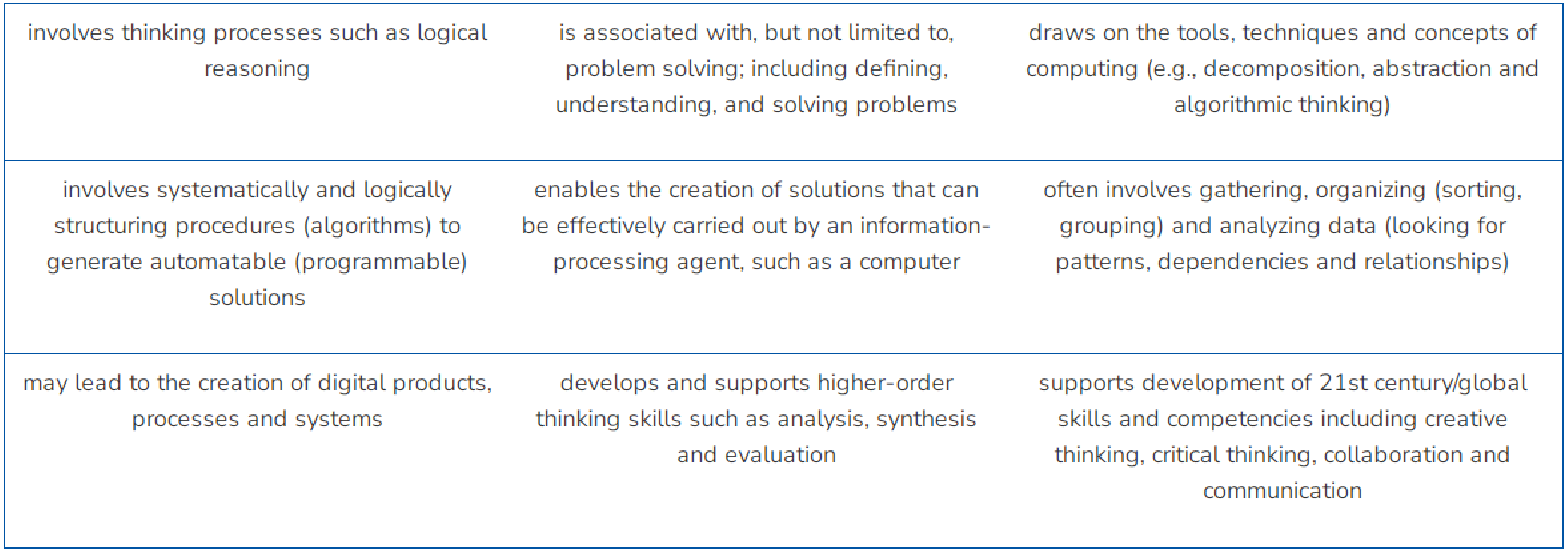
LOGICAL AND ALGORITHMIC THINKING

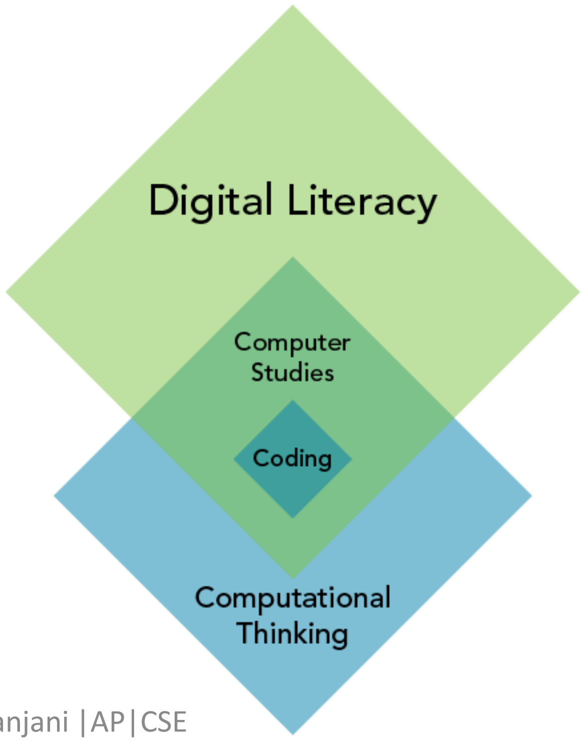
Computational Thinking…



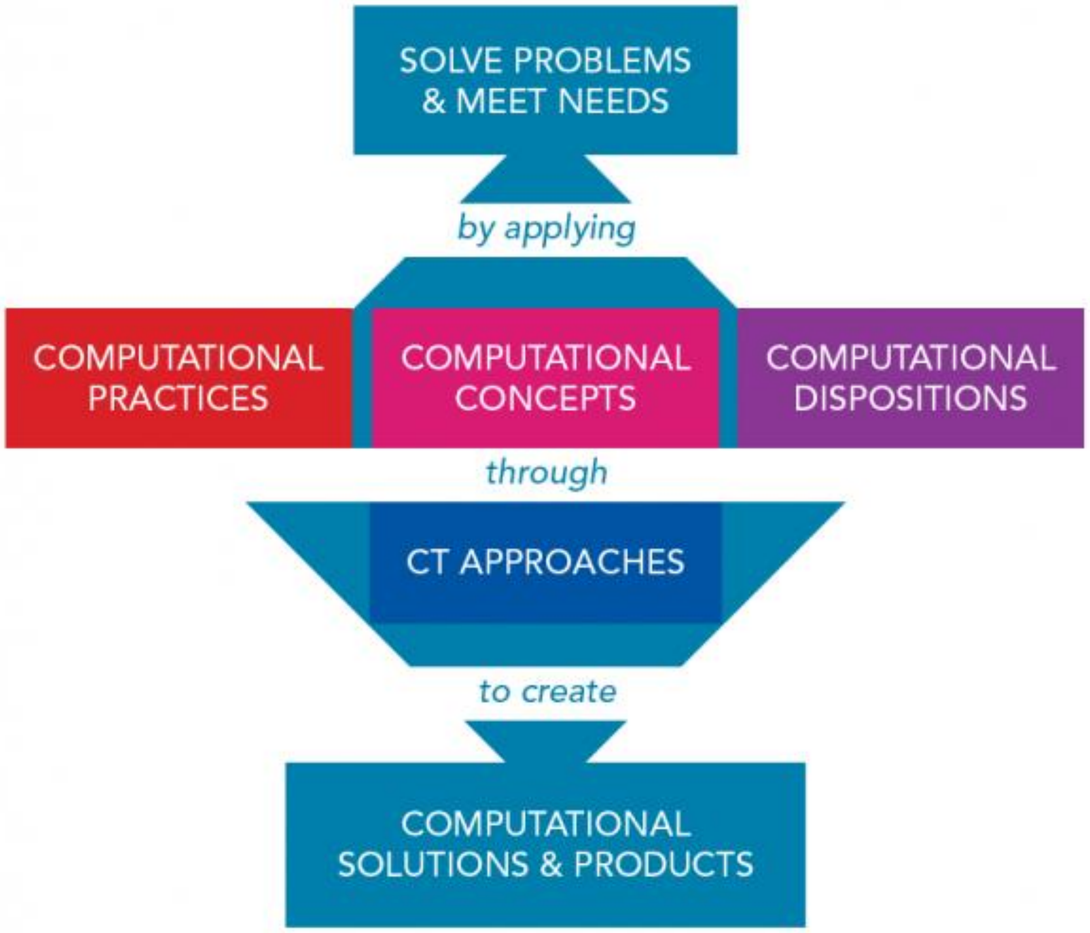
involves thinking processes such as logical reasoning	is associated with, but not limited to, problem solving; including defining, understanding, and solving problems	draws on the tools, techniques and concepts of computing (e.g., decomposition, abstraction and algorithmic thinking)
involves systematically and logically structuring procedures (algorithms) to generate automatable (programmable) solutions	enables the creation of solutions that can be effectively carried out by an information-processing agent, such as a computer	often involves gathering, organizing (sorting, grouping) and analyzing data (looking for patterns, dependencies and relationships)
may lead to the creation of digital products, processes and systems	develops and supports higher-order thinking skills such as analysis, synthesis and evaluation	supports development of 21st century/global skills and competencies including creative thinking, critical thinking, collaboration and communication

WHERE DOES COMPUTATIONAL THINKING FIT?

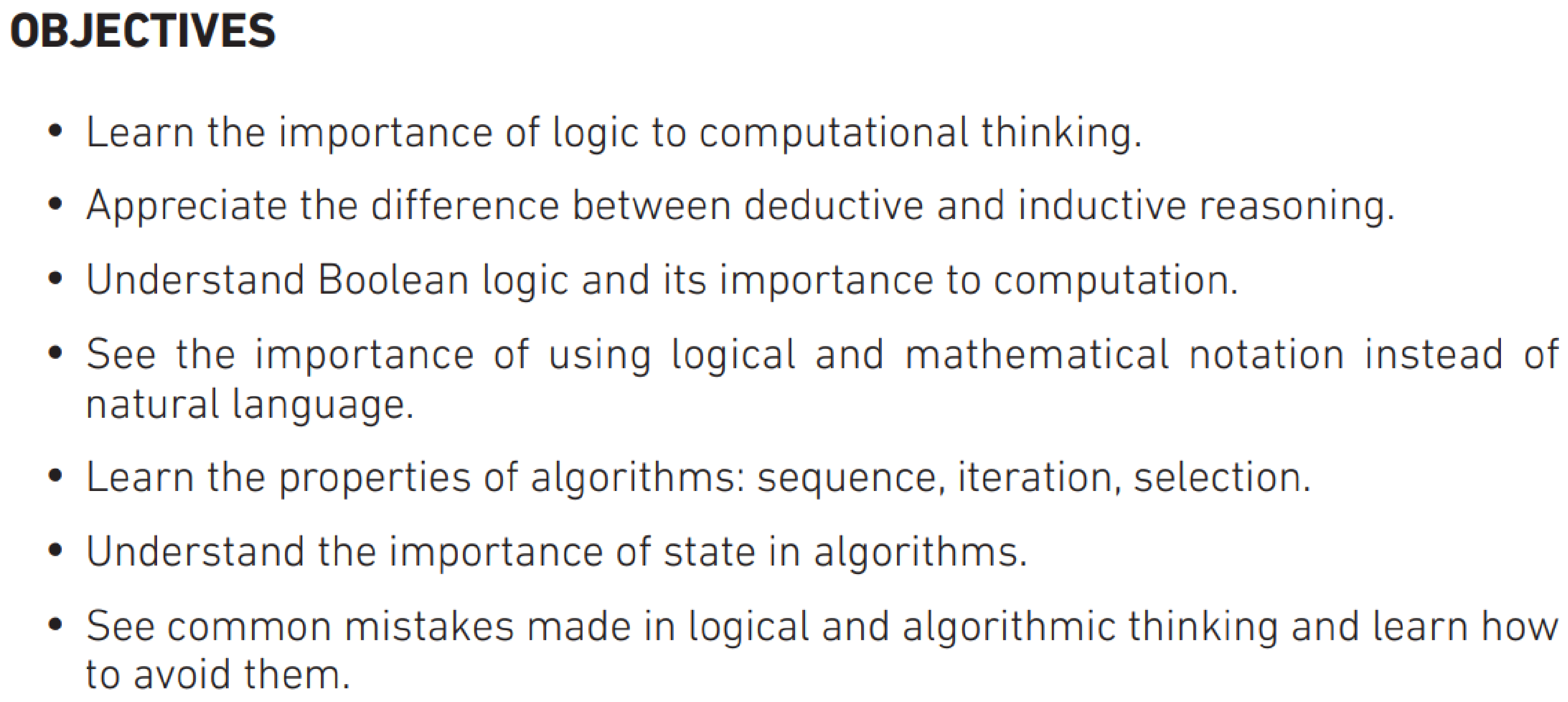
- CT falls within **Digital Literacy** (DL)
- DL is "the ability to locate, organize, understand, evaluate, and create information using digital technology for a knowledge-based society".



COMPUTATIONAL THINKING FRAMEWORK


SOLVE PROBLEMS & MEET NEEDS
by applying
COMPUTATIONAL PRACTICES
COMPUTATIONAL CONCEPTS
COMPUTATIONAL DISPOSITIONS
through
CT APPROACHES
to create
COMPUTATIONAL SOLUTIONS & PRODUCTS

LOGICAL AND ALGORITHMIC THINKING



OBJECTIVES

- Learn the importance of logic to computational thinking.
- Appreciate the difference between deductive and inductive reasoning.
- Understand Boolean logic and its importance to computation.
- See the importance of using logical and mathematical notation instead of natural language.
- Learn the properties of algorithms: sequence, iteration, selection.
- Understand the importance of state in algorithms.
- See common mistakes made in logical and algorithmic thinking and learn how to avoid them.

LOGICAL THINKING

How important logic is to computational thinking????????????????

It introduces logical reasoning, along with Boolean and symbolic logic, and shows how to turn intuitive ideas into mathematically sound, logical expressions.

Logic

- logic is a system used for distinguishing between correct and incorrect arguments.
- The philosophical idea of an argument; namely a chain of reasoning that ends up in a conclusion.
- Logic includes a set of principles that, when applied to arguments, allow us to demonstrate what is true.

Importance of Logic in Computer Problem-Solving

- **Automating Reasoning**: Computers are used to automate logical reasoning processes.
- **Mastering Logic**: Understanding and applying logic is essential before writing a computer solution.
- **Developing Hypotheses**: Logic helps in forming and testing hypotheses.
- **Building on Knowledge**: Logic allows you to use known facts to reach new conclusions.

Premises in Logical Arguments

- **Premise Definition**: A premise is a statement that can be evaluated as 'true' or 'false.'
- **Truth Value**: Each premise has a truth value.
- **Examples**:
  - "All men are mortal" (true/false)
  - "Socrates is a man" (true/false)
- **Non-Premises**: Questions or commands cannot be premises (e.g., "What time is it?" or "Break a leg!").

Conclusion from Premises

- **Analysis Step**: After stating premises, the next step is to analyze them.
- **Drawing Conclusions**: This is where logical reasoning is applied to form conclusions.

Inductive vs deductive arguments

Strength of Logical Arguments

- **Argument Strength**: Not all logical arguments are equally strong.
- **Categories of Arguments**: Arguments can be classified based on their certainty.

Two Main Categories

- **Deductive Arguments**: Provide absolute certainty if the premises are true.
- **Inductive Arguments**: Provide probable conclusions based on evidence.

Deductive Arguments

- **Definition**: A deductive argument is one where the conclusion logically follows from the premises. If the premises are true, the conclusion must also be true.
- **Process**: Starts with a general rule or theory and applies it to a specific case.
- **Certainty**: Provides absolute certainty if the premises are correct.
- **Example**:
  - **Premise**: All humans need oxygen to live.
  - **Premise**: Sarah is a human.
  - **Conclusion**: Therefore, Sarah needs oxygen to live.
- In **computational thinking**, deductive reasoning is used in algorithms where general rules (like mathematical or logical principles) are applied to specific data sets to derive correct and certain outputs.

Inductive Arguments

- **Definition**: An inductive argument makes generalizations based on observations or specific instances. The conclusion is probable but not guaranteed.
- **Process**: Begins with specific cases or data and develops a general rule.
- **Certainty**: The conclusion is probable, but there is a possibility it could be wrong.
- **Example**:
  - **Premise**: Every time I use this program, it crashes after 10 minutes.
  - **Conclusion**: This program always crashes after 10 minutes.
- In **computational thinking**, inductive reasoning is used in areas like machine learning, where patterns are detected in specific data, and these patterns are generalized to make predictions or develop models.

Key Differences

- **Deductive reasoning** is useful for scenarios where a guaranteed outcome is necessary (like mathematical proofs or deterministic algorithms).
- **Inductive reasoning** is more applicable to areas where patterns and trends are drawn from data, such as artificial intelligence, data mining, or hypothesis generation.

- Deductive arguments are strong, they have very strict standards, which makes them hard to construct.
- A deductive argument can fail in one of two ways.
- First, one of its premises could turn out to be false.
- For example:
- 1. Missie is a dog.
- 2. All dogs are brown.
- 3. Therefore, Missie is brown.
- Premise 2 is false: not all dogs are brown.
- Any argument with false premises fails

- The second way a deductive argument fails is when the conclusion doesn’t necessarily follow from the premises.
- For example:

1. All tennis balls are round.
2. The Earth is round.
3. Therefore, the Earth is a tennis ball.

- This argument fails because of faulty logic.

Real-World Use of Deductive Arguments

- **Rarity**: Deductive arguments are uncommon in real life, where knowledge is often incomplete or messy.
- **Inductive Reasoning**:
  - Deals with probabilities, not absolutes.
  - More practical for real-life situations where issues are nuanced.

Inductive Argument Example

1. A bag contains 99 red balls and 1 black ball.
2. 100 people each drew a ball.
3. Sarah is one of them.
4. **Conclusion**: Sarah **probably** drew a red ball.
    1. Emphasizes **probability**, not certainty.

Importance in Computational Thinking (CT)

- The computer’s answer is only as good as its reasoning.
- **Your Responsibility**:
  - Ensure the reasoning is valid.
  - Provide reliable input.
  - Interpret the computer's result: Is it **unquestionably true** (deductive) or **probably true** (inductive)?

- Even though much of our reasoning is inductive, computers are not well equipped to deal with shades of grey.
- Their binary nature makes them more apt to deal with black and white issues.
- In order to instruct computers to make logical decisions, we need a system of logic that maps well onto this way of thinking.

Boolean logic

- Boolean logic is such a method.
- It’s a form of logic that deals with statements having one of only two values: true or false (usually).
- Different corresponding values could be used in other contexts: 1 or 0 for example, on or off, black or white

Propositions

- Statements in Boolean logic are also known as propositions, which have several basic properties.

1. First, a proposition can only have one value at any one time.

- In other words, a single proposition can’t be both true and false simultaneously.
- There is no way to express levels of certainty. True means true; false means false.

2.propositions must have clear and unambiguous meaning.

- For example, a statement like:

‘It is travelling fast’, can certainly be evaluated as either true or false.

However, it’s ambiguous as stated.

If ‘it’ is a car travelling at 150 mph along the motorway, that’s certainly fast.

But if ‘it’ is a spacecraft travelling towards Mars at 150mph, that’s undoubtedly slow.

Qualify statements where necessary to avoid ambiguity. For example, you could restate the example above as: ‘It is travelling fast, where “fast” is 70mph or greater.

- **Ambiguity:** The statement "The meal is spicy" might be interpreted differently depending on individual tolerance to spice. One person might find a dish mildly spicy, while another might find it very hot.
- **Qualified Statement:** To clarify, you could restate the scenario as: "The meal is spicy, where 'spicy' means it has a heat level of 5 or higher on a scale from 1 to 10."

3.it's possible to combine individual propositions to make more complex ones (called compound propositions).

We make compound propositions by connecting single propositions together using logical operators.

- Compound propositions are created by combining simpler, individual propositions using logical operators such as AND, OR, and NOT.
- This allows us to evaluate more complex statements. Here's a simple example:

Logical operators

1. AND OPERATOR

- **Scenario: "The movie is entertaining and the theater is comfortable."**
- **Individual Propositions:**

1. The movie is entertaining.
2. The theater is comfortable.

- **Compound Proposition:** "The movie is entertaining and the theater is comfortable."
- **Logical Operator:** AND
- **Explanation:** This compound proposition is true if and only if both individual propositions are true. If either one of them is false, the entire compound proposition is false.

- **Proposition 1:** "The coffee is hot."
- **Proposition 2:** "The coffee is strong."
- **Compound Proposition:** "The coffee is hot and the coffee is strong."
- In this case, the compound proposition will only be true if both propositions about the coffee being hot and strong are true.

- **Statement:** "If the weather is sunny and I’m on holiday, then I’m going to lie in the garden."
- **Individual Propositions:**

1. The weather is sunny.
2. You’re on holiday.

- **Logical Operator:** AND
- **Compound Proposition:** "If the weather is sunny AND I’m on holiday, THEN I’m going to lie in the garden."
- **Explanation:**
- **True:** You’ll lie in the garden if both propositions are true. That means:
  - The weather is indeed sunny.
  - You are on holiday.
- If both conditions are satisfied, then you will lie in the garden.
- **False:** If either of the propositions is false, the entire statement becomes false. That means:
  - If the weather is not sunny (e.g., it's rainy), or
  - If you are not on holiday (e.g., you have to work),
- Then, you won’t be lying in the garden, regardless of the other condition.

Logical operators

- In logical reasoning, the conjunction operator (AND) is used to link multiple propositions such that all of them must be true for the overall statement to be true. Here’s a quick summary:
- **Conjunction (AND):** Connects propositions so that the compound statement is true only if all connected propositions are true.

- **Example:**
- **Propositions:**
  - "It is raining."
  - "I have an umbrella."
- **Compound Proposition:** "It is raining AND I have an umbrella."

Form Truth table for the above scenario

Truth Table:

It is raining	I have an umbrella	Compound Statement
True	True	True
True	False	False
False	True	False
False	False	False

- If any of the propositions is false, the compound statement using AND will also be false.
- Explicitly stating AND in logical arguments helps clarify the required conditions and ensures precision in communication and reasoning.

OR

- OR: the technical name for this operator is disjunction.
- This operator chains propositions together in a way that at least one of them must be true for the conclusion to be true also.
- The only way that the conclusion is falsified is if all propositions are false.

- **Example:**
  "If it is raining **or** it is snowing, then we will cancel the picnic."
  - The picnic will be canceled if **either** condition is true (if it is raining **or** if it is snowing).
  - Even if both conditions are true (if it is raining **and** snowing), the picnic will still be canceled.

- NOT: the technical name for this operator is negation. This operator doesn’t chain propositions together itself, rather it modifies a single proposition. Specifically, it flips the truth value. Sometimes, negating a proposition can make it easier to express the chain of reasoning.

Scenario: Deciding to Go for a Walk

Description:

- **Condition:** You decide to go for a walk if it is **not** raining.
- **Logical Expression:** NOT (It is raining)

Visual Aid:

Decision Tree Diagram:

1. **Question:** "Is it raining?"
    - **Yes** → NOT (True) → **Don't go for a walk**
    - **No** → NOT (False) → **Go for a walk**

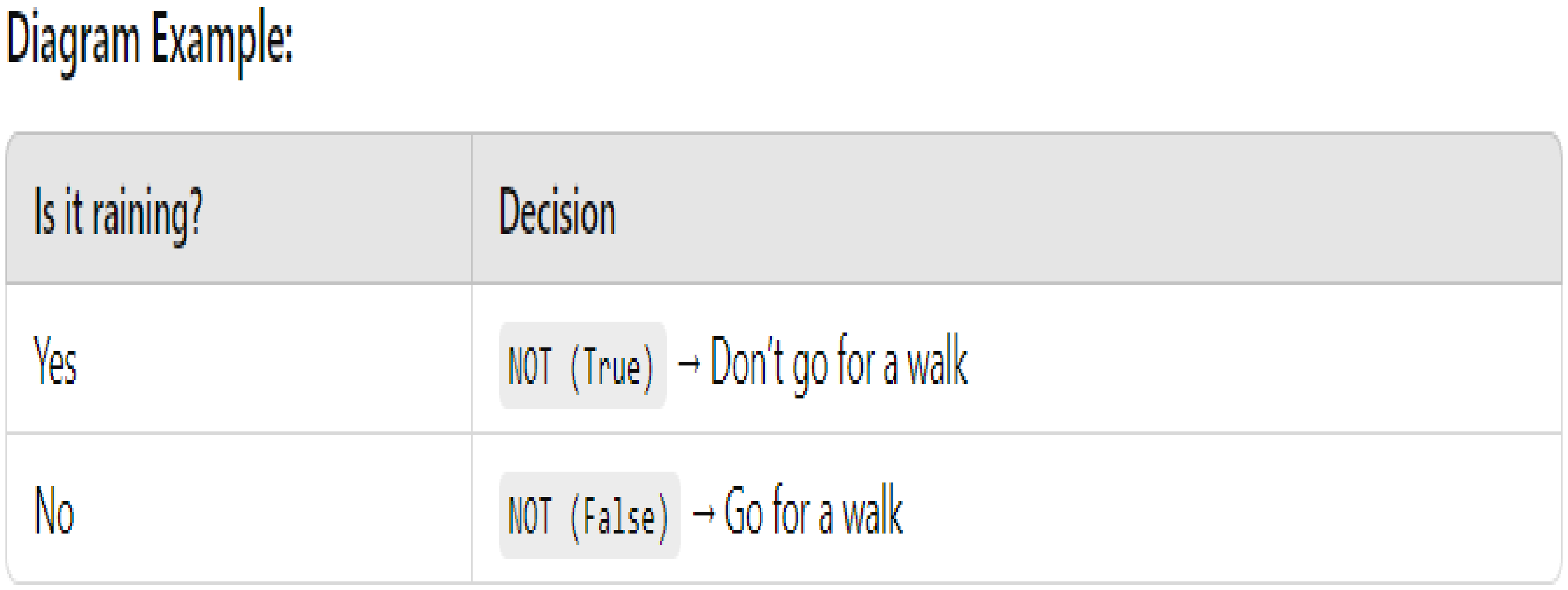


Diagram Example:

Is it raining?	Decision
Yes	`NOT (True)` → Don't go for a walk
No	`NOT (False)` → Go for a walk

The NOT operator inverts the result of the condition. If the condition is true, NOT makes it false, and vice versa.

- IMPLIES: the technical name for this operator is implication.
- Using this operator is to state that there is a correlation between the two statements.
- If the first statement is true, then the second must be true also.
- Keep in mind, this is a correlation not a causation.

- **Scenario:** Homework and Watching TV
- **Statement:**
- **If you finish your homework, then you can watch TV.**
- **Explanation:**
- This implies that finishing homework leads to watching TV.
- However, it does not mean that if you're watching TV, you must have finished your homework. You could be watching TV for other reasons (like free time).

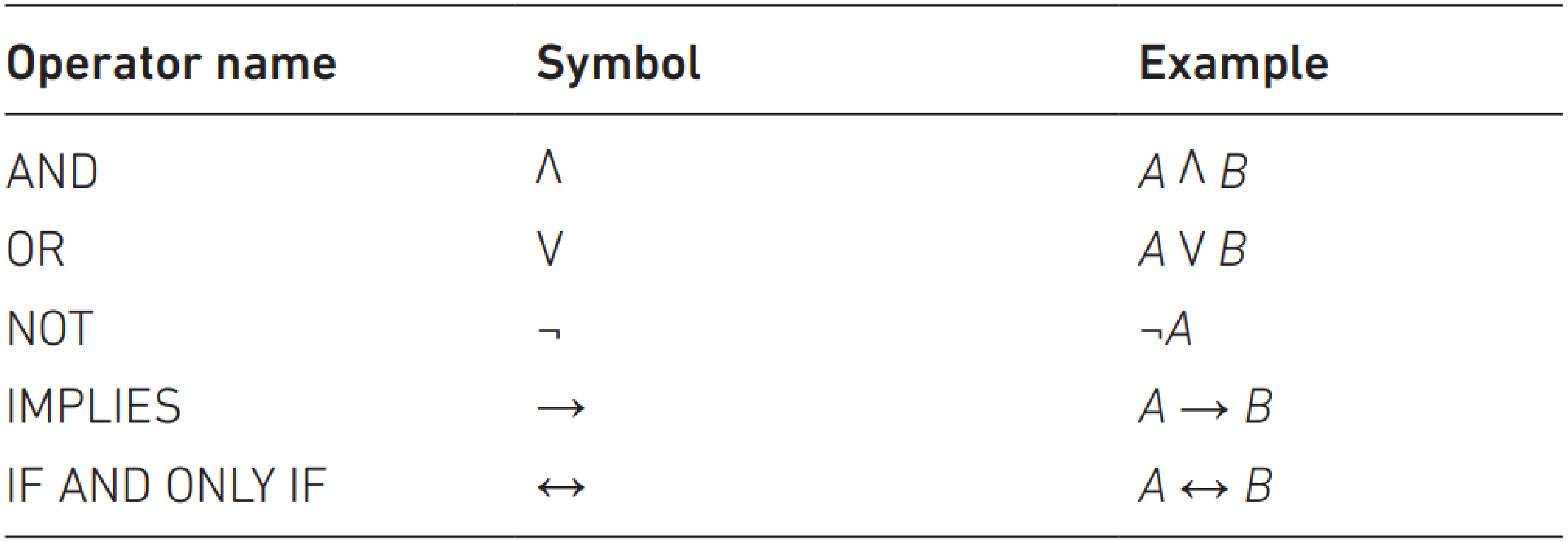
- **IF AND ONLY IF:** the technical name for this operator is biconditional.
- This behaves very similarly to implication, but a biconditional means that the second proposition is influenced solely by the first.
- If the first is true, the second is true.
- If the first is false, the second is false. No exceptions.

- **Scenario:** Light Switch
- **Statement:**
- **The light is on if and only if the switch is up.**
- **Explanation:**
- This means two things:
  - **If the switch is up, the light is on.**
  - **If the light is on, the switch must be up.**
- Both statements must be true together or false together.

Symbolic logic

- Logic requires precision and an absence of ambiguity

Logical operators and their symbols



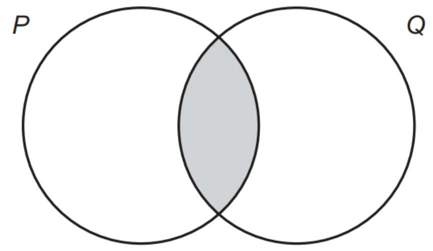
Operator name	Symbol	Example
AND	$\wedge$	$A \wedge B$
OR	$\vee$	$A \vee B$
NOT	$\neg$	$\neg A$
IMPLIES	$\rightarrow$	$A \rightarrow B$
IF AND ONLY IF	$\leftrightarrow$	$A \leftrightarrow B$

- Symbolic logic also gives each logical operator formal rules, which promote precision and eliminate ambiguity.
- The truth table states whether each combination is logically valid or not.
- Let’s start by looking at the truth table for conjunctions.

Truth table of conjunction (logical AND)

P	*Q*	*P* AND *Q*
True	True	True
True	False	False
False	True	False
False	False	False

Venn diagram for the AND operator



P	*Q*	*P* AND *Q*
0	0	0
0	1	0
1	0	0
1	1	1

Truth table of disjunction (logical OR)

P	*Q*	*P* OR *Q*
True	True	True
True	False	True
False	True	True
False	False	False

Figure 2.2 Venn diagram for the OR operator

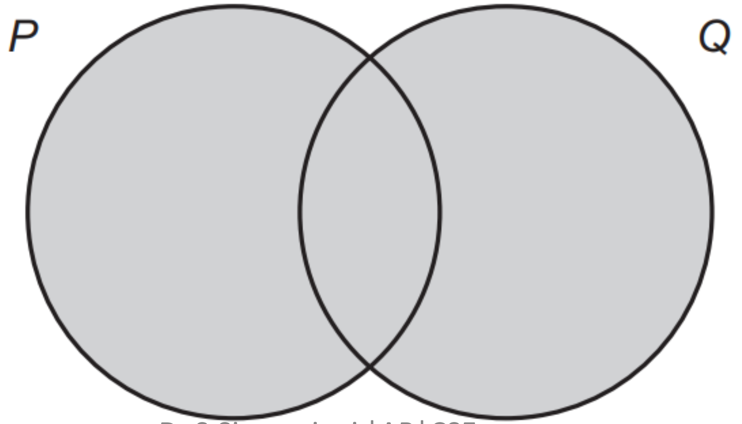


Table 2.5 Truth table of negation (logical NOT)

P	NOT *P*
True	False
False	True

Figure 2.3 Venn diagram for the NOT operator

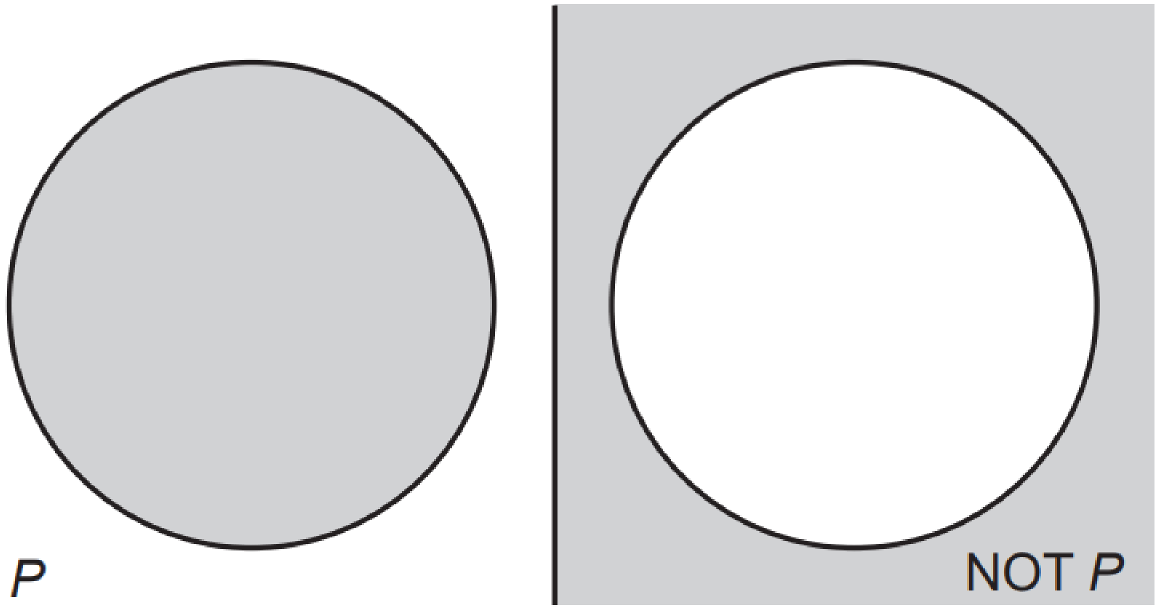
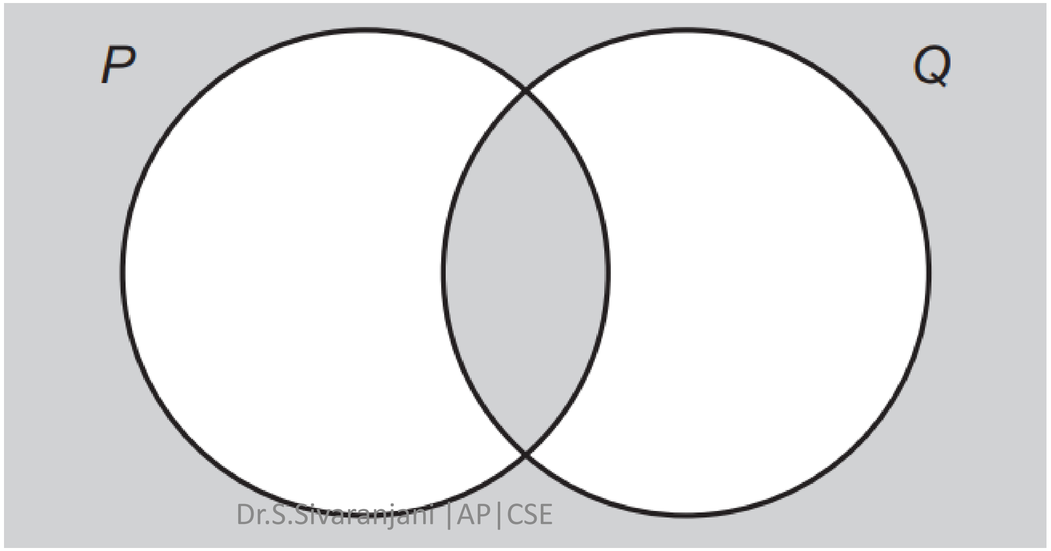


Table 2.7 Truth table of biconditional (logical IF AND ONLY IF)

P	*Q*	*Q* IF AND ONLY IF *P*
True	True	True
True	False	False
False	True	False
False	False	True

Figure 2.5 Venn diagram for the IF AND ONLY IF operator



ALGORITHMIC THINKING

- Logic and algorithms are not the same.
- Rather, algorithms build on logic because, as part of their work, they make logical decisions.

- Logic gives you a set of rules that allow you to reason about some aspect of the world.
- Logic can deal with things that are dynamic and continually changing

- **Algorithm:** a sequence of clearly defined steps that describe a process to follow a finite set of unambiguous instructions with clear start and end points

Defining algorithms

- **Collection of individual steps**
- **Definiteness** [meaning that every step must be precisely defined. Each step in an algorithm can have one and only one meaning, otherwise it is ambiguous]
- **Sequential:** The steps that make up the process must be carried out in the order specified.

- State: the current configuration of all information kept track of by a program at any one instant in time

- A variable in an algorithm is more like a scratchpad that’s used as a placeholder for important information the computer should keep note of.
- It can also have its values updated throughout an algorithm’s execution (this is called assignment).

Controlling algorithm execution

- Two ways of controlling the execution of an algorithm

1. Iteration
2. Selection

Example algorithm

- **Pseudocode:** which is an informal means of writing an algorithm.
- It’s aimed at a human audience rather than a computer, but it nevertheless closely follows the conventions of programming languages.
- This means you can easily translate your ideas into real program code by first writing them as pseudocode.

Pseudocode

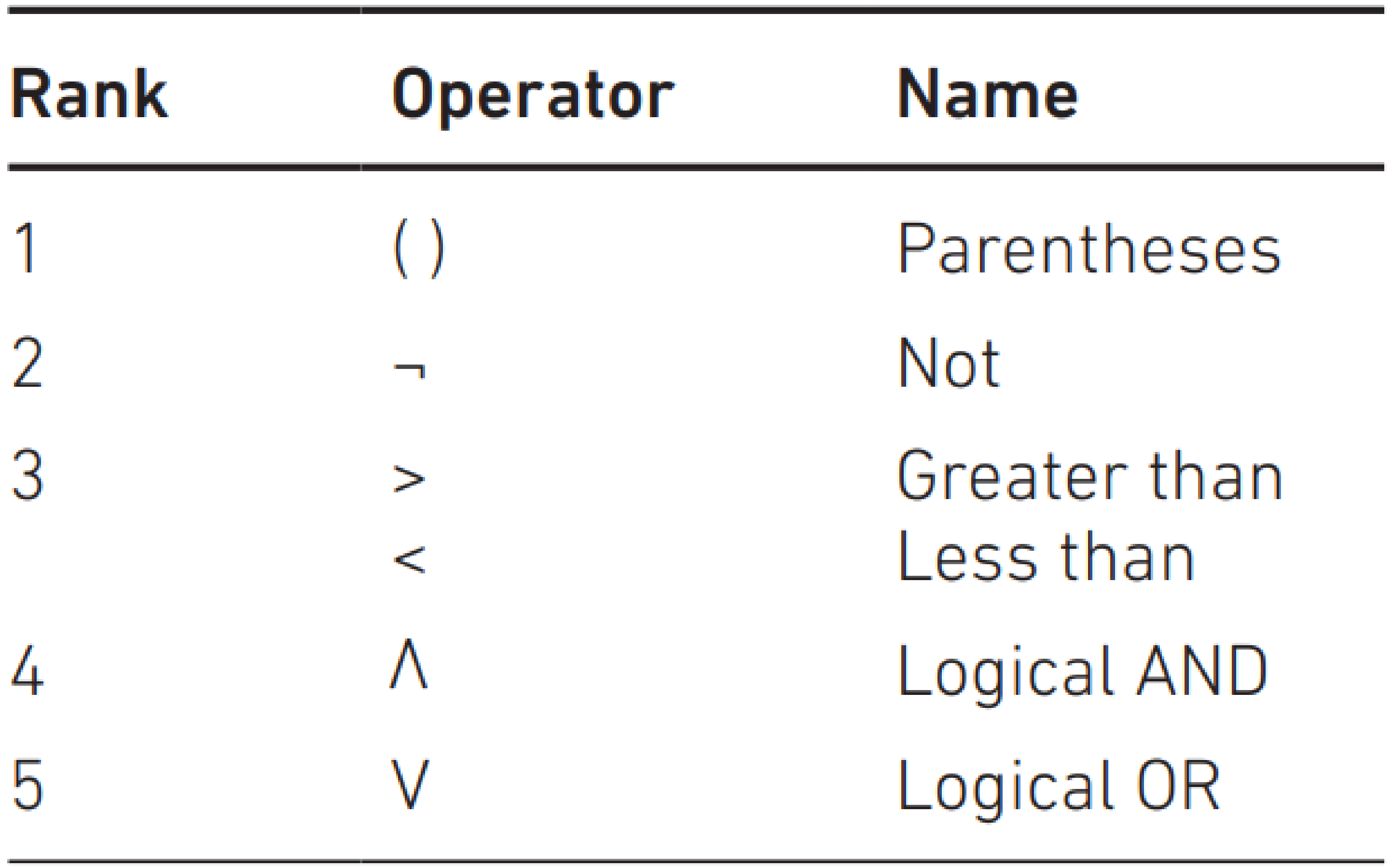
```
 1. game is started
 2. begin loop:
    3. prompt player to choose a square
    4. if chosen square is not occupied, then put player's symbol in
       that square
    5. check board to see if a row has been achieved
    6. if a row has been achieved, then game is won
    7. if a row has not been achieved and no squares are available,
       then game is drawn
    8. switch to other player
 9. exit loop if game is won or game is drawn
10. display message 'Game over'
```

- When this algorithm is executed, the computer goes through each line one at a time. (This is an example of **sequence**.)
- Line 1 initialises a variable, `game`, with a particular value, `started` (**state**).
- Line 2 sets up the starting point of a loop (**iteration**). All lines that are 'inside' this loop are indented to make it clearer what's inside and outside.
- Line 3 asks for input from the player. The choice is recorded.
- Line 4 makes a **selection** based on the player's choice. (Incidentally, this is a rather user-unfriendly algorithm. If a player accidentally chooses an occupied square, they get no opportunity to fix their mistake!)
- Lines 6 and 7 both make selections, altering the value of the `game` variable under certain conditions.

- Line 9 is the end of the loop. A choice is made whether or not to go through the loop again depending on the associated condition **at this point**. There are two possibilities:

  1. The `game` variable hasn't been altered yet, meaning it still has the value `started`. In this case, execution moves back to line 3.
  2. At some point, the game was won or drawn. In either case, the loop ends and execution continues onto line 10, whereupon the game finishes with a traditional 'Game over' message.

Order of precedence for selected logical operators



Rank	Operator	Name
1	( )	Parentheses
2	¬	Not
3	>	Greater than
	<	Less than
4	∧	Logical AND
5	∨	Logical OR

- When evaluating something like

$$9-3\times4$$

- First the multiplication 9–(3×4)
- then the subtraction (9–(3×4))
- the multiplication must happen first and so the correct answer is **-3.**

Activity

- Split as 10 per team
- For the given real time scenario , create logic algorithm , flowchart/ visualization diagram, pseudocode.
- Present it after completion