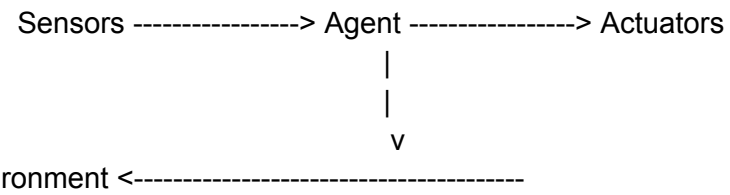
AI Agent Layman's Definition:

An **agent** is anything (like a robot, software, or machine) that can **see or sense its surroundings** using **sensors** (like eyes or cameras), and then **do something** in that surrounding using **actuators** (like hands, wheels, or motors).



==========

What does it mean by '<u>Environment</u>' in AI Agent:

The environment is simply the **surroundings** or **situation** the AI is placed in. It's where the AI gets information from and tries to do something useful.

Agent: Robot vacuum cleaner

Environment:

- The house: walls, furniture, dirt patches

Sensors: Bump sensors, dirt sensors

Actuators: Wheels to move, brush to clean

Interaction: It detects a dirty floor and moves to clean it

==========

<u>Percept:</u> The Contents the Agents Sensor's are Perceiving

<u>Percept Sequence:</u> The Complete history of things the agent has ever perceived

==========

<u>Concept of Rationality</u>:

A rational agent acts to **maximize expected performance** based on its **percepts and knowledge**.

Key Elements:

1. **Performance Measure**
   Defines what counts as success for the agent.
2. **Percept Sequence**
   The complete history of what the agent has perceived so far.
3. **Knowledge**
   What the agent knows about the environment, including built-in and learned knowledge.
4. **Available Actions**
   The agent chooses from the actions it can actually perform.
5. **Autonomy**
   A rational agent improves by learning from experience rather than relying only on built-in rules.

Clarifications:

- Rational ≠ Perfect
  It means doing the best with what's known, not guaranteeing success.

- Rational ≠ Omniscient
  The agent doesn't know the future, it just makes the best possible decision given the current information.

==========

Types of Agents

Simple Reflex Agent:

A **Simple Reflex Agent** is the most basic kind of agent.
It **looks at the current situation** (percept) and **acts immediately** based on a set of condition–action rules.

{IF condition THEN action}

- **No memory** – it can't remember what it did before.
- **Fails in partially observable environments** – can't work well if it can't see the whole situation.
- **No learning** – it cannot improve over time.

A Simple Reflex Agent chooses actions **only based on the current input**, using **fixed rules**.
It is fast and simple, but not smart or flexible.

Model Based Agent:

A **Model-Based Agent** is smarter than a simple reflex agent.
It **keeps track of the world**, not just the current percept.

How it works:

1. **Perceives** the current state from sensors
2. **Updates internal model** (a memory of what the world is like)
3. **Uses rules** ("if [condition], then [action]") just like a simple agent
4. **Chooses action** based on:
    - Current percept
    - Internal state (past info)

Limitations:

- Still rule-based
- Still doesn't plan ahead or optimize goals
- No learning (unless combined with learning components)

Goal-Based Agent:

A Goal-Based Agent decides what to do by considering its **goals** — specific outcomes it wants to achieve.
It does **not just react**, it **thinks ahead** and chooses actions that help reach the goal.

{IF action leads to goal THEN do action}

Has a model of the world – can predict the effects of actions.

Can handle new situations – it reasons to find a path to the goal.

Still rule-based, but more flexible – actions depend on goals, not just current conditions.

A Goal-Based Agent chooses actions by comparing possible future outcomes to its goals.
It is more intelligent than reflex agents but still does not measure how *good* different goals are.

Utility-Based Agent:

A Utility-Based Agent goes one step further than goal-based agents.
It tries to **maximize happiness or satisfaction**, not just reach any goal.

{IF action leads to highest utility THEN do action}

Uses a **utility function** – a way to measure how good or bad an outcome is.

Can compare multiple goals – and choose the one that gives the **best overall result**.

Handles trade-offs – useful in complex environments with conflicting goals.

A Utility-Based Agent chooses actions that maximize expected utility.
It is the most flexible and intelligent kind of agent among these, but also the most complex to design.

==========

SImple Flow chart for preparing the Agents

Step 1: Prepare Model

|

└── [Create any ML/CNN Model, in this case:]------> full_model.keras (CNN for fundus classification)

✅ Why: We need perception first; it feeds the agent

Step 2: Create Sensor Module (sensor.py)

|

└── Loads model, preprocesses image, returns (class, confidence)

✅ Why: Abstracts perception logic, reusable across agents

Step 3: Define Simple Decision Logic (decision.py)

|

└── Uses rules like: if confidence > threshold → "Refer"

✅ Why: Decouples decision-making from perception

Step 4: Implement Agent Loop (agent_loop.py)

|

└── Controls: → perceive → decide → act (log/store action)

✅ Why: Core logic of the agent; lets you run simulations

Step 5: Add Simulated Environment (environment.py)

|

└── Returns reward or penalty based on action vs. true label

✅ Why: Enables evaluation of agent behavior, not just accuracy

Step 6: Run Agent Demo Notebook (agent_demo.ipynb)

|

└── Iterate over test images, log predictions/actions/rewards

✅ Why: Manual testing & debugging of loop

Step 7: Evaluate & Improve

|

├── Add confidence calibration (utility agent)

├── Add internal memory/state (model-based)

└── Add learning (Q-learning, RL)

✅ Why: Evolve agent to be more intelligent

Visual Representation of Agent Execution Flow