Travel Planner Agent: Roman Urdu Mein Notes

1 Taaruf

Yeh notes ek working Travel Planner Agent ke code aur concepts (Agentic AI, dataclass, callable instructions, Runner class, generics) ko Roman Urdu mein samjhatay hain. Code practice ke liye hai aur pytest tests ke saath aata hai.

2 Concepts Overview

2.1 Dataclass

`Agent` class ek `dataclass` hai jo configuration (instructions, tools, model) store karta hai. Yeh code ko simple aur readable banata hai.

Misal: Dataclass ek admission form ki tarah hai jo fields khud organize karta hai.

2.2 Callable Instructions

Instructions static string ya callable function ho sakta hai, taake dynamic prompts ban sakein.

Misal: Callable instructions ek smart lesson plan hai jo student ke hisab se change hota hai.

2.3 Runner Class

`Runner` class agent ka engine hai jo user prompt aur tools ko use karke tasks execute karta hai. `run` method ek classmethod hai taake bina object banaye call ho sake.

2.4 Generics aur TContext

Generics multiple data types ke liye flexible code banate hain. `TContext` agent ke context (string ya dict) ko represent karta hai.

Misal: Generics ek multi-purpose bag hai jisme kuch bhi daal sakte ho.

3 Working Code

Yeh code ek Travel Planner Agent banata hai jo mausam check karta hai aur trip plan karta hai.

Code Example:

```python
from dataclasses import dataclass
from typing import TypeVar, Generic, List, Callable, Optional
import re

TContext = TypeVar("TContext")

class SDK:
    def get_weather(self, city: str) -> str:
        """Mock weather API jo city ka mausam deta hai."""
        return f"{city} ka mausam sunny hai aur temperature 25řC hai."

    def get_hotels(self, city: str, budget: int) -> str:
        """Mock hotel API jo budget ke hisab se hotels suggest karta hai."""
        if budget >= 100:
            return f"{city} mein budget hotels available hain: Hotel Star, Hotel Moon."
        return f"{city} mein is budget (${budget}) mein koi hotel nahi mila."

@dataclass
class Agent(Generic[TContext]):
    instructions: str | Callable[[TContext], str]
    tools: List[str]
    context: TContext
    model: str = "gpt-4"

    def __call__(self, user_input: str) -> str:
        """Agent ko callable banata hai taake direct call ho sake."""
        instructions = self.instructions(self.context) if callable(self.instructions) else self.instructions
        return f"{instructions}: {user_input} ko process kar raha hai."

class Runner:
    @classmethod
    def run(cls, agent: Agent, user_prompt: str, sdk: SDK) -> str:
        """Agent aur user prompt ke saath task execute karta hai."""
        print(f"Tools ke saath chal raha hai: {agent.tools}")

        instructions = agent.instructions(agent.context) if callable(agent.instructions) else agent.instructions

        city_match = re.search(r"\b(Karachi|Lahore|Paris|London)\b", user_prompt, re.IGNORECASE)
```

```
        budget_match = re.search(r"\$(\d+)", user_prompt)

        city = city_match.group(0) if city_match else "unknown city"
        budget = int(budget_match.group(1)) if budget_match else 100

        if "mausam" in user_prompt.lower():
            weather = sdk.get_weather(city)
            return f"{agent(user_prompt)}\n{weather}"

        elif "trip" in user_prompt.lower():
            weather = sdk.get_weather(city)
            hotels = sdk.get_hotels(city, budget)
            return f"{agent(user_prompt)}\nMausam: {weather}\nHotels: {hotels}"

        return agent(user_prompt)

def dynamic_instructions(context: dict) -> str:
    """Context ke hisab se dynamic instructions banata hai."""
    user = context.get("user", "Guest")
    budget = context.get("budget", 1000)
    return f"Tum ek planner ho {user} ke liye jiska budget ${budget} hai."

def main():
    sdk = SDK()

    agent_static = Agent[str](
        instructions="Tum ek travel planner ho",
        tools=["weather_api", "hotel_api"],
        context="User ek tourist hai",
        model="llama-3"
    )

    print("Static Agent Test:")
    response1 = Runner.run(agent_static, "Karachi ka mausam kya hai?", sdk)
    print(response1)
    print("\n")

    response2 = Runner.run(agent_static, "Paris ka trip plan karo $200 ke budget mein", sdk)
    print(response2)
    print("\n")

    agent_dynamic = Agent[dict](
        instructions=dynamic_instructions,
        tools=["weather_api", "hotel_api"],
        context={"user": "Ali", "budget": 200}
```

```
    )

    print("Dynamic Agent Test:")
    response3 = Runner.run(agent_dynamic, "Lahore ka trip plan karo $150 ke budget mein", sdk)
    print(response3)

if __name__ == "__main__":
    main()
```

4 Testing with Pytest

Yeh pytest code functionality ko verify karta hai.

Code Example:

```
import pytest
from travel_planner_agent import Agent, Runner, SDK, dynamic_instructions

@pytest.fixture
def sdk():
    """Mock SDK object for testing."""
    return SDK()

@pytest.fixture
def static_agent():
    """Static instructions wala agent."""
    return Agent[str](
        instructions="Tum ek travel planner ho",
        tools=["weather_api", "hotel_api"],
        context="User ek tourist hai",
        model="llama-3"
    )

@pytest.fixture
def dynamic_agent():
    """Dynamic instructions wala agent."""
    return Agent[dict](
        instructions=dynamic_instructions,
        tools=["weather_api", "hotel_api"],
        context={"user": "Ali", "budget": 200}
    )

def test_static_agent_weather(static_agent, sdk):
    """Test static agent with weather prompt."""
    response = Runner.run(static_agent, "Karachi ka mausam kya hai?", sdk)
    assert "Tum ek travel planner ho" in response
    assert "Karachi ka mausam sunny hai" in response

```

```python
def test_static_agent_trip_plan(static_agent, sdk):
    """Test static agent with trip planning prompt."""
    response = Runner.run(static_agent, "Paris ka trip plan karo $200 ke budget mein", sdk)
    assert "Tum ek travel planner ho" in response
    assert "Paris ka mausam sunny hai" in response
    assert "Hotel Star, Hotel Moon" in response

def test_dynamic_agent_trip_plan(dynamic_agent, sdk):
    """Test dynamic agent with trip planning prompt."""
    response = Runner.run(dynamic_agent, "Lahore ka trip plan karo $150 ke budget mein", sdk)
    assert "Tum ek planner ho Ali ke liye jiska budget $200 hai" in response
    assert "Lahore ka mausam sunny hai" in response
    assert "Hotel Star, Hotel Moon" in response

def test_callable_agent(dynamic_agent):
    """Test agent is callable."""
    response = dynamic_agent("Test input")
    assert "Tum ek planner ho Ali ke liye jiska budget $200 hai" in response
```

5 Practice Ke Liye Tips

- Code ko line-by-line samjhein aur run karein.
- Alag alag prompts try karein (e.g., London ka mausam kya hai?).
- New tools add karke SDK extend karein (e.g., flight booking API).
- Pytest cases ko run karke functionality verify karein.
- Code ko GitHub par push karke practice karein.
- Concepts ko doosron ko samjha kar clear karein.