**Basics:**

**State:** It’s like having the entire context of the application. More of like data that’s being passed, which can be updated while passing through the nodes.

**Nodes:**

* These are the individual specific tasks performers at specific point in the workflow. It updates the state while it executes.
* These can be condition specific.
* These are nothing but a simple python functions that take the state, perform some operations and then updated the state. So, It can be determined that nodes take state as an input and return state as the output.
* While building nodes, emphasis should be on providing doc strings to each and every node. Because eventually, we will be building AI agents. For these to have context of what each node is doing, Doc Strings are necessary.

**Graph:** An Entire hub of nodes which are connected with the help of edges (connection b/w one node and another) which has a start and end point where workflows starts and ends while executing the nodes.

**Edge:** Edges, we can consider these as connectors of two or multiple node. These are useful while adding nodes to the graph.

**Conditional Graph:** It’s a type of graph based on certain conditions that lets the router know which edge it should select to execute the node based on certain condition.

**Syntax of conditional graph:**

{

Source(router mostly),

Path (node function that decides which is the edge that is to be selected based on conditions),

{ Edge : Node, Edge: Node}

}

**Looping Graph**:

* It’s a type of graph where looping is involved. Within the graph, a loop decides where to follow a specific edge or not until a condition is met.
* If condition is met, the graph will follow another edge which is not associated with the loop.

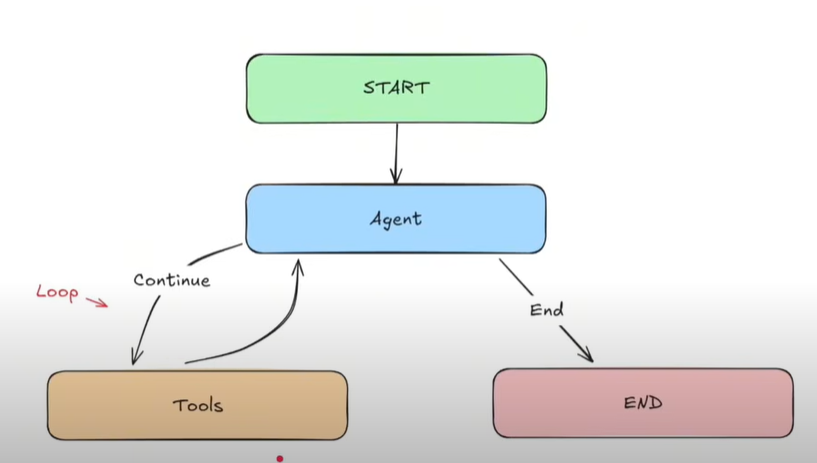
**Human and AI Messages:**

* Human Messages are the messages that we input to the LLMS for it to generate the response.
* AI Messages are the messages generated by the LLMS
* To use these we need to import the Following Classes

from langchain\_core.messages import HumanMessage, AIMessage

**ReAct Agent: Reasoning and Acting Agent:**

* This is a type of agent that will choose the tools necessary for the steps and knows when to stop the execution aswell.
* Image representation of ReAct Agent Graph



**Few Important Notes:**

from langchain\_core.tools import tool

from langgraph.graph.message import add\_messages

from langchain\_core.messages import SystemMessage, ToolMessage, BaseMessage

**Let me Explain about these above imports**

* tool is a decorator which can be used with function to bind the function with the tool
* **add messages** in langraph function which keeps **adding messages** or query to the existing state of the agent without us explicitly appending messages.
* **System Message** is a type of message that gives instructions to the LLM being used.
* **Tool Messaage** is a type of message which is passed back to the LLM after the tool is called.
* **Base Message** is a foundational class for all of the message types in langgraph**.(i.e., Any class that is inherited from Base Message is considered within state being used)**
* **Post**

**Important:**

* Usually we take the state[key] and then change or update and finally return the state.
* There’s also another way to update the state. Simply, return the key that’s available in the state that is being used along with the value that you want to replace/update the current state with.