Weather agent

Example of PydanticAI with multiple tools which the LLM needs to call in turn to answer a question.

Demonstrates:

- tools
- agent dependencies
- streaming text responses

In this case the idea is a "weather" agent — the user can ask for the weather in multiple locations, the agent will use the `get_lat_lng` tool to get the latitude and longitude of the locations, then use the `get_weather` tool to get the weather for those locations.

Running the Example

To run this example properly, you might want to add two extra API keys **(Note if either key is missing, the code will fall back to dummy data, so they're not required)**:

- A weather API key from tomorrow.io set via `WEATHER_API_KEY`
- A geocoding API key from geocode.maps.co set via `GEO_API_KEY`

With dependencies installed and environment variables set, run:

pip

```
python -m pydantic_ai_examples.weather_agent
```

uv

```
uv run -m pydantic_ai_examples.weather_agent
```

Example Code

pydantic_ai_examples/weather_agent.py

```python
from __future__ import annotations as _annotations

import asyncio
import os
from dataclasses import dataclass
from typing import Any

import logfire
from devtools import debug
from httpx import AsyncClient

from pydantic_ai import Agent, ModelRetry, RunContext

# 'if-token-present' means nothing will be sent (and the example will work) if you don't have logfire configured
logfire.configure(send_to_logfire='if-token-present')


@dataclass
class Deps:
    client: AsyncClient
    weather_api_key: str | None
    geo_api_key: str | None


weather_agent = Agent(
    'openai:gpt-4o',
    system_prompt='Be concise, reply with one sentence.',
    deps_type=Deps,
    retries=2,
)


@weather_agent.tool
async def get_lat_lng(
    ctx: RunContext[Deps], location_description: str
) -> dict[str, float]:
    """Get the latitude and longitude of a location.

    Args:
        ctx: The context.
        location_description: A description of a location.
    """
    if ctx.deps.geo_api_key is None:
        # if no API key is provided, return a dummy response (London)
        return {'lat': 51.1, 'lng': -0.1}

    params = {
        'q': location_description,
        'api_key': ctx.deps.geo_api_key,
    }
    with logfire.span('calling geocode API', params=params) as span:
        r = await ctx.deps.client.get('https://geocode.maps.co/search', params=params)
        r.raise_for_status()
        data = r.json()
        span.set_attribute('response', data)

    if data:
        return {'lat': data[0]['lat'], 'lng': data[0]['lon']}
    else:
        raise ModelRetry('Could not find the location')


@weather_agent.tool
async def get_weather(ctx: RunContext[Deps], lat: float, lng: float) -> dict[str, Any]:
    """Get the weather at a location.

    Args:
        ctx: The context.
        lat: Latitude of the location.
        lng: Longitude of the location.
    """
```

```python
    if ctx.deps.weather_api_key is None:
        # if no API key is provided, return a dummy response
        return {'temperature': '21 °C', 'description': 'Sunny'}

    params = {
        'apikey': ctx.deps.weather_api_key,
        'location': f'{lat},{lng}',
        'units': 'metric',
    }
    with logfire.span('calling weather API', params=params) as span:
        r = await ctx.deps.client.get(
            'https://api.tomorrow.io/v4/weather/realtime', params=params
        )
        r.raise_for_status()
        data = r.json()
        span.set_attribute('response', data)

    values = data['data']['values']
    # https://docs.tomorrow.io/reference/data-layers-weather-codes
    code_lookup = {
        1000: 'Clear, Sunny',
        1100: 'Mostly Clear',
        1101: 'Partly Cloudy',
        1102: 'Mostly Cloudy',
        1001: 'Cloudy',
        2000: 'Fog',
        2100: 'Light Fog',
        4000: 'Drizzle',
        4001: 'Rain',
        4200: 'Light Rain',
        4201: 'Heavy Rain',
        5000: 'Snow',
        5001: 'Flurries',
        5100: 'Light Snow',
        5101: 'Heavy Snow',
        6000: 'Freezing Drizzle',
        6001: 'Freezing Rain',
        6200: 'Light Freezing Rain',
        6201: 'Heavy Freezing Rain',
        7000: 'Ice Pellets',
        7101: 'Heavy Ice Pellets',
        7102: 'Light Ice Pellets',
        8000: 'Thunderstorm',
    }
    return {
        'temperature': f'{values["temperatureApparent"]:0.0f}°C',
        'description': code_lookup.get(values['weatherCode'], 'Unknown'),
    }


async def main():
    async with AsyncClient() as client:
        # create a free API key at https://www.tomorrow.io/weather-api/
        weather_api_key = os.getenv('WEATHER_API_KEY')
        # create a free API key at https://geocode.maps.co/
        geo_api_key = os.getenv('GEO_API_KEY')
        deps = Deps(
            client=client, weather_api_key=weather_api_key, geo_api_key=geo_api_key
        )
        result = await weather_agent.run(
            'What is the weather like in London and in Wiltshire?', deps=deps
        )
        debug(result)
        print('Response:', result.data)


if __name__ == '__main__':
    asyncio.run(main())
```