JASECI

Getting Started

Coursework Series : Week 3

Coursework Outline & Outcomes

Using STD Actions

Using Abilities & Walkers in Depth

Add more features to our small Application with Jaseci

JASECI

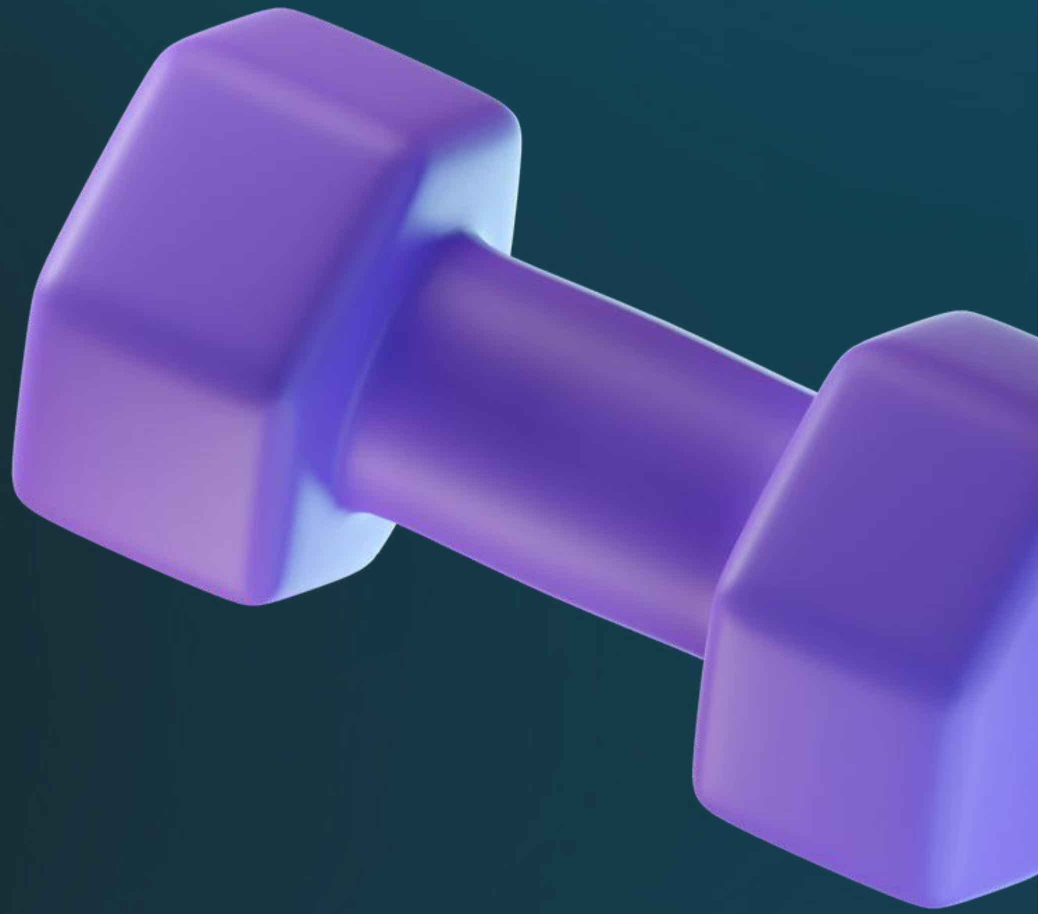
Explanation

The app you're building will let users load a family tree data file in JSON format, create a graph using Jaseci with each person as a node and relationships as edges, and visualize it in Jaseci Studio. Users can then use Jaseci's simulation capabilities to analyze the graph and explore family dynamics.

- THE APP CAN LOAD JSON FILES (ACTIONS) <--
- *We will create nodes for each person (Nodes)*
- *Connect them to each other using edges (Edges)*
- DIFFERENT FUNCTIONALITIES <--
- *Visualize the Graph in Jaseci Studio (Graph)*

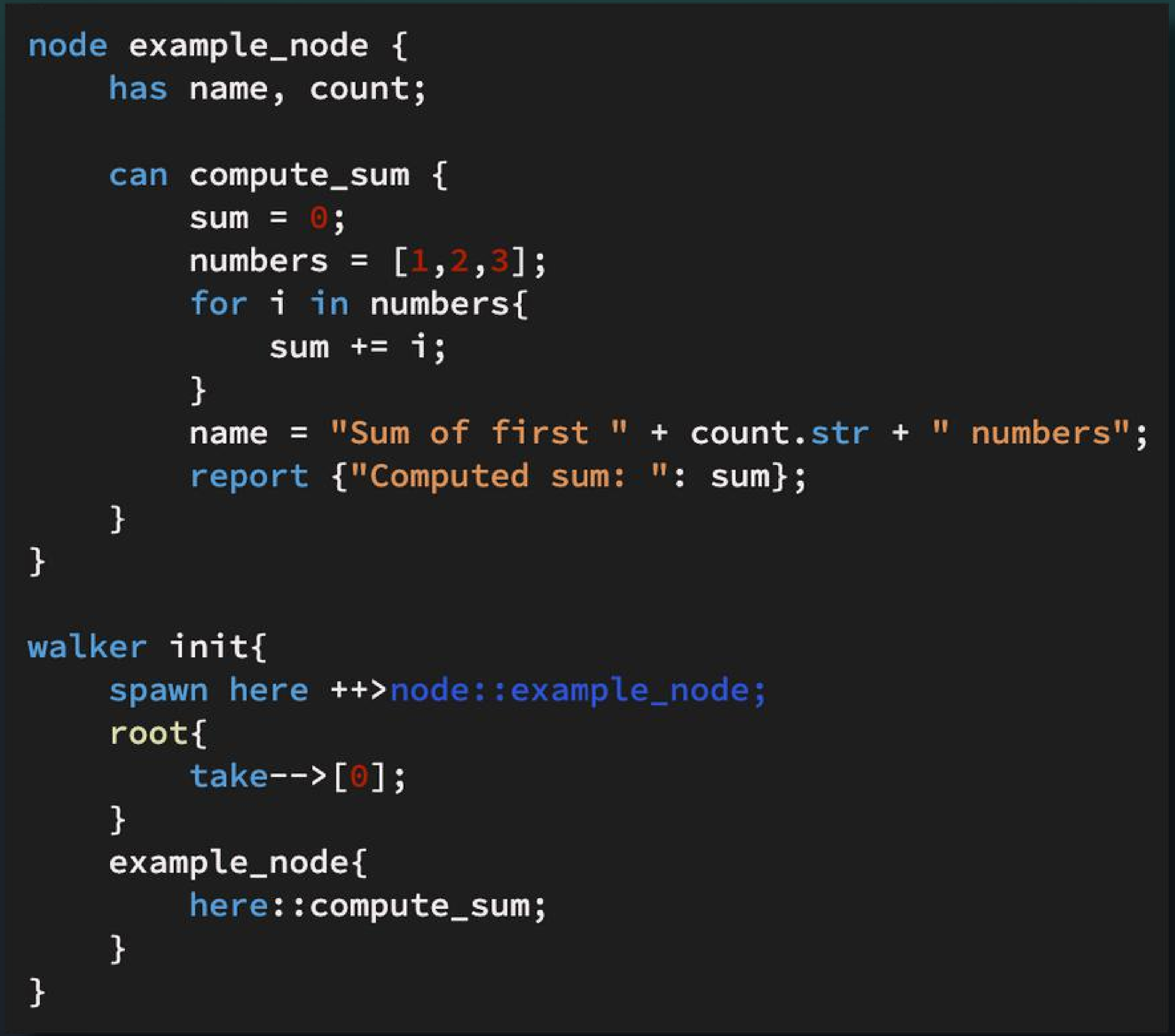
What are Abilities?

In a system with nodes, edges, and walkers, abilities are like methods that can be specified within the definition of a node, edge, or walker. They have their own body, enclosed within curly braces, and can interact with the context and local variables of the node/edge/walker they are attached to. Abilities do not have a return value and are like self-contained compute operations that can be used within the larger system.



JASECI

Example Code of Abilities

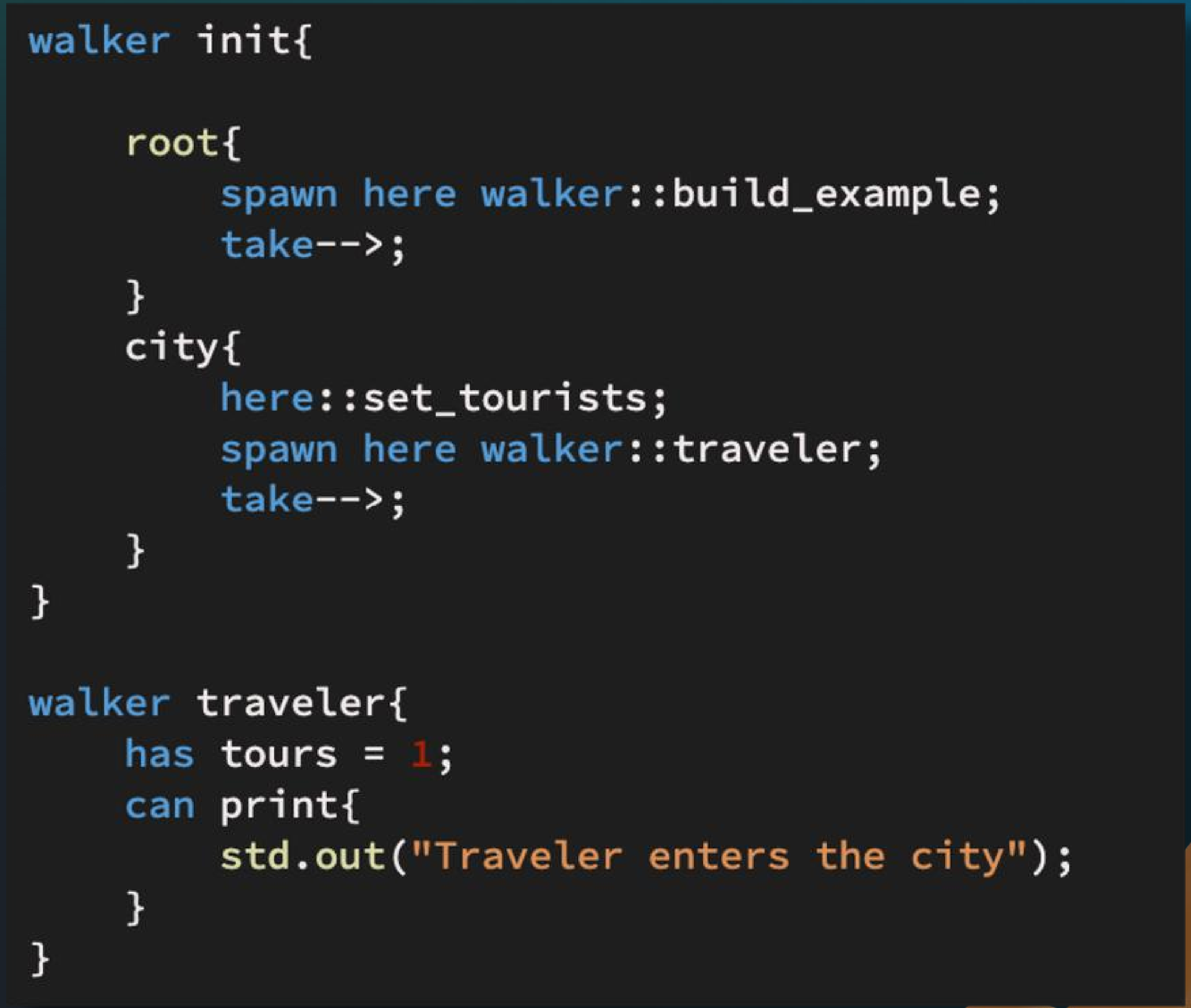


```
node example_node {
    has name, count;

    can compute_sum {
        sum = 0;
        numbers = [1,2,3];
        for i in numbers{
            sum += i;
        }
        name = "Sum of first " + count.str + " numbers";
        report {"Computed sum: ": sum};
    }
}

walker init{
    spawn here ++>node::example_node;
    root{
        take-->[0];
    }
    example_node{
        here::compute_sum;
    }
}
```

Nodes with Abilities



```
walker init{

    root{
        spawn here walker::build_example;
        take-->;
    }
    city{
        here::set_tourists;
        spawn here walker::traveler;
        take-->;
    }
}

walker traveler{
    has tours = 1;
    can print{
        std.out("Traveler enters the city");
    }
}
```

Walker with Ability

JASECI

For our app we are going to create 2 abilities

1. Given a node find the age by today
2. Given a node find how many days left to have his/her next birthday
3. Create a walker that walks through all the graph and finds upcoming birtdays

What are STD Actions?

Actions in Jac/Jaseci are similar to traditional function calls with return values, but their main purpose is to connect to external functionality that exists outside of the Jac/Jaseci system, typically in a Python module. Actions essentially serve as bindings to this external functionality, like library calls in other programming languages. They are implemented as a Jaseci action library that directly connects to the Python implementations of the external functions. Jaseci provides few functions out-of-the-box, those are STD Actions.

Lets Get to it

Next week

1. **Use AI Actions to add AI features to the app.**

ASSIGNMENT
Groups of 5, Need to add a new feature to the Ancestry Example. Need to Submit a Video of the feature in action and the Source code.

Allowed to make changes to every aspect of the app including nodes, edges, walkers, dataset.

Example - You can add a new field to the node saying hobbies, you can cluster people who have similar hobbies etc.