Orientation Activity Guide: Experiential Algorithmic Thinking

Target Audience: Incoming CS/IT Students (No programming experience required)

Objective: To introduce the concepts of algorithms, precision, edge cases, coordinates, and classification through interactive, non-coding physical challenges.

Duration: 60–90 minutes

Introduction for the Facilitator

Students often mistake "programming" for "typing code in Python or Java." This orientation session is designed to break that misconception immediately. By the end of these three tasks, students will realize that computer science is not about memorizing syntax—it is a way of thinking, problem-solving, and communicating with absolute precision.

Phase 1: The Human Robot (The Concept of Algorithms)

Time: 20 Minutes

Core Concepts: Step-by-step execution, precision, implicit vs. explicit instruction, edge cases.

The Setup

Address the room with authority but high energy:

> *"Imagine I am a robot. I know absolutely nothing. I have zero intuition, zero common sense, and zero life experience. However, I am incredibly obedient. I will follow your instructions exactly as you write them down. No more, no less."*

The Assignment

Instruct students to work in pairs or small groups (3-4 students) for 5 minutes. They must choose **one** of the following tasks and write down a step-by-step algorithm for it:

1. **How to make a cup of tea.**
2. **How to brush your teeth.**
3. **How to tie your shoelaces.**

Rule: Every single step must be completely explicit. Make absolutely no assumptions.

Example given to students:

- ❌ **Bad (Implicit):** Make tea.
- ✅ **Good (Explicit):**
    1. Take a vessel.
    2. Pour 200 ml of water into the vessel.
    3. Place the vessel on the stove.
    4. Turn the stove ON.

The Next-Day (or Post-Task) Reflection

Once they finish writing, the facilitator acts out some of the instructions literally to show how easily "code" breaks when it lacks precision.

Ask the room the following questions to drive the point home:

- *"What happens if the robot doesn't know what a 'cup' is?"*
- *"What happens if you forgot to tell me to turn the stove ON before waiting for the water to boil?"*
- *"What if you said 'add sugar' but didn't specify how much? Do I dump the whole bag in?"*

The "Ah-Ha!" Takeaway

Before writing a single line of code, students have just discovered:

- **Algorithms:** A finite sequence of rigorous instructions.
- **Precision:** Computers only do what you tell them to do, not what you *intended* for them to do.
- **Edge Cases & Ambiguity:** The hidden assumptions in human language that cause software bugs.

Phase 2: Blindfold Navigation (Logic & Control Flow)

Time: 25 Minutes

Core Concepts: Sequential logic, loops/conditions (control flow), coordinates, robotics thinking.

The Setup

Clear a safe path in the classroom. Ask for one brave volunteer to be "The Explorer" and another to be "The Navigator."

- **The Explorer** is blindfolded (or must close their eyes tightly).
- **The Navigator** must guide the Explorer from the classroom door to a specific desk or chair on the other side of the room.

The Rules

1. Only verbal instructions are allowed.
2. No pointing, physical guiding, or gestures.
3. No demonstrations beforehand.

The Execution

The Navigator will quickly realize that vague commands fail.

- ❌ **Failed Command:** *"Just walk forward and turn when you get to the desk."* (The Explorer doesn't know where the desk is or when to stop).
- ✅ **Successful Command:** *"Take 10 normal steps forward. Pause. Turn 90 degrees to your right. Take 4 steps forward."*

The "Ah-Ha!" Takeaway

By trying to move a human through physical space with pure text, they have just reverse-engineered:

- **Sequential Logic:** The order of execution matters. A turn before a step is a completely different destination.
- **Control Flow:** Learning how to give conditions (e.g., *"Take steps forward until your shin gently touches a solid object, then stop"*).

- **Coordinate Systems:** Realizing that computers and robots navigate using grid systems, angles, and discrete steps.

Phase 3: The Alien and the Cat (Foundations of Machine Learning)

Time: 20 Minutes

Core Concepts: Feature extraction, classification, training data, false positives, false negatives.

The Setup

Present this final scenario to the class:

> *"An alien has landed on Earth. They have never seen Earth's wildlife. Your job is to teach this alien how to identify a cat, and only a cat. You cannot say 'because it looks like a cat.' You must define its features."*

The Challenge

Ask the students to list the defining features of a cat. They will likely suggest:

1. Has ears.
2. Has a tail.
3. Has whiskers.
4. Walks on four legs.
5. Has fur.

The Facilitator's Counter-Attack (The Stress Test)

Now, act as the data environment and test their "model" by presenting edge cases:

- *"I found a creature with ears, a tail, whiskers, four legs, and fur. It weighs 400 pounds and has orange and black stripes. Is this a cat?"* (Introducing **Tigers / Scale / Outliers**)
- *"I found a toy stuffed cat. It fits all your rules perfectly. Is this a cat?"* (Introducing **Context / False Positives**)
- *"I found a cat that lost its tail in an accident and has a shaved patch from a vet visit. It doesn't fit your rules. Is it still a cat?"* (Introducing **Noise / Missing Data / False Negatives**)

The "Ah-Ha!" Takeaway

Without touching a single neural network or mathematical formula, you have just introduced them to:

- **Classification & Features:** Grouping data based on specific, measurable attributes.
- **Training Data vs. Real-World Data:** How a model built on perfect assumptions breaks when it meets the messy, real world.
- **False Positives & Negatives:** Learning that AI is rarely 100% certain, but rather a game of probability and refining parameters.

The Closing Address (The "Mind-Blow" Moment)

Gather everyone's attention to wrap up the orientation.

> *"Look at what we did today.*

Today, you didn't install software. You didn't write a single line of Python, C++, or JavaScript.

But make no mistake: you did computer science. You designed an algorithm, you mapped out navigation logic, and you built a classification model for artificial intelligence.

From this day forward, when you look at code, remember: the code is just the translation