



Running Agent as an Iterator

To demonstrate the `AgentExecutorIterator` functionality, we will set up a problem where an Agent must:

- Retrieve three prime numbers from a Tool
- Multiply these together.

In this simple problem we can demonstrate adding some logic to verify intermediate steps by checking whether their outputs are prime.

```
import os

import dotenv
import pydantic
from langchain.agents import AgentExecutor, initialize_agent, AgentType
from langchain.schema import AgentFinish
from langchain.agents.tools import Tool
from langchain import LLMMathChain
from langchain.chat_models import ChatOpenAI
```

API Reference:

- `AgentExecutor` from `langchain.agents`
- `initialize_agent` from `langchain.agents`
- `AgentType` from `langchain.agents`
- `AgentFinish` from `langchain.schema`
- `Tool` from `langchain.agents.tools`
- `ChatOpenAI` from `langchain.chat_models`

```
# Uncomment if you have a .env in root of repo contains OPENAI_API_KEY
# dotenv.load_dotenv("../..../..../.env")

# need to use GPT-4 here as GPT-3.5 does not understand, however hard
# you insist, that
# it should use the calculator to perform the final calculation
```

```
llm = ChatOpenAI(temperature=0, model="gpt-4")
llm_math_chain = LLMMathChain.from_llm(llm=llm, verbose=True)
```

Define tools which provide:

- The `n`th prime number (using a small subset for this example)
- The `LLMMathChain` to act as a calculator

```
primes = {998: 7901, 999: 7907, 1000: 7919}

class CalculatorInput(pydantic.BaseModel):
    question: str = pydantic.Field()

class PrimeInput(pydantic.BaseModel):
    n: int = pydantic.Field()

def is_prime(n: int) -> bool:
    if n <= 1 or (n % 2 == 0 and n > 2):
        return False
    for i in range(3, int(n**0.5) + 1, 2):
        if n % i == 0:
            return False
    return True

def get_prime(n: int, primes: dict = primes) -> str:
    return str(primes.get(int(n)))

async def aget_prime(n: int, primes: dict = primes) -> str:
    return str(primes.get(int(n)))

tools = [
    Tool(
        name="GetPrime",
        func=get_prime,
        description="A tool that returns the `n`th prime number",
        args_schema=PrimeInput,
        coroutine=aget_prime,
    ),
    Tool.from_function(
```

```

func=llm_math_chain.run,
name="Calculator",
description="Useful for when you need to compute mathematical
expressions",
args_schema=CalculatorInput,
coroutine=llm_math_chain.arun,
),
]

```

Construct the agent. We will use the default agent type here.

```

agent = initialize_agent(
    tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)

```

Run the iteration and perform a custom check on certain steps:

```

question = "What is the product of the 998th, 999th and 1000th prime
numbers?"

for step in agent.iter(question):
    if output := step.get("intermediate_step"):
        action, value = output[0]
        if action.tool == "GetPrime":
            print(f"Checking whether {value} is prime...")
            assert is_prime(int(value))
            # Ask user if they want to continue
            _continue = input("Should the agent continue (Y/n)?:\n")
            if _continue != "Y":
                break

```

```

> Entering new chain...
I need to find the 998th, 999th and 1000th prime numbers first.
Action: GetPrime
Action Input: 998
Observation: 7901
Thought:Checking whether 7901 is prime...
Should the agent continue (Y/n)?:
Y
I have the 998th prime number. Now I need to find the 999th prime

```

number.

Action: GetPrime

Action Input: 999

Observation: 7907

Thought:Checking whether 7907 is prime...

Should the agent continue (Y/n)?:

Y

I have the 999th prime number. Now I need to find the 1000th prime number.

Action: GetPrime

Action Input: 1000

Observation: 7919

Thought:Checking whether 7919 is prime...

Should the agent continue (Y/n)?:

Y

I have all three prime numbers. Now I need to calculate the product of these numbers.

Action: Calculator

Action Input: 7901 * 7907 * 7919

> Entering new chain...

7901 * 7907 * 7919```text

7901 * 7907 * 7919

```

...numexpr.evaluate("7901 \* 7907 \* 7919")...

Answer: 494725326233

> Finished chain.

Observation: Answer: 494725326233

Thought:Should the agent continue (Y/n)?:

Y

I now know the final answer

Final Answer: 494725326233

> Finished chain.