Agents and Implementation(Spade Package)

In artificial intelligence, an intelligent agent (IA) refers to a self-governing element which acts, coordinating its movement towards accomplishing objectives (i.e. it is a agent), upon a domain utilizing perception through sensors and ensuing actuators (i.e. it is intelligent) Intelligent agents may likewise learn or utilize information to accomplish their objectives. They might be basic or complex. A reflex machine, for example, an indoor regulator, is viewed for instance of a intelligent agent.
Intelligent agents in man-made brainpower are firmly identified with agents in financial aspects, and forms of the intelligent agent paradigm are considered in subjective science, morals, the logic of handy reason, just as in numerous interdisciplinary socio-psychological displaying and computer social reproductions.
Intelligent agents are additionally firmly identified with software agents (an independent PC program that completes assignments in the interest of clients). In software engineering, the term intelligent agent might be utilized to refer to a software agent that has some knowledge, in any case on the off chance that it's anything but a balanced operator by Russell and Norvig's definition. For instance, self-sufficient projects utilized for administrator help or information mining (now and again refer to as bots) are additionally called "intelligent agents".

Types of agents
Agents can be characterized by various parameters. We will currently talk about an assortment of sorts of agents that are arranged by these, and other, parameters.
The sorts of agents that we will take a look at are not totally unrelated : an interface specialist can be receptive or utility based. It can likewise be versatile or nonversatile.
The primary classes of agents are characterized as follows:
- reactive agents
- collaborative agents
- interface agents
- mobile agents
- information-gathering agents

1) reactive agent
A reactive agent (otherwise called a reflex agent) is a creation framework where contributions from nature are contrasted with standards that figure out which activities to complete. At the end of the day, reactive agents essentially respond to events in their condition as indicated by predefined standards.
Receptive agents basically recover pre-set practices like reflexes without keeping up any interior state. Then again, deliberative operators carry on progressively like they are considering, via seeking through a space of practices, keeping up inward state, and foreseeing the impacts of activities. In spite of the fact that the line among responsive and deliberative specialists can be to some degree hazy, an agent with no interior state is absolutely receptive, and one which puts together its activities with respect to the anticipated activities of different specialists is deliberative.
For instance, The robots will likely move together in a military development, for example, a precious stone, section, or wedge. They intermittently run over deterrents which avoid at least one of the robots from moving in a straight line. In the wake of passing the hindrance, all robots must modify so as to recapture their development. The agents responsively convert their tactile information (which incorporates the places of different robots) to movement vectors for keeping away from deterrents, staying away from robots, moving to an objective area, and development support. The genuine robot movement is a straightforward weighted aggregate of these vectors.
A reactive agent does not tend to perform well when its environment changes or when something happens that it has not been told about.

1.1)Goal based agents

Goal based agents are more perplexing than reactive agents. Rather than following predetermined arrangement of standards, an goal based agent acts to attempt to accomplish an objective. This is regularly done by utilizing search or planning.
It operates based on a goal in front of it and makes decisions based on how best to reach that goal. Unlike a simple reflex agent that makes decisions based solely on the current environment, a goal-based agent is capable of thinking beyond the present moment to decide the best actions to take in order to achieve its goal. In this regard, a goal-based agent operates as a search and planning function, meaning it targets the goal ahead and finds the right action in order to reach it. This helps a goal-based agent to be proactive rather than simply reactive in its decision-making.
You may take a goal-based approach to tasks at work. For example, you might set a goal for yourself to become a more efficient typist, which will help you in completing assignments more quickly. A step toward that goal, then, might be to enroll in a typing course or to devote 15 minutes a day to practice in order to increase your word count per minute. Your decisions are flexible, a hallmark of goal-based agents, but the focus is always on achieving the goal ahead.

1.2)Utility-based agent
An utility based agent is like an objective based agent, however not withstanding endeavoring to accomplish a lot of goals, the utility-based agent is additionally attempting to amplify some utility value. The utility value can be thought of as the joy of the agent, or how fruitful it is being. It might likewise consider how much work the agent needs to do to accomplish its objectives.
Consider it as: An goal based agent settles on choices dependent on accomplishing a set objective. You need to venture out from Los Angeles to San Diego, the objective based specialist will get you there. San Diego is the objective and this agent will delineate right way to get you there. In any case, in case you're heading out from Los Angeles to San Diego and experience a shut down street, the utility-based specialist will jumpstart and break down different courses to get you there, choosing the best choice for most extreme utility.

2) Interface agent
An interface agent can be thought of as an individual partner. Interface agent are normally self-sufficient specialists, equipped for learning so as to do assignments in the interest of a human client. Commonly, interface agents team up with the client, however don't have to work together with different agents; in spite of the fact that now and again, interface agents learn by seeking advice of other agents.
For instance Kasbah is a market framework where every client has a agent. The client programs the agent with a purchasing conduct profile, and the operator consults to purchase and sell things for the client. Results: Users needed increasingly "human like" arrangement from the agents, generally welcomed.
Sardine is a closeout agent that endeavors to buy an aircraft ticket for the client, in light of some predetermined inclinations. The client's agent consults with movement specialists to verify the best arrangement.

3) Mobile agents
Mobile agents are those skilled of "moving" from one spot to another. In the instance of portable robots, this truly implies moving in physical space. On account of portable programming agents, this portability for the most part refers to the Internet or other network. An agent that isn't mobile is static.
The primary advantage of mobile agent are in efficiency. An agent that needs to speak with various remote servers and request enormous amounts of data so as to settle on a choice uses a lot of transfer speed, which can be maintained a strategic distance from if the operator can physically move to the remote server and question it locally.
In the mid 1990s, General Magic made the Telescript language and condition for composing and executing versatile specialists, and depicted it with the now-prominent "cloud" illustration; as portrayed by Andy Hertzfeld:
"The excellence of Telescript," says Andy, "is that now, rather than simply having a gadget to program, we currently have the whole Cloud out there, where a solitary program can proceed to make a trip to a wide range of wellsprings of data and make kind of a virtual administration.

In this way,mobile agents can be utilized to create an appropriated processing design, where calculation happens on numerous PCs at self-assertive areas. A further preferred position of portable specialists is that they can do their tasks asynchronously: the client can set a mobile agent off on a specific assignment and would then be able to continue ahead with other work, or possibly turn the PC off. At the point when the client is prepared to get the results, the specialist can be reviewed.

4) Information agents
Information agents , otherwise called information gathering agents, are normally utilized on the Internet as are additionally here and there called Internet agents. An information agent is utilized to enable a client to discover, channel, and arrange data from the immense range of sources accessible on the Internet.
Information agents know how to search the Internet, usually using a number of search tools. In this way, they are able to cover as much content as possible and thus maximize their recall . The real challenge is usually precision. This is heavily dependent on the ability of the agent to receive input instructions from the user. Some agents learn by example: the user shows the agent examples of pages that are relevant and pages that are not relevant, and the system learns to differentiate the two groups. Other agents are directed by keywords or more sophisticated information retrieval techniques to identify relevant material for the user.

5) Multiagent frameworks
Multiagent frameworks are a typical method for misusing the potential intensity of agents by consolidating numerous operators in a single framework. Every agent in a multiagent framework has deficient data and is unequipped for taking care of the whole issue without anyone else, yet joined together, the agents structure a framework that has adequate data and capacity to take care of the issue. The framework does not have a concentrated control instrument for taking care of the issue.
An example of how many simple agents can combine together to produce complex behavior can be seen by examining the way that ant colonies function. Each ant has very little intelligence and very little ability to learn. Taken as a whole, however ,the ant colony is able to deal with complex situations and in some ways behaves as a single living entity.
Communication and collaboration are desirable properties of multiagent systems. Communication means, for example, that agents can inform each other of changes in the environment or of new discoveries they have made. Collaboration means that agents can work together to solve a common goal.

6) collaborative agents
Collaborative agent frameworks are multiagent frameworks in which the agents team up with one another to achieve objectives. This property, of coordinating to accomplish a typical goal,is known as benevolence.
Collaborative agent frameworks can exploit their parallel nature so as to take care of issues quicker than would somehow or another be conceivable. They are likewise more dependable than customary frameworks in light of the fact that extra agents can be added to give excess: on the off chance that one operator fizzles, or gives wrong data, this won't influence the general execution of the framework on the grounds that different agents will give remedial data.

ABOUT SPADE PACKAGE
- Smart Python agent Advancement Environment
- A multi-agent frameworks stage composed in Python and based on moment informing (XMPP).
- Develop operators that can chat both with other agents and humans.
- Free computer program: MIT permit

Features

- Multi-agent stage based on XMPP
- Presence notice permits the framework to know the current state of the operators in real-time
- Python >=3.6
- Asyncio-based Agent
- demonstrate based on behaviours
- Supports FIPA
- metadata utilizing XMPP Information Shapes (XEP-0004: Information Forms)
- Web-based interface
- Use any XMPP server

The Agent Model is essentially made out of an association instrument to the stage, a message dispatcher, and a lot of various practices that the dispatcher gives the messages to. Each agents needs an identifier called Jabber ID a.k.a. JID and a legitimate secret phrase to set up an association with the XMPP server.
The JID (created by a username, a @, and a server space) will be the name that recognizes an operator in the stage, for example myagent@myprovider.com.
Association with the stage
Correspondences in SPADE are taken care of inside by methods for the XMPP convention. This convention has an instrument to enroll and confirm clients against a XMPP server. After a successful register, every agent holds an open and industrious XMPP stream of correspondences with the stage. This procedure is naturally activated as a major aspect of the specialist enlistment process.

In [ ]:

```
from spade import agent

class DummyAgent(agent.Agent):
    async def setup(self):
        print("Hello World! I'm agent {}".format(str(self.jid)))

dummy = DummyAgent("Uzma@xmpp.jp", "uzma123")
dummy. start()

dummy.stop()
```

In [ ]:

```
Output

python dummyagent.py
Hello World! I'm agent Uzma@xmpp.jp
```

In [ ]:

```
import time
import asyncio
from spade.agent import Agent
from spade.behaviour import AddBehaviour
class DummyAgent(Agent):
    class MyBehav(AddBehaviour):
        async def on_start(self):
            print("Starting behaviour . . .")
            self.add = 0
self.b=2
self.c=3
        async def run(self):
            print("addition of two numbers is: {}".format(self.add))
            self.add= b+c
            await asyncio.sleep(1)
    async def setup(self):
        print("Agent starting . . .")
        d = self.MyBehav()
        self.add_behaviour(d)
    dummy.stop()
```

In [ ]:

```
Output
python dummyagent.py
Agent starting . . .
Starting behaviour . . .
Addition of two numbers is: 5
```

In [ ]:

```
a1 = Template()
a1.sender = "sender1@host"
a2 = Template()
a2.to = "recv1@host"
a2.metadata = {"performative": "query"}
m = Message()
m.sender = "sender1@host"
m.to = "recv1@host"
m.metadata = {"performative": "query"}

# And AND operator
assert (a1 & a2).match(m)

t3 = Template()
t3.sender = "not_valid_sender@host"

# A NOT complement operator
assert (~t3).match(m) t1 = Template()
```

In [ ]:

```
Output
python sender.py
Sender Agent started
InformBehav running
Message sent!
Agent finished with exit code: Job Finished!
```