Type Annotation :-

Normal Dictionary :-

movie = {"name": "Avengers Endgame", "year": 2099}

* efficient Data retrival
* flexible and easy to implement.
* leads to challenge in ensuring that the data is particular structure especially for large projects.
* Does not check if the data is the correct type or structure.

Typed Dictionary :-

eg

```
from typing import TypeDict
class movie(TypedDict):
    name : str
    year : int
movie = {"name": "Avengers Endgame", "year": 2099}
```

Union :-

* value can be more than one type.
* flexible and easy to code
* Type safety: helps to catch incorrect usage.

eg

```
from typing import Union
def square (x : Union[int, float]) -> float:
    return x * x
```

x = 4 ⟶ both will work
x = 1.459 ⟶

x = "string" ⇒ will throug an error.

Optional :-

```
from typing import Optional

def nice_message(name: Optional[str]) -> None:
    if name is None:
        print("Hey Random person")
    else:
        print(f"Hi there, {name}")
```

* in this above case "name" can be either string or None.
* it can not be anything else.

Any :-

Anything and everything is allowed.

```
from typing import Any

def print_value(x: Any):
    print(x)
```

Elements of langgraph:

State

* The state is a shared data structure that holds the current information or context of the entire application.
* In simple terms, it is like the application's memory. Keeping track of the variables and data that nodes can access and modify as they execute.

Analogy eg: whiteboard in a meeting room. Participants (nodes) write and read information on the whiteboard (state) to stay updated and coordinated actions.

Nodes

* Nodes are individual functions or operation that perform specific tasks within the graph.
* each node receives input (often the current state), process it, and produces an output or update state.

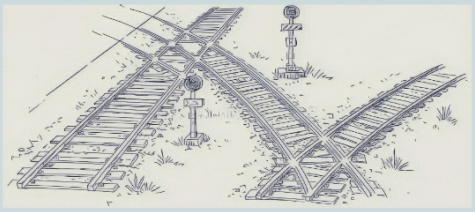
Analogy eg: in assembly line station each station does one job - attach a part, paint it, inspect quality and other process.

Edges

* Edges are the connections b/w nodes that determine the flow of execution.
* They tell us which node should be executed next after the current one complete its task.

Analogy eg: in train track (edge) connects the station (nodes) together in a specific direction.

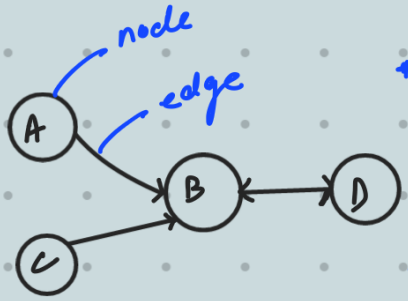
Conditional Edge



* conditional edge are specialized connections that decide the next node to execute based on specific conditions or logic applied to the current state.

Analogy eg:- Traffic lights — the condition (light color) decides the next steps.

Graph

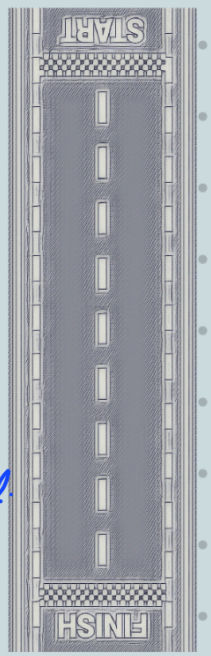


* a graph in langgraph is the overarching structure that maps out how different tasks (nodes) are connected and executed.
* it visually represents the workflow, showing the sequence and conditional paths b/w various operations.

Start

* the start node is a virtual entry point in Langgraph, making where the workflows begins
* it does not perform any operations itself but serves as the designated starting position for the graph's execution.

Analogy eg:- Race Starting Line :- The place where a race officially begins.



End

* the END node signifies the conclusion of the workflow in langgraph.
* Upon reaching this node the graph's execution stops, indicating that all intended process have been completed.

Analogy eg:- finish line in a Race :- The race is over when you cross it.

Tools

* Tools are specialized functions or utilities that nodes can utilize to perform specific task such as fetching data from an API.
* They enhance the capabilities of nodes by providing additional functionalities.
* Nodes are part of the graph structure, while tools are functionalities used within nodes.

Analogy/eg: Tool in a Toolbox:- A hammer for nails, a screwdriver for screws, each tool has a distinct purpose.