DATA ENGINEERING
WEEK 1: DAY 2

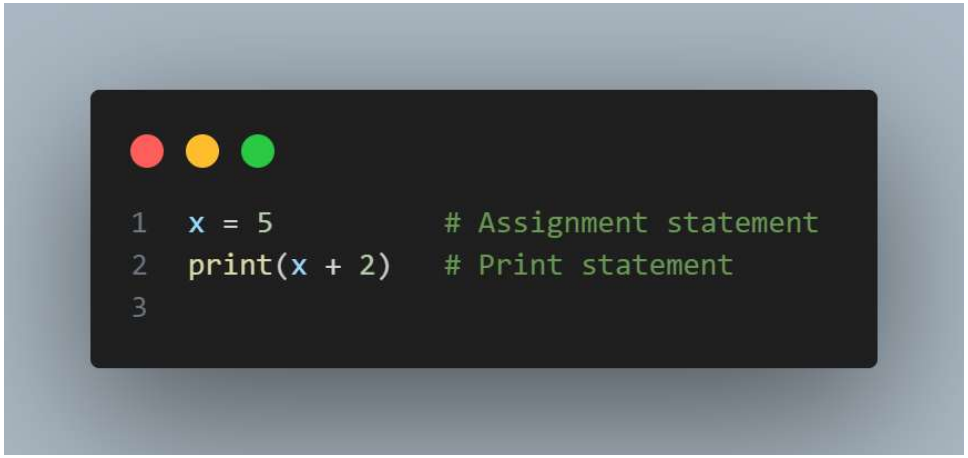
TECHRISE 2.0

Statements, Pseudocode, Algorithm, Flowchart, and Expressions

Planning and Understanding Before Coding

🎯 Learning Objectives

- **By the end of this lesson, mentees should be able to:**
  - Understand what a statement and expression are in Python.
  - Define and differentiate an algorithm, pseudocode, and flowchart.
  - Write simple pseudocode
  - Represent logic visually using basic flowcharts.



A **statement** is a complete instruction in Python that the interpreter can execute.

What is a Statement?

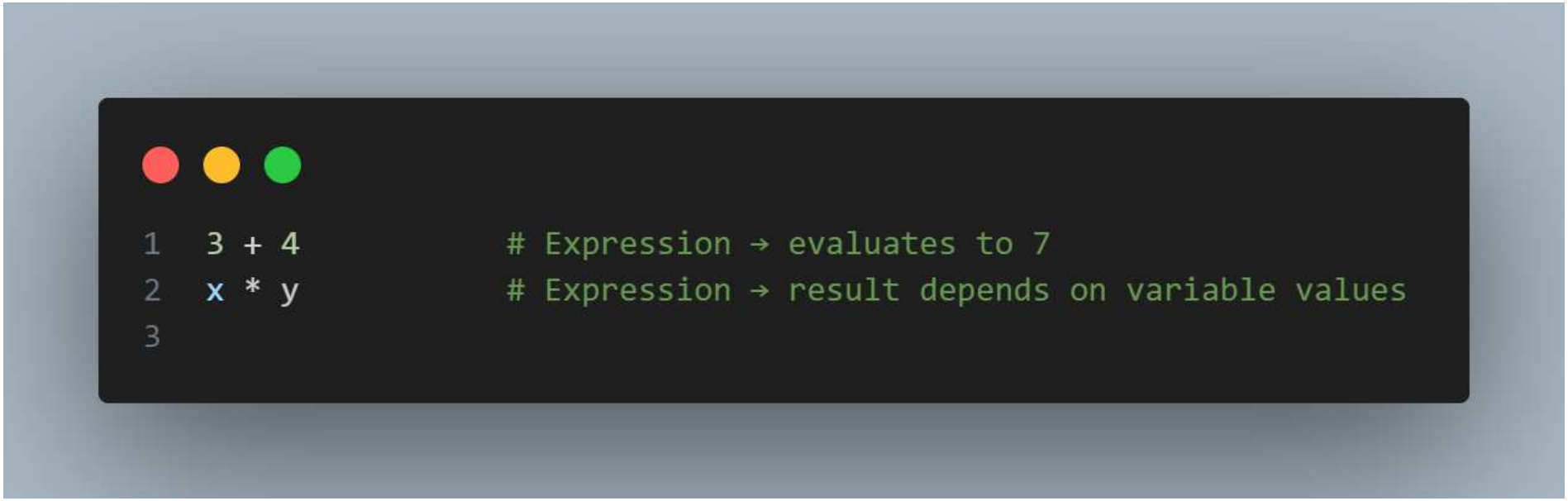
Examples

> A statement performs an action.

#	Statement	Action Performed
1	`name = "Alice"`	Assigns a value to a variable
2	`print("Hello World")`	Displays text output
3	`x = 3 + 5`	Evaluates expression and stores result
4	`if age >= 18:`	Starts a conditional decision
5	`for i in range(5):`	Initiates a loop

What is an Expression?

An **expression** is a combination of values, variables, and operators that produces a result.



Examples

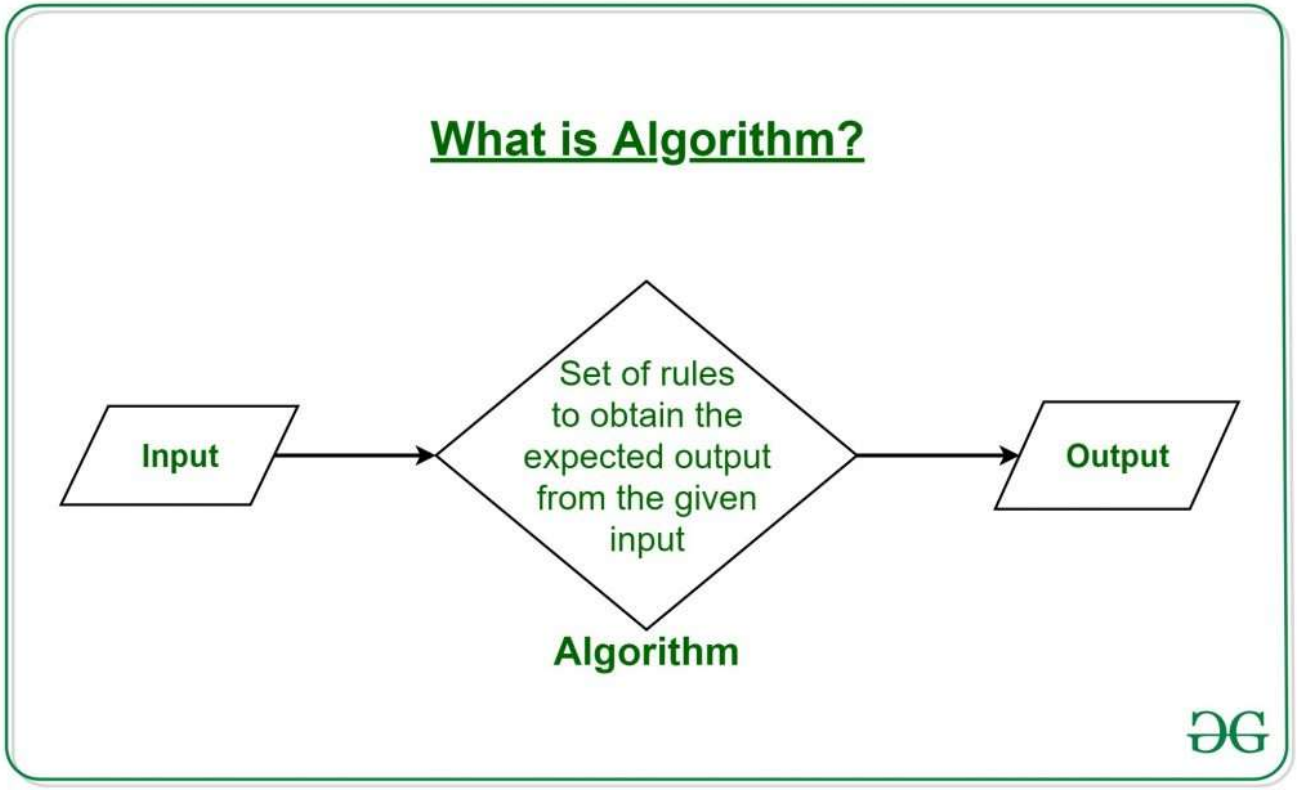
> An expression evaluates to a value.

#	Expression	Description
1	`3 + 4`	Adds two numbers → `7`
2	`x * y`	Multiplies two variables
3	`"Hi" + " there"`	Concatenates strings → `"Hi there"`
4	`a > b`	Compares two values → `True` / `False`
5	`not is_ready`	Logical NOT operation → reverses Boolean

Difference Between Expression and Statement

- ✅ **Note:** Every expression can be a part of a statement, but not all statements are expressions.

Aspect	Expression	Statement
Produces value	✅ Yes	🚫 Not always
Can be assigned	✅ Yes (e.g., `x = 3 + 4`)	🚫 No
Executes	As part of a statement	As a full instruction
Example	`3 + 5`	`x = 3 + 5`



What is an Algorithm?

A **step-by-step** procedure or set of instructions to solve a problem.

5 Examples of Algorithm

- A **step-by-step** procedure or set of instructions to solve a problem.

> Step-by-step solution to a problem (written in plain English).

#	Problem	Algorithm (Steps)
1	Add 2 numbers	1. Start 2. Input A, B 3. Add A+B 4. Display result 5. End

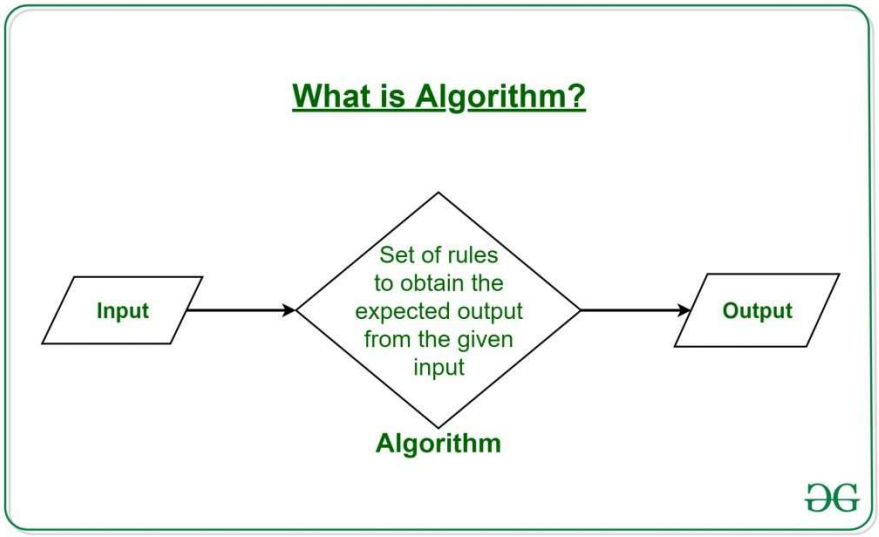
Examples

2	Find if a number is even	1. Start 2. Input number 3. If divisible by 2 → even 4. Else → odd 5. End
3	Get the square of a number	1. Start 2. Input number 3. Multiply number by itself 4. Display result 5. End

Examples

4	Calculate the area of a rectangle	1. Start 2. Input length, breadth 3. Area = length × breadth 4. Print area 5. End
5	Check eligibility to vote	1. Start 2. Input age 3. If age ≥ 18 → Eligible 4. Else → Not eligible 5. End

Algorithms typically follow a logical structure:



- **Input:** The algorithm receives input data.
- **Processing:** The algorithm performs a series of operations on the input data.
- **Output:** The algorithm produces the desired output.

What is the Need for Algorithms?

Algorithms are essential for solving complex computational problems efficiently and effectively. They provide a systematic approach to:

- **Solving problems:** Algorithms break down problems into smaller, manageable steps.
- **Optimizing solutions:** Algorithms find the best or near-optimal solutions to problems.
- **Automating tasks:** Algorithms can automate repetitive or complex tasks, saving time and effort.

What is Pseudocode?

- A Pseudocode is defined as a step-by-step description of an algorithm. Pseudocode does not use any programming language in its representation instead it uses the simple English language text as it is intended for human understanding rather than machine reading.
- Pseudocode is the intermediate state between an idea and its implementation(code) in a high-level language.

```
START
INPUT num1, num2
IF num1 > num2 THEN
   PRINT num1
ELSE
   PRINT num2
END IF
END
```

5 Examples of Pseudocode

> Code-like instructions (not actual code).

#	Task	Pseudocode

1	Add two numbers	`INPUT A, B`
		`SET Sum = A + B`
		`PRINT Sum`

2	Check even or odd	`INPUT num`
		`IF num MOD 2 = 0 THEN`
		`PRINT "Even"`
		`ELSE PRINT "Odd"`

3	Find square of number	`INPUT num`
		`SET square = num * num`
		`PRINT square`

4	Calculate rectangle area	`INPUT length, width`
		`SET area = length * width`
		`PRINT area`

5	Voting eligibility	`INPUT age`
		`IF age >= 18 THEN`
		`PRINT "Eligible"`
		`ELSE PRINT "Not eligible"`

What is the need for Pseudocode

- Pseudocode is an important part of designing an algorithm, it helps the programmer in planning the solution to the problem as well as the reader in understanding the approach to the problem. Pseudocode is an intermediate state between algorithm and program that plays supports the transition of the algorithm into the program.

How to write Pseudocode?

- Before writing the pseudocode of any algorithm the following points must be kept in mind.
- Organize the sequence of tasks and write the pseudocode accordingly.
- At first, establishes the main goal or the aim.
- Use standard programming structures such as **if-else**, **for**, **while**, and **cases** the way we use them in programming.
- Use appropriate naming conventions.
- Reserved commands or keywords must be represented in **capital letters**.
- Check whether all the sections of a pseudo code are complete, finite, and clear to understand and comprehend. Also, explain everything that is going to happen in the actual code.
- Don't write the pseudocode in a programming language. It is necessary that the pseudocode is simple and easy to understand even for a layman or client, minimizing the use of technical terms.

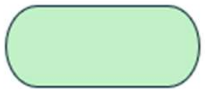
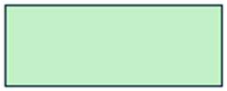
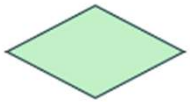
Algorithm vs Pseudocode

Feature	Algorithm	Pseudocode
Definition	Step-by-step procedure to solve a problem	Human-readable format to represent an algorithm
Form	Plain English or structured natural language	Structured, code-like but not actual code
Syntax rules	No strict rules	Follows programming logic but not strict syntax
Ease of use	Easy to write and understand	Easier to translate to code
Focus	Logic and steps of solving a problem	Code structure and logic
Conversion to code	Requires interpretation	Easy to convert into real programming code
Example	"Input two numbers and add them"	`INPUT A, B; SET C = A + B; PRINT C`

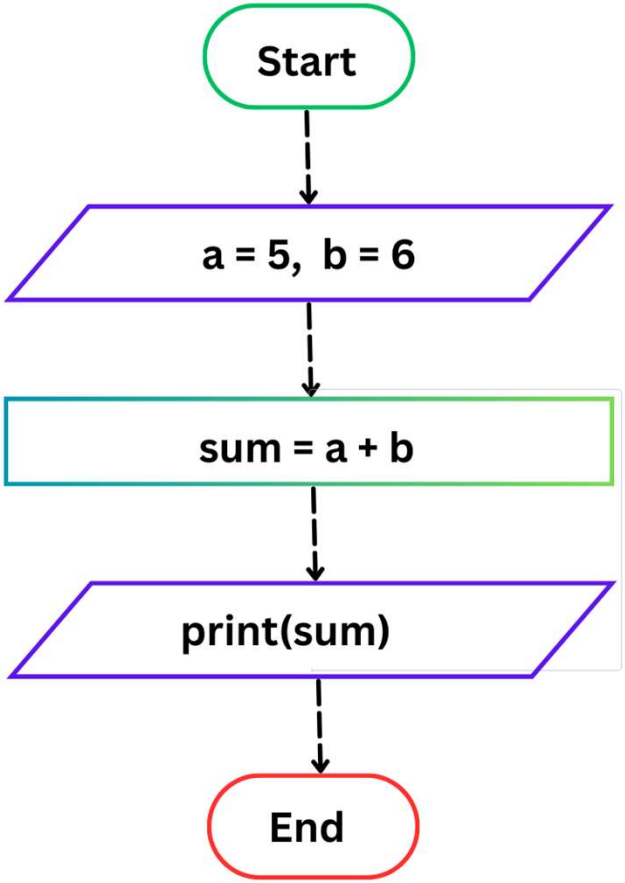
What is a Flowchart?

A **visual diagram** that represents an algorithm using symbols.

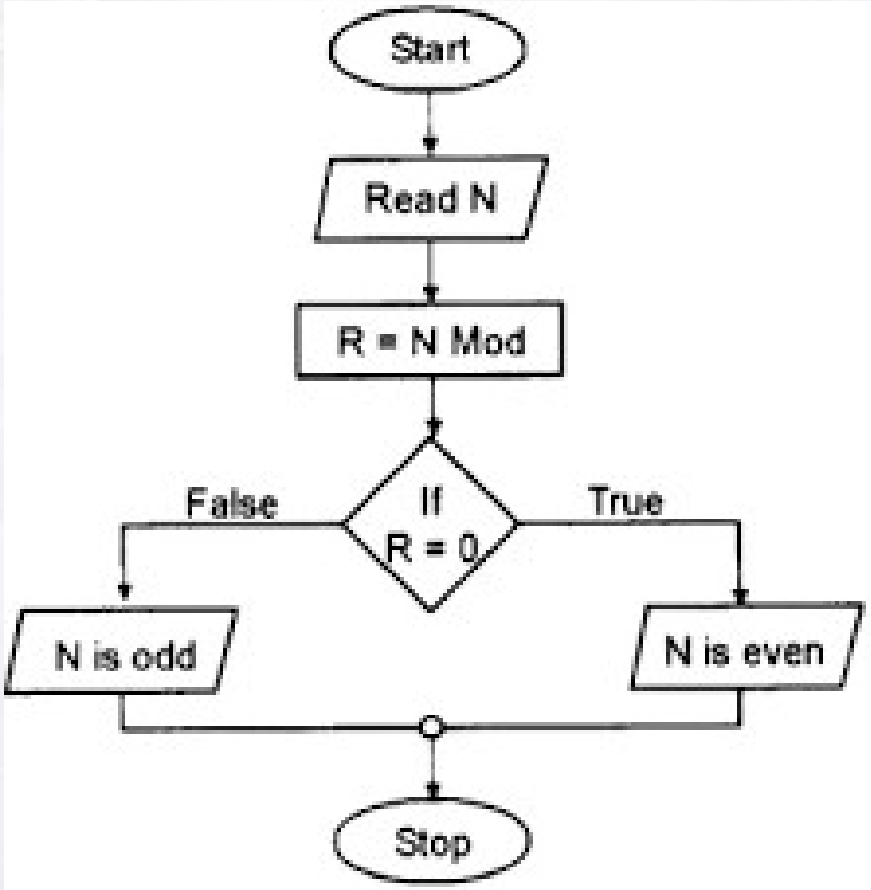
Common symbols:

Symbol	Name	Function
	Oval	Represents the start or end of a process
	Rectangle	Denotes a process or operation step
	Arrow	Indicates the flow between steps
	Diamond	Signifies a point requiring a yes/no
	Parallelogram	Used for input or output operations

Simple Flowchart

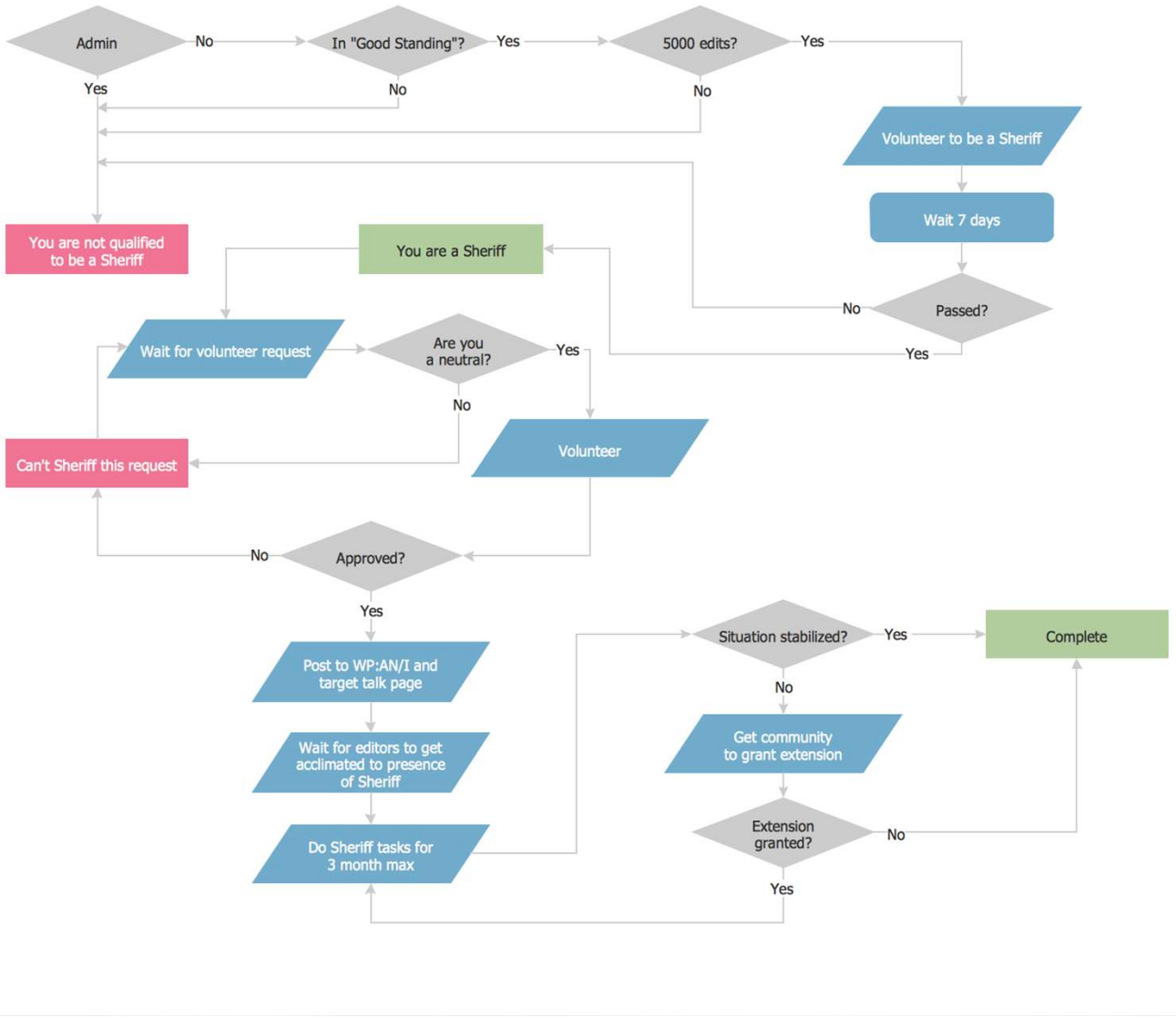


Simple Flowchart



Cross-functional Flowchart

Process Flowchart



Flowchart vs Pseudocode

Feature	Flowchart	Pseudocode
Definition	Diagrammatic/visual representation of logic	Textual representation of logic
Format	Uses symbols (ovals, diamonds, rectangles)	Uses structured English/code-like statements
Clarity	Clear for visual learners	Clear for those used to text/code
Symbolic use	Relies on standardized flowchart symbols	Uses logical keywords (`IF`, `PRINT`, etc.)
Purpose	Visualize process flow	Prepare code structure before actual coding
Conversion to code	Requires translation and interpretation	Easy to convert to actual code
Best used for	Explaining logic to non-programmers	Bridging the gap between logic and programming
Tool required	Drawing tools (e.g., diagrams.net, Lucidchart)	Can be done with pen and paper or text editor

Assignment