GRAPH BASED AI AGENT🤖

```
raph langchain-groq langchain_community #architecture Stateful graph --> State, node, edges>>> RATNAPRVA Mohapatra
```

```
  Downloading marshmallow-3.26.1-py3-none-any.whl.metadata (7.3 kB)
Collecting typing-inspect<1,>=0.4.0 (from dataclasses-json<0.7.0,>=0.6.7->langchain_community)
  Downloading typing_inspect-0.9.0-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from groq<1.0.0,>=0.30.
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from groq<1.0.0,>=0.30
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from groq<1.0.0,>=0.30
Requirement already satisfied: sniffio in /usr/local/lib/python3.12/dist-packages (from groq<1.0.0,>=0.30.0->langc
Requirement already satisfied: typing-extensions<5,>=4.10 in /usr/local/lib/python3.12/dist-packages (from groq<1.
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/python3.12/dist-packages (from langchain
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/python3.12/dist-packages (from langchai
Collecting ormsgpack>=1.10.0 (from langgraph-checkpoint<4.0.0,>=2.1.0->langgraph)
  Downloading ormsgpack-1.11.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: orjson>=3.10.1 in /usr/local/lib/python3.12/dist-packages (from langgraph-sdk<0.3.0
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmith
Requirement already satisfied: zstandard>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from langsmith>=0.1.1
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/local/lib/python3.12/dist-packages (from pydantic-set
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2->langc
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2-
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2-
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-packages (from SQLAlchemy<3,>=1.4->la
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->gr
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1,>
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/dist-packages (from jsonpatch<2.0.0,>
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7.0,>=0.6.7->langchain_commun
  Downloading mypy_extensions-1.1.0-py3-none-any.whl.metadata (1.1 kB)
Downloading langgraph-1.0.1-py3-none-any.whl (155 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 155.4/155.4 kB 9.5 MB/s eta 0:00:00
Downloading langchain_groq-0.3.8-py3-none-any.whl (16 kB)
Downloading langchain_community-0.3.31-py3-none-any.whl (2.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 80.8 MB/s eta 0:00:00
Downloading dataclasses_json-0.6.7-py3-none-any.whl (28 kB)
Downloading groq-0.33.0-py3-none-any.whl (135 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.8/135.8 kB 13.5 MB/s eta 0:00:00
Downloading langgraph_checkpoint-3.0.0-py3-none-any.whl (46 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.1/46.1 kB 4.5 MB/s eta 0:00:00
Downloading langgraph_prebuilt-1.0.1-py3-none-any.whl (28 kB)
Downloading langgraph_sdk-0.2.9-py3-none-any.whl (56 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.8/56.8 kB 5.8 MB/s eta 0:00:00
Downloading requests-2.32.5-py3-none-any.whl (64 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.7/64.7 kB 6.9 MB/s eta 0:00:00
Downloading marshmallow-3.26.1-py3-none-any.whl (50 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.9/50.9 kB 5.3 MB/s eta 0:00:00
Downloading ormsgpack-1.11.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (207 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.6/207.6 kB 21.4 MB/s eta 0:00:00
Downloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
Downloading mypy_extensions-1.1.0-py3-none-any.whl (5.0 kB)
Installing collected packages: requests, ormsgpack, mypy-extensions, marshmallow, typing-inspect, langgraph-sdk, g
  Attempting uninstall: requests
    Found existing installation: requests 2.32.4
    Uninstalling requests-2.32.4:
      Successfully uninstalled requests-2.32.4
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This be
google-colab 1.0.0 requires requests==2.32.4, but you have requests 2.32.5 which is incompatible.
Successfully installed dataclasses-json-0.6.7 groq-0.33.0 langchain-groq-0.3.8 langchain_community-0.3.31 langgrap
```

```
groq_api_key = "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"
```

```
from langchain_groq import  ChatGroq
```

```
llm = ChatGroq(groq_api_key=groq_api_key,model_name= 'llama-3.1-8b-instant')
```

```
from typing import Annotated
from typing_extensions import TypedDict
from langgraph.graph.message import add_messages
from langgraph.graph import StateGraph,START,END
```

```
class State(TypedDict):
  messages:Annotated[list,add_messages]
graph_builder = StateGraph(State)
```

```
graph_builder
```

```
<langgraph.graph.state.StateGraph at 0x7a8f8bd8da00>
```

```
def chatbot(state:State):
  return{"messages":llm.invoke(state['messages'])}
```

```
graph_builder.add_node('chatbot',chatbot)
```

```
<langgraph.graph.state.StateGraph at 0x7a8f8bd8da00>
```

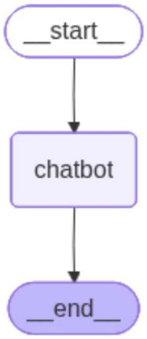
```
graph_builder
```

```
<langgraph.graph.state.StateGraph at 0x7a8f8bd8da00>
```

```
graph_builder.add_edge(START,'chatbot')
graph_builder.add_edge('chatbot',END)
```

```
<langgraph.graph.state.StateGraph at 0x7a8f8bd8da00>
```

```
graph_builder.compile()
```



```
graph = graph_builder.compile()
```

```
while True:
  user_input = input("user:")
  if user_input.lower() in ['quit','q']:
    print('Bye, thank you using Langgraph Framework')
    break
  for event in graph.stream({'messages':('user',user_input)}):
    print(event.values())
    for value in event.values():
      print(value['messages'])
      print('Assistant', value['messages'].content)
```

```
user:what the future of quantum ai , agentic ai
dict_values([{'messages': AIMessage(content="The future of Quantum AI and Agentive AI (also known as Autonomy or A
content="The future of Quantum AI and Agentive AI (also known as Autonomy or Autonomous AI) holds immense promise
Assistant The future of Quantum AI and Agentive AI (also known as Autonomy or Autonomous AI) holds immense promise
```

```
**Quantum AI:**

1. **Exponential scaling**: Quantum computing has the potential to exponentially scale up AI capabilities, enablin
        * Optimization problems (e.g., logistics, finance)
        * Machine learning (e.g., faster training, more accurate predictions)
        * Simulation (e.g., climate modeling, materials science)
2. **Quantum algorithms**: Researchers are developing quantum algorithms that exploit the unique properties of qua
3. **Quantum-inspired AI**: Even if we don't have fully functional quantum computers, researchers are exploring qu
4. **Challenges and risks**: Quantum AI also raises concerns about:
        * Error correction and noise reduction
        * Scalability and control
        * Security risks (e.g., potential for quantum computers to break certain encryption algorithms)

**Agentive AI (Autonomy):**

1. **Autonomous decision-making**: Agentive AI systems can make decisions without explicit human oversight, leadin
        * Robotics and automation
        * Healthcare and medical research
        * Financial analysis and trading
2. **Self-organization and adaptation**: Autonomy enables AI systems to learn from their environment, adapt to new
3. **Cognitive architectures**: Researchers are developing cognitive architectures that simulate human cognition,
4. **Challenges and risks**: Agentive AI raises concerns about:
        * Safety and accountability (e.g., who's responsible when AI makes a mistake?)
        * Fairness and transparency (e.g., how can we ensure AI systems are unbiased and explainable?)
        * Potential for unintended consequences (e.g., AI systems may pursue goals that are not aligned with human

**Convergence of Quantum AI and Agentive AI:**

1. **Hybrid approaches**: Researchers are exploring the combination of quantum computing and autonomy to create hy
2. **Quantum-inspired cognitive architectures**: Quantum-inspired AI techniques can be used to develop more effici
3. **Autonomous quantum systems**: Agentive AI can be used to develop autonomous quantum systems that can adapt an

**Timeline:**

* **Short-term (2025-2035)**: Expect incremental advancements in Quantum AI and Agentive AI, with a focus on devel
* **Mid-term (2035-2050)**: Quantum AI and Agentive AI are likely to converge, leading to the development of more
* **Long-term (2050+)**: As these technologies continue to advance, we can expect significant breakthroughs in are

Keep in mind that this is a high-level overview, and the actual timeline may vary depending on the pace of researc
user:What is Batch Gradient Descent?
dict_values([{'messages': AIMessage(content="**Batch Gradient Descent (BGD)**\n===================================
content="**Batch Gradient Descent (BGD)**\n=====================================\n\nBatch Gradient Descent (BGD) i
Assistant **Batch Gradient Descent (BGD)**
=====================================

Batch Gradient Descent (BGD) is an optimization algorithm used in machine learning to minimize the cost function o

**How BGD Works**
------------------

BGD involves the following steps:

1. **Initialization**: Initialize the model parameters (weights and biases) randoml
```

```
from groq import Groq

client = Groq(api_key=groq_api_key)

models = client.models.list()

print("Available Groq Models:")
for model in models.data:
  print(model.id)
```

```
Available Groq Models:
playai-tts-arabic
whisper-large-v3
openai/gpt-oss-120b
llama-3.3-70b-versatile
openai/gpt-oss-20b
meta-llama/llama-guard-4-12b
qwen/qwen3-32b
```

```
meta-llama/llama-4-maverick-17b-128e-instruct
meta-llama/llama-4-scout-17b-16e-instruct
groq/compound-mini
moonshotai/kimi-k2-instruct-0905
playai-tts
moonshotai/kimi-k2-instruct
whisper-large-v3-turbo
meta-llama/llama-prompt-guard-2-22m
groq/compound
llama-3.1-8b-instant
meta-llama/llama-prompt-guard-2-86m
allam-2-7b
```

Start coding or generate with AI.