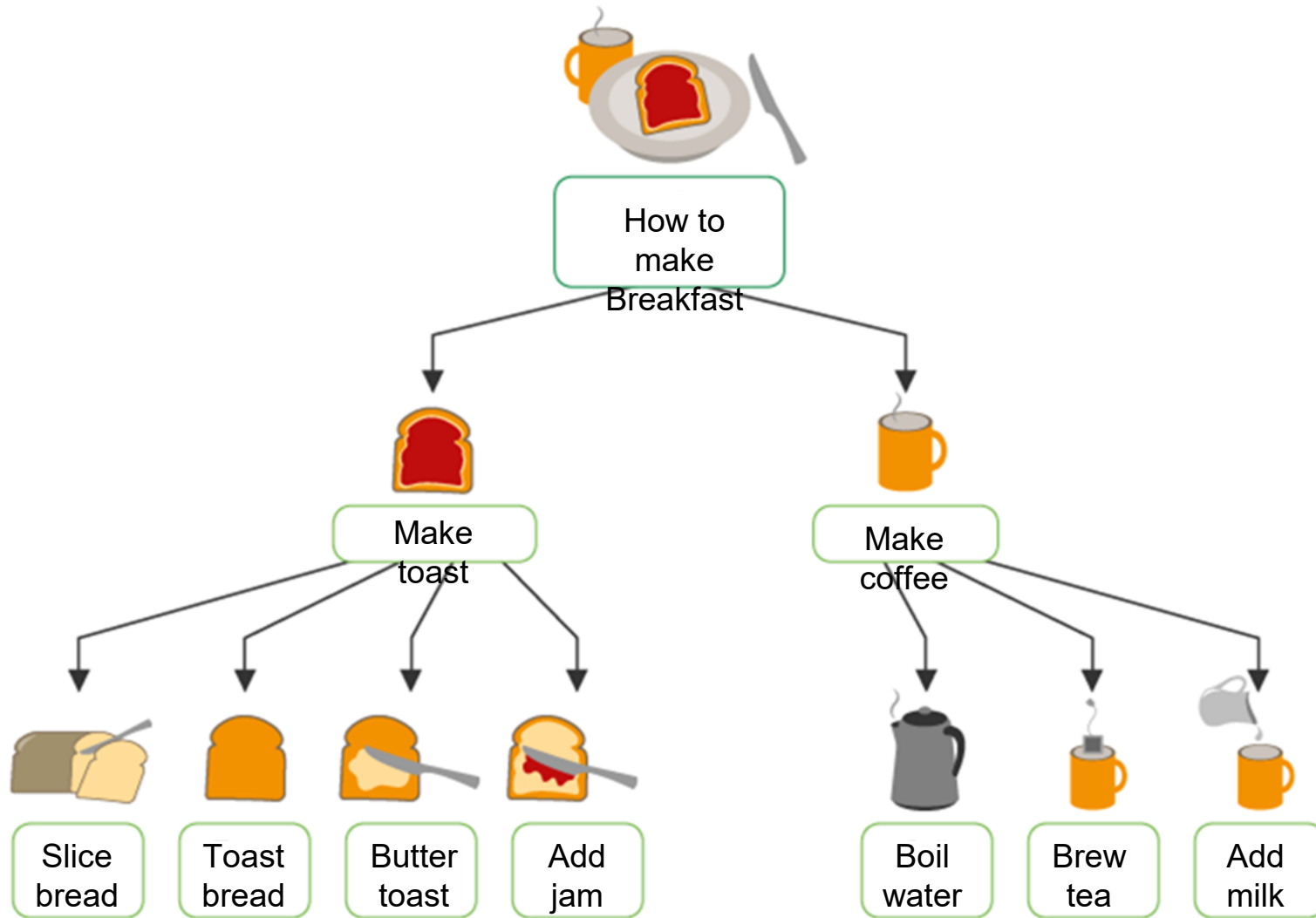
What does it mean to apply Computational Thinking in practice?

We've already defined the concept, now it's time to explore how it works.

Computational Thinking can be broken down into four core components: Decomposition, Pattern Recognition, Abstraction and Algorithms.

In this unit, we'll explore each of these principles to understand how they help us make sense of complex problems and build logical and efficient solutions.





Decomposing the process of making an English Breakfast

Core components:
Decomposition

Decomposition

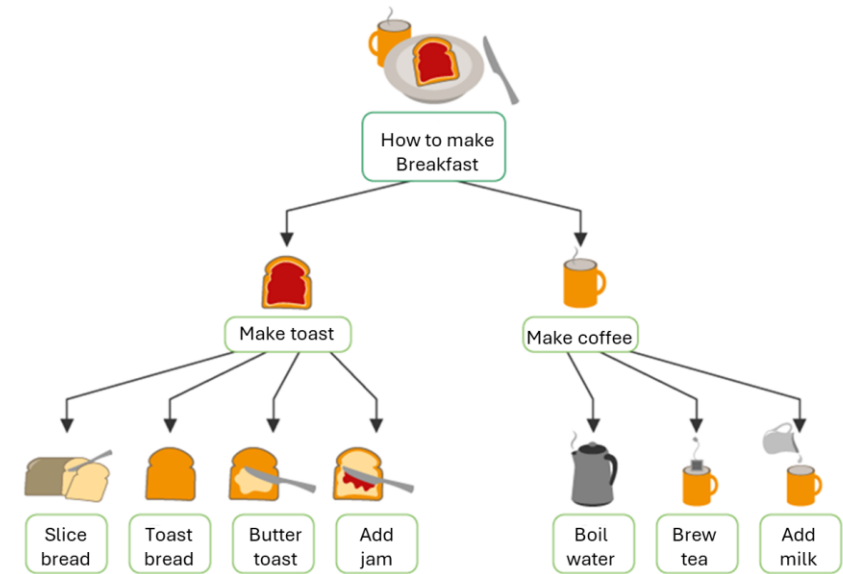
When we're faced with a challenge, it may often feel too complicated to solve all at once. Decomposition aims to break down that big problem into multiple smaller problems.

Decomposition is the process of breaking down a complex problem or system into smaller parts. This allows for easier analysis, solution development and overall understanding of each component before tackling the problem as a whole. It's an essential first step in computational thinking.

In this example, "making breakfast" is divided into simpler subtasks like making toast and brewing tea. Each of those is further broken down. For example: slicing bread, toasting, boiling water, etc.

This approach serves two main purposes:

- ✓ It reduces the sense of being overwhelmed by complexity.
- ✓ It allows responsibilities to be distributed. For instance, one person makes the toast while another prepares the tea.



“Divide and conquer” is a fundamental pillar of Computational Thinking. By solving the smaller problems, we increase the chances of reaching the overall solution more quickly and effectively.



Decomposition

When we decompose a problem, we simplify it to make it easier to solve, but also easier to teach, delegate and adapt. For adult learners especially, this technique helps shift focus from being overwhelmed to identifying a clear starting point.

What trainers should know:

- ✓ Decomposition builds learners' confidence. Small wins matter, achieving smaller subtasks encourages progress and motivation.
- ✓ It's a powerful way to model critical thinking out loud. Talk through how you would break down a problem.
- ✓ In group settings, decomposition is perfect for collaboration as each learner or group can tackle one part and later bring the full solution together.



Trainer activity idea

Choose a daily task (For example, planning a birthday party). Ask learners to work in pairs or small groups and break it down into subtasks. Then ask:

- *Which subtasks are dependent on others?*
- *Can some be done in parallel?*
- *Who would you assign each to?*

Decomposition

Example in real life:

Planning a holiday trip becomes less stressful when split into parts: choosing a destination, setting a budget, booking transport, finding accommodation, creating an itinerary.

Example in business:

Imagine organizing a company training event. Decompose into:

- ✓ Booking the venue
- ✓ Sending invites
- ✓ Collecting feedback
- ✓ Preparing learning materials
- ✓ Arranging catering

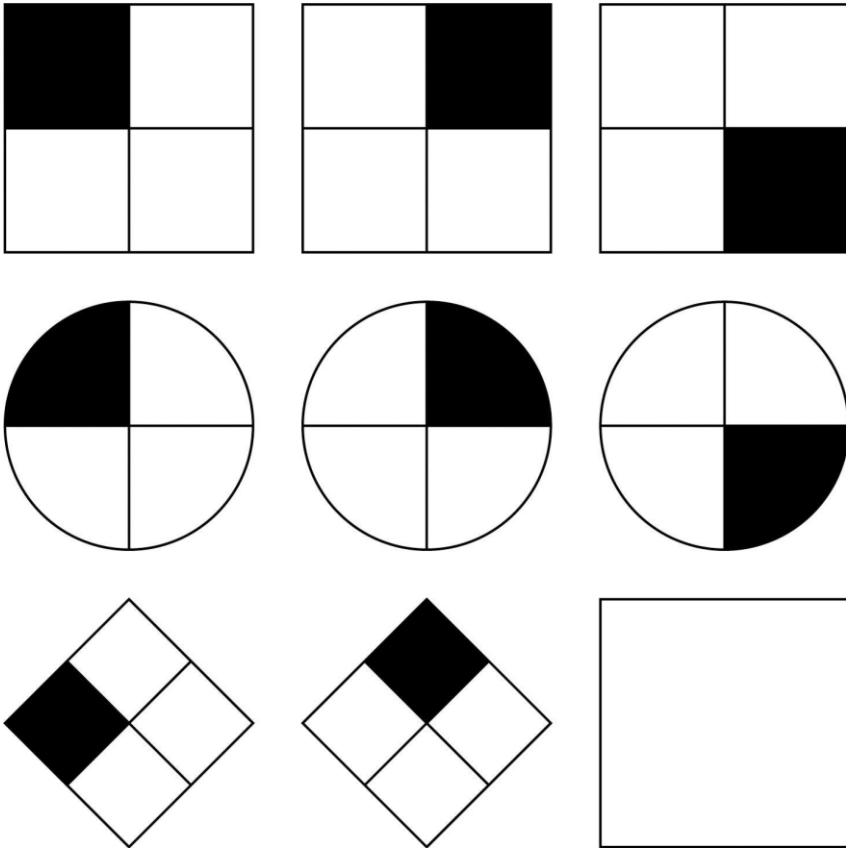
These tasks can be handled by different team members and planned across a timeline.

What trainers can do:

- Highlight real-world value: Learners often engage better when they can see how a skill transfers to their personal or professional lives.
- Encourage learners to reflect: *“What complex task did you face recently? How could you have broken it down?”*
- Visualize decomposition using mind maps, post-its or digital whiteboards.

Use decomposition to design your own sessions. Break your lesson into intro, hands-on, group work and reflection.
Model the technique while teaching it.





***Patterns and
shapes***

Core components:

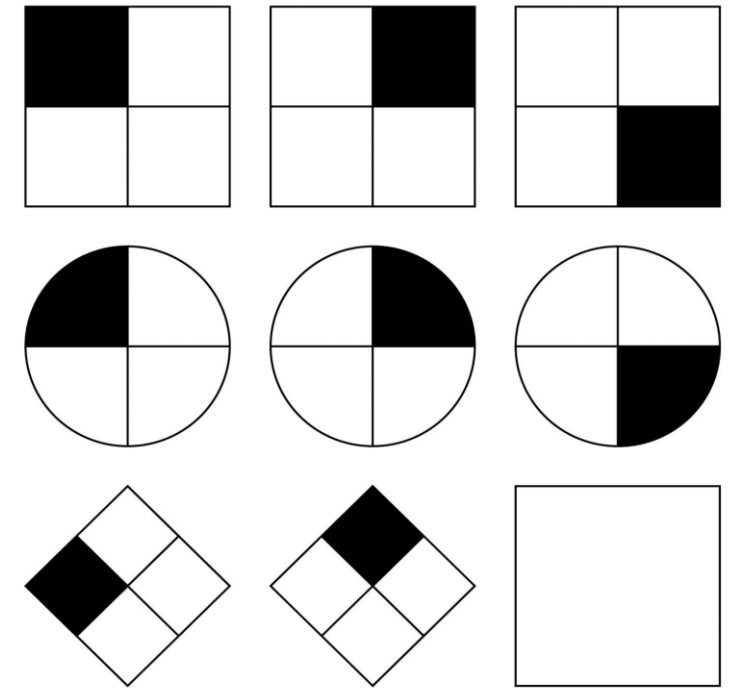
Pattern Recognition

Pattern Recognition

When we try to solve a problem, we aim to find the most effective and productive approach. This is where identifying patterns becomes valuable. Once we've broken down the initial problem, we may notice that certain parts repeat or resemble each other, these are what we call patterns. Sometimes, the situation may even remind us of a previous problem we've already solved. If we can detect these similarities, it becomes much easier to tackle the smaller parts of the problem.

Pattern recognition is the process of identifying trends, similarities, or recurring themes within problems. This makes it possible to simplify the solution process and reuse strategies. It also helps us anticipate outcomes and build efficient plans, making it a core part of Computational Thinking.

In this image, we can observe different shapes that repeat a visual sequence. By analyzing how each figure changes slightly from one to the next, we start to notice patterns. For example, the rotation or placement of the black sections.



Recognizing these visual patterns helps illustrate how our brain detects similarities, which is a key skill when solving problems using Computational Thinking.



Pattern Recognition

When we recognize a pattern, we reduce complexity by reusing what we already know. For adult learners, this is especially powerful as it builds on their experience. Instead of starting from scratch, learners start to say that they've seen this before. This ability is essential in everyday decisions and problem-solving.

What trainers should know:

- ✓ Pattern recognition boosts problem-solving capabilities. Learners don't have to reinvent the wheel as they can spot what has worked before.
- ✓ It's a great opportunity to connect new knowledge to prior experience. You can ask learners to share situations where they recognized patterns before.



Trainer activity idea

Choose a real-life task like going grocery shopping. Ask learners to think about how they usually do it and identify any repeating steps or habits. Then ask:

- *Do you always check your kitchen first?*
- *Do you buy similar things every week?*
- *Can you think of a time when noticing a pattern helped you avoid a problem?*

Pattern Recognition

Example in real life:

Detecting patterns in your daily commute, like when traffic tends to be heavier, helps you adjust departure times.

Example in data science:

Analyzing customer purchase behavior to identify common purchasing patterns, which can inform future marketing campaigns.

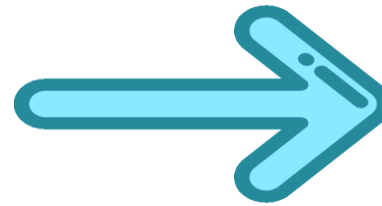
What trainers can do:

- Use familiar examples from learners' lives, like cleaning routines, shopping habits or cooking steps, to spark recognition.
- Ask learners to spot repeated steps and compare them: *“What do you always do first?” “What repeats every time?”*
- Emphasize that finding and reusing patterns saves time and energy.

Spotting patterns help learners solve problems faster. Encourage them to notice repeated routines and show them how using those patterns makes work easier and more efficient.



Core components: Abstraction



Abstracting the detailed view of Warsaw's metro to create a much simpler and easier to read schema

Abstraction

When solving problems, we often need to ignore irrelevant details and focus only on what truly matters. This is where abstraction comes into play.

Abstraction is the process of simplifying a complex problem by focusing on its key features and ignoring unnecessary details. It helps make tasks easier to understand and manage. It means identifying what different elements have in common and setting aside the specifics that vary. This allows us to create general solutions that can be reused across multiple similar problems.

For example, when solving a puzzle, we might start by grouping pieces by color or edge shape, ignoring other features. This abstraction makes it easier to begin solving the problem. Later, we can look at more specific details to complete the picture.

This simplified metro map image on the right is an abstracted version of the original. It removes unnecessary details like street names or geographic accuracy and keeps only the essential elements, which are stations and connections, making it easier to understand how to get from one place to another.



Abstraction means focusing on what matters most. Like a simplified metro map, it helps learners navigate problems by highlighting the essentials and ignoring distractions.



Abstraction

When we teach abstraction, we're helping students focus on what really matters. This skill is valuable for adults as it helps reduce overwhelm and improves decision-making. Eliminating distractions and identifying key elements makes it easier to understand problems and find practical solutions.

What trainers should know:

- ✓ Abstraction helps learners focus on what's essential and ignore distractions. This is especially helpful for adults who may get overwhelmed by too much information.
- ✓ Teaching abstraction encourages learners to simplify problems and build solutions step by step, without needing to know every single detail from the start.
- ✓ It's useful for developing problem-solving strategies that apply to multiple situations, saving time and effort.



Trainer activity idea

Use the daily routine of getting ready to leave the house for work or school. Walk learners through each step, then guide them to identify what truly matters. Ask:

- *What are the essential steps you always do before leaving?*
- *What things could you skip and still be ready?*
- *How does focusing only on the key steps help save time or reduce stress?*

Abstraction

Example in real life:

When designing a house, you focus on the number of rooms and layout rather than getting bogged down by minor details like paint colors or furniture.

Example in software development:

When designing a website, abstraction involves focusing on core functionality (e.g., navigation, user login) before thinking about visual aesthetics

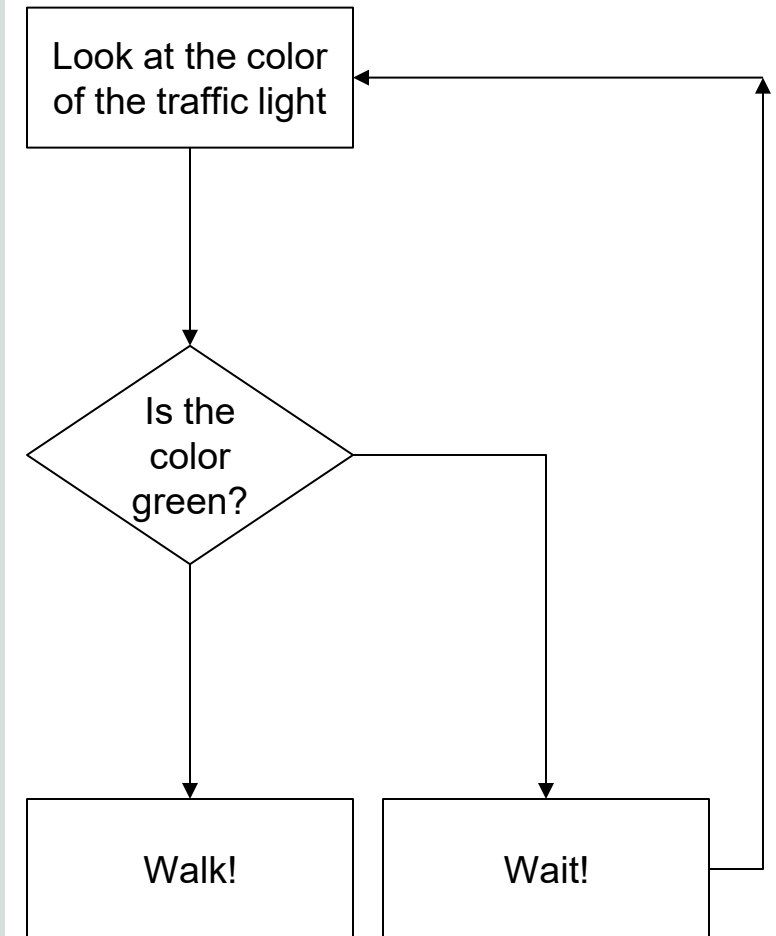
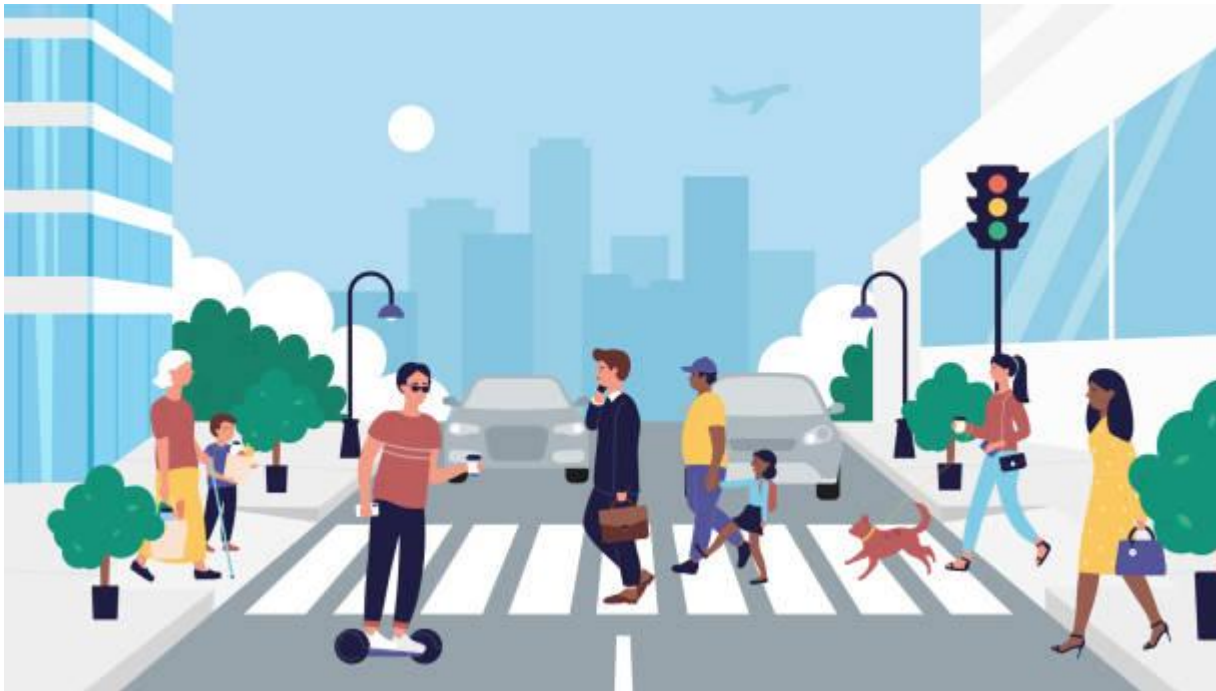
What trainers can do:

- Encourage simplifying instructions: *What do others need to know first? What can be left out without changing the goal?*
- Use role-play to compare overly detailed vs. clear communication and reflect on which is easier to follow.
- Help learners filter information by asking: *What's the most important thing to focus on here?* This encourages prioritizing key ideas.

Use abstraction to focus your learners' attention on what really matters. Reducing unnecessary detail improves clarity and communication.



Core components: Algorithms



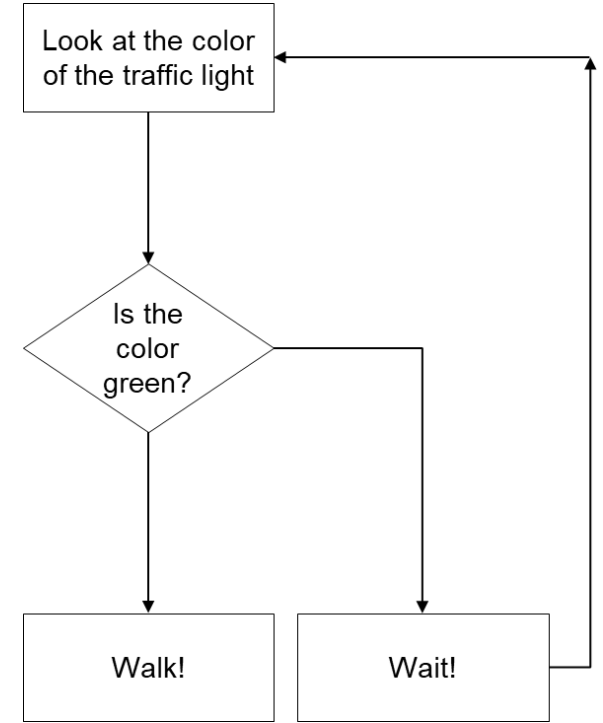
A simple algorithm for pedestrians to cross the street

Algorithms

When we face a complex task, it's not just about doing things, but doing them in the right order. That's where algorithm design comes in. Algorithms give us a clear, step-by-step roadmap to follow, which helps us plan ahead and avoid getting lost in the process.

Algorithm design is the creation of a step-by-step process or set of rules to solve a problem or complete a task efficiently and accurately. It involves breaking down a problem into smaller, manageable steps to ensure optimal solutions and clear execution.

This flowchart is a simple algorithm for crossing the street. It checks the traffic light color and gives a clear instruction: "Walk" if it's green, "Wait" if not. The steps repeat until it's safe to go. It shows how algorithms use logical steps to guide actions.



Algorithm design is like writing down a recipe for solving a problem: easy to follow, step by step, so anyone can repeat the process and get the same result.



Algorithms

Algorithm design helps learners approach tasks logically, knowing what comes next and how to reach a solution. When they know the steps, they feel more confidence while reaching the results.

What trainers should know:

- ✓ Algorithm design promotes structured thinking. Learners become more precise and organized in their actions.
- ✓ It helps develop planning skills, especially for tasks that require sequence and accuracy.
- ✓ Algorithms make it easier for learners to explain their process and reflect on what worked (or didn't).



Trainer activity idea

Ask learners to describe how they make a cup of coffee or tea. Have them write or say each step in order, as if explaining it to someone who has never done it before. Then ask:

- *Are the steps clear and in the right order?*
- *Would someone be able to follow your instructions exactly?*
- *Is there anything missing that would cause confusion?*

Algorithms

Example in real life:

A cooking recipe is an algorithm: it provides step-by-step instructions to achieve a specific result (the meal).

Example in technology:

Search engines use algorithms to fetch relevant results based on the keywords you input.

What trainers can do:

- Break down everyday routines into ordered steps and ask: *What happens if one step is skipped or done out of order?*
- Ask learners to write instructions for a simple task and test if others can follow them. Then ask: *What was missing or unclear?*
- Use flowcharts to visualize decision-making processes, then ask: *Where do choices affect the outcome?*

Teach your students to think in steps. Clear instructions help them work more efficiently and solve problems without getting stuck.



A woman with short brown hair and black-rimmed glasses is seated at a white table, focused on writing on a tablet with a white pen. She is wearing a light-colored cardigan over an orange top. In the background, a man in a white shirt is also seated at a table, looking down at his work. The setting appears to be a bright, modern educational or professional workspace with bookshelves visible in the distance.

UNIT 5

*Computational Thinking in
the European educational
structure*

Current landscape of CT in Europe

Computational Thinking has gained prominence in recent years in European educational discussions for its potential to develop important skills like critical thinking and problem-solving. However, its application has been concentrated in compulsory education at the primary and secondary levels, leaving adult education out of the discussion.

Since 2018, the European Commission has acknowledged the importance of Computational Thinking as a core component of 21st-century digital competence. Through the Digital Education Action Plan (2021–2027), the European Commission has actively promoted the integration of CT into education systems to better prepare citizens for digital transformation.

In most European countries, there is still no clear strategy for incorporating CT into adult training programs, even though it is a key competence for employability and digital citizenship.



Which European countries are leading in Computational Thinking?



Estonia has been a pioneer in the region, integrating CT into the school system since 2012 and expanding initiatives into vocational training.

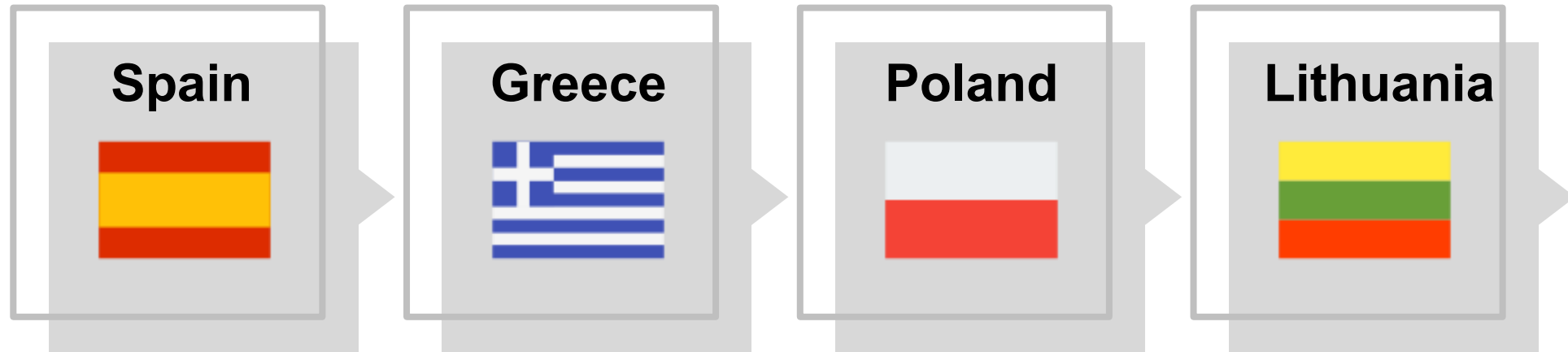
Finland, incorporated CT into the national curriculum in 2016 and has also launched pilot projects in adult and continuing education.

The United Kingdom made it mandatory in basic education in 2014 and has also promoted programs aimed at improving digital and computational skills among adults actively seeking employment.

France has implemented CT at the secondary education level since 2016 and has launched public digital training initiatives targeting workers and unemployed individuals.

Germany is integrating CT into technical and vocational education, with a particular focus on digital transformation within professional training programs.

Countries developing CT: Where are the first steps being taken?



Spain has started to integrate CT into the educational plans of some autonomous communities, especially at the compulsory education levels. On adult education there are isolated initiatives promoted by universities and community centers but no consolidated national policy yet.

Greece does not yet have a structured national program for CT, although academic institutions and NGOs have begun developing training experiences for adults, focusing on programming, educational robotics and logical thinking.

Poland has included Computational Thinking in the school curriculum, but its application for adults is mainly limited to vocational training projects or local initiatives in digital inclusion centers.

Lithuania, although increasingly focused on technology in its education system, is still taking its first steps to incorporate CT into non-formal or continuing education for adults.

The background image shows a woman in the foreground, wearing glasses and a light-colored cardigan over an orange top, focused on writing on a tablet. In the background, a man in a white shirt is also working at a desk. The scene is set in a bright, modern office or library with bookshelves visible in the distance.

UNIT 6

*Case study
and activities*

Case studies & good practices

Case study: smart traffic control system with Computational Thinking

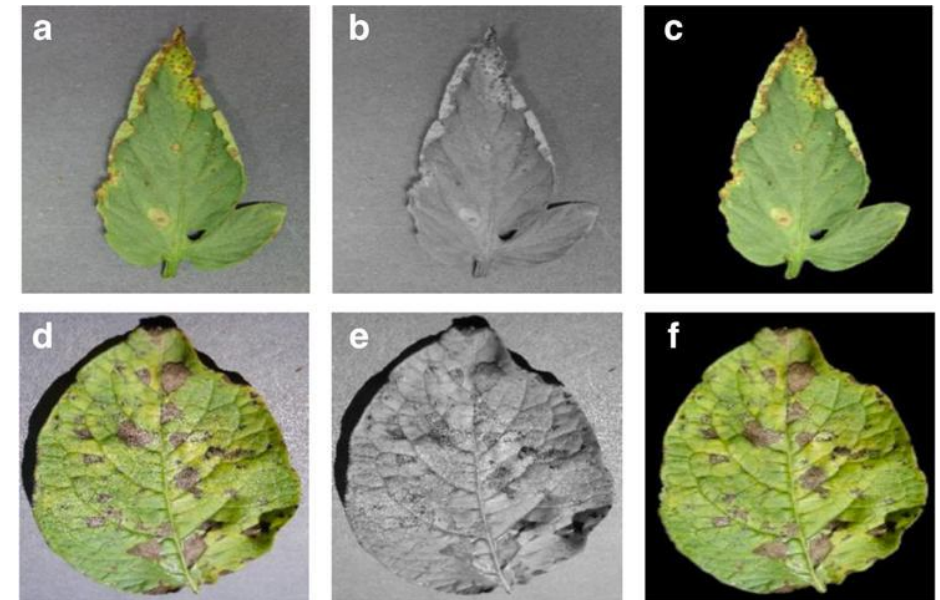
- **Example:**
A smart system was developed to ease traffic congestion on Tai Tam Road, Hong Kong.
- **Practice:**
Using computational thinking, the system analyzed real-time traffic patterns through video analytics, adjusting traffic light timings based on vehicle queue lengths, resulting in double time savings.
- **Good practice:**
The system applied **pattern recognition** and **abstraction** to identify congestion trends and adjust green light durations, reducing delays and improving traffic flow by automating key decisions.



Case studies & good practices

Case study: plant disease detection with Computational Thinking

- **Example:**
A system was developed to detect plant diseases using image processing and machine learning techniques.
- **Practice:** The system used **abstraction** to focus on key disease patterns, and **automation** allowed for faster, more accurate disease detection using Support Vector Machines (SVM) combined with Active Contour edge detection
- **Good practice:**
Using computational thinking, the process was broken down into edge detection and classification, applying **pattern recognition** for image analysis and **decomposition** for disease identification.



Activity 1 – Decomposition

Break a complex task into smaller, manageable parts.

- **Example:** Planning a trip to Barcelona (choosing a destination, booking flights, making an itinerary).
- **Objective:** Understand how breaking down a problem simplifies its solution



Activity 2 – Pattern Recognition

Analyze the following number sequence:

- Sequence A: 2, 4, 7, 11, 16, ...
 - Sequence B: 2, 4, 8, 16, 32, ...
 - Sequence C: 1, 1, 2, 3, 5, 8, 13...
-
- **Identify** the pattern in how the numbers change.



Activity 3 – Abstraction

Summarize a movie plot

- Pick any movie you know well (e.g., a superhero movie).
- **Abstract** the plot by focusing on the core elements:
 - Main character's goal
 - Key conflicts
 - Resolution



Activity 4 – Algorithm Design

Develop step-by-step instructions to solve a task.

- **Example:** Create an algorithm to make a sandwich(pick ingredients, slice the bread, etc.)
- **Objective:** Illustrate how precise steps can solve problems systematically



SUMMARY

Computational Thinking is a structured way of solving problems that draws from core concepts in computer science but applies far beyond it. In this module, we explored what CT means, its origins, and why it matters, especially for adult learners.

We examined how CT helps us break down complex challenges, recognize patterns, focus on essential information and design logical, step-by-step solutions. By comparing how computers and humans think, we gained insight into CT as a bridge between logic and creativity.

CT is not just about programming, it's a mindset. For adult education, it offers practical value: supporting autonomy, critical thinking, and decision-making across everyday life, work, and learning. As trainers, understanding CT empowers us to help learners build confidence and adapt to a fast-changing world.



CALL TO ACTION:

Reflect on what you've learned:

- *How can you help learners use decomposition, patterns, abstraction, and algorithms in their daily lives?*
- *What real-world activities could you adapt to introduce these CT concepts with low-qualified adult learners?*
- *How will you model computational thinking in your own teaching to help learners see its value?*

GLOSSARY

Computational Thinking or CT: Solving problems like a computer would, step-by-step.

Decomposition: Breaking a big problem into smaller parts.

Abstraction: Focusing only on the important details.

Pattern Recognition: Spotting trends or things that repeat.

Algorithm: A set of instructions to complete a task.

Iteration: Repeating a process to improve it.

Unplugged Activities: Learning CT without screens using games, puzzles, etc.

Debugging: Finding and fixing errors in a process.

Soft Skills: Non-technical abilities that help people work well with others and adapt to challenges.

Gamification: Using game elements (like points or challenges) in learning.

Digital Literacy: Knowing how to use digital tools safely and effectively.

Inclusion: Making learning accessible to everyone, no matter their background.

Scaffolding: Supporting learners step-by-step so they can gradually do more on their own.

References

- Aho, A. V. (2011). Computation and computational thinking. Ubiquity, 2011(January), 1–1. <https://dl.acm.org/doi/10.1145/1922681.1922682>
- Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. In Proceedings of the 16th Koli Calling Conference on Computing Education Research (pp. 120–129). ACM. <https://dl.acm.org/doi/10.1145/2999541.2999542>
- Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717–3725. <http://denninginstitute.com/pjd/PUBS/long-quest-ct.pdf>
- Smart City Consortium. (n.d.). Case studies. Logistics and Supply Chain MultiTech R&D Centre. <https://www.lscm.hk/eng/channel.php?channel=case-stcs>
- Smart City Blueprint for Hong Kong. (n.d.). Smart City initiatives. <https://www.smartcity.gov.hk/>
- Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43. <https://journals.sagepub.com/doi/10.3102/0013189X12463051>
- Bocconi, S., Chiocciariello, A., Dettori, G., & Kampylis, P. (2016). Developing computational thinking in compulsory education: Implications for policy and practice. Joint Research Centre – European Commission. <https://op.europa.eu/en/publication-detail/-/publication/093eadcc-c820-11e6-a6db-01aa75ed71a1/language-en>
- European Commission. (2021). Digital Education Action Plan 2021–2027: Resetting education and training for the digital age. <https://education.ec.europa.eu/focus-topics/digital/digital-education-action-plan>
- Computational Thinking. Resources and strategies for teaching computational thinking. <https://computationalthinking.org/>