julep-ai-assignment

June 8, 2025

[39]:
```
pip install julep
```

```
Requirement already satisfied: julep in /usr/local/lib/python3.11/dist-packages
(2.10.0)
Requirement already satisfied: anyio<5,>=3.5.0 in
/usr/local/lib/python3.11/dist-packages (from julep) (4.9.0)
Requirement already satisfied: distro<2,>=1.7.0 in
/usr/local/lib/python3.11/dist-packages (from julep) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in
/usr/local/lib/python3.11/dist-packages (from julep) (0.28.1)
Requirement already satisfied: pydantic<3,>=1.9.0 in
/usr/local/lib/python3.11/dist-packages (from julep) (2.11.5)
Requirement already satisfied: python-dotenv<1.1,>=1.0 in
/usr/local/lib/python3.11/dist-packages (from julep) (1.0.1)
Requirement already satisfied: ruamel-yaml<0.19,>=0.18.6 in
/usr/local/lib/python3.11/dist-packages (from julep) (0.18.13)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-
packages (from julep) (1.3.1)
Requirement already satisfied: typing-extensions<5,>=4.10 in
/usr/local/lib/python3.11/dist-packages (from julep) (4.14.0)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-
packages (from anyio<5,>=3.5.0->julep) (3.10)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-
packages (from httpx<1,>=0.23.0->julep) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-
packages (from httpx<1,>=0.23.0->julep) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-
packages (from httpcore==1.*->httpx<1,>=0.23.0->julep) (0.16.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->julep) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in
/usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->julep)
(2.33.2)
Requirement already satisfied: typing-inspection>=0.4.0 in
/usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->julep) (0.4.1)
Requirement already satisfied: ruamel.yaml.clib>=0.2.7 in
/usr/local/lib/python3.11/dist-packages (from ruamel-yaml<0.19,>=0.18.6->julep)
(0.2.12)
```

[40]:
```
from julep import Julep
import time
import yaml

client = Julep(api_key="eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.
 ↪eyJzdWIiOiIwNzg1YTZiOS1jMDNlLTUwMTAtOGYyNC1mMTk0OWMxY2YwNmEiLCJlbWFpbCI6InNhdXJhYmhkZXNoZXR
 ↪wfkze-ZhcyyQHESr6mT7smbsh3SYJW2WuesLOG8lFtSqORrDaCJ_3ZnQQ1UdXqFhXz9QlxH2CqufIh4pAOA9Dw")
```

[41]:
```
agent = client.agents.create(name="Foodie Tour Agent", model="claude-3.
 ↪5-sonnet", about="Suggests foodie tours based on weather and local cuisine.")
```

[42]:
```
task_definition = {
    "name": "Foodie Tour Task",
    "description": "For a list of cities, suggests a foodie tour based on␣
 ↪weather and local cuisine.",
    "input_schema": {
        "type": "object",
        "properties": {
            "cities": {
                "type": "array",
                "items": {"type": "string"},
                "description": "List of city names to process"
            }
        }
    },
    "main": [
        {
            "prompt": [
                {
                    "role": "system",
                    "content": "You are a travel and food expert assistant."
                },
                {
                    "role": "user",
                    "content": "For each of these cities: {steps[0].input.
 ↪cities}\nFor each city:\n1. Check today's weather and suggest whether indoor␣
 ↪or outdoor dining is better.\n2. List 3 iconic local dishes.\n3. Suggest␣
 ↪top-rated restaurants for each dish.\n4. Create a one-day foodie tour␣
 ↪(breakfast, lunch, and dinner), assigning each dish to a meal. Write a fun,␣
 ↪engaging narrative for the tour, factoring in the weather."
                }
            ]
        }
    ]
}

task = client.tasks.create(agent_id=agent.id, **task_definition)
```

[43]:
```
execution = client.executions.create(
    task_id= "06845a28-ea1e-7e56-8000-848af246d2fe",
    input={"cities": ["Mumbai", "Paris", "Tokyo"]}
)

# Wait for the task to finish
while (result := client.executions.get(execution.id)).status not in␣
 ↪['succeeded', 'failed']:
    print("Working...")
    time.sleep(1)

if result.status == "succeeded":
    print(result.output)
else:
    print(f"Error: {result.error}")
```

```
Working…
Error: None
```