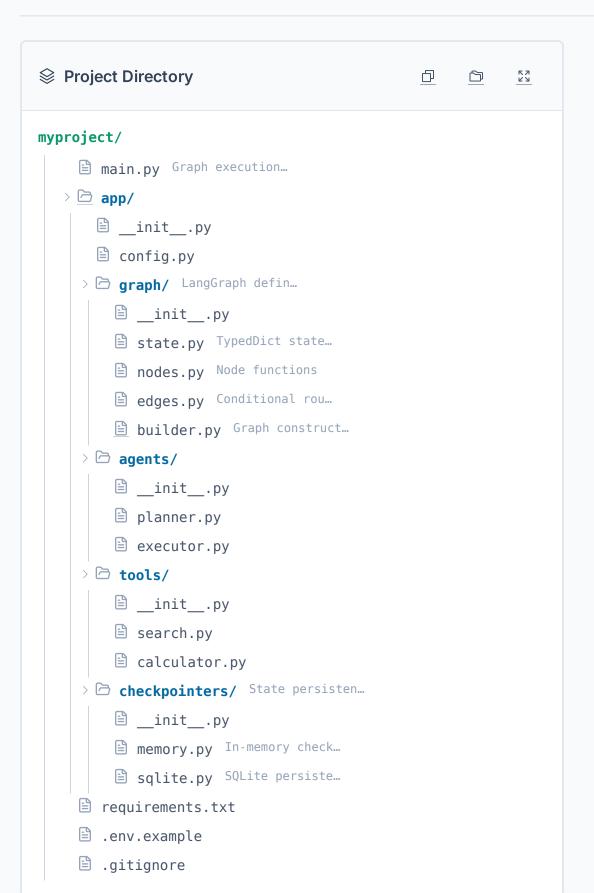
LangGraph Agent Workflow Structure

Stateful agent workflows with graph-based control flow and persistence.

#langchain #langgraph #python #agents #state-machine



○ Why This Structure?

LangGraph builds cyclic, stateful agent workflows as directed graphs. Unlike simple chains, it supports loops, conditional branching, and persistent state via checkpointers. The graph/ folder separates state schema, node logic, and edge routing.

☑ PNG

🗇 Сору

</> Prompt

PDF PDF

□ Key Directories

app/graph/state.py - TypedDict defining workflow state
app/graph/nodes.py - Functions that process state
app/graph/edges.py - Conditional routing logic
app/graph/builder.py - StateGraph construction

app/checkpointers/ - State persistence options

</> Graph Definition

```
from langgraph.graph import StateGraph

graph = StateGraph(AgentState)
graph.add_node("plan", plan_node)
graph.add_node("execute", execute_node)
graph.add_conditional_edges("plan", should_continue)
app = graph.compile(checkpointer=MemorySaver())
```

☑ When To Use This

- Complex agent workflows with loops
- Human-in-the-loop patterns
- Workflows requiring state persistence
- Multi-step planning and execution

₫ Trade-offs

Learning curve - Graph concepts take time to master

Debugging - Graph execution harder to trace than chains

Overhead - Simpler tasks don't need full graph