# 6.2.1 自定义代理

这一节我们介绍了如何创建自定义代理（agent）。

一个代理由两部分组成：

* 工具（tools）：代理可以使用的工具。
* 代理类本身：决定采取哪种行动。

我们将逐步介绍如何创建一个自定义代理。

Tool，AgentExecutor，BaseSingleActionAgent是从langchain.agents模块导入的类，用于创建自定义的Agent和工具。OpenAI和SerpAPIWrapper是从langchain模块导入的类，用于访问OpenAI的功能和SerpAPI的包。

from langchain.agents import Tool, AgentExecutor, BaseSingleActionAgent  
from langchain import OpenAI, SerpAPIWrapper

创建一个SerpAPIWrapper实例，然后将其run方法封装到一个Tool对象中。

search = SerpAPIWrapper()  
tools = [  
 Tool(  
 name="Search",  
 func=search.run,  
 description="useful for when you need to answer questions about current events",  
 return\_direct=True,  
 )  
]

定义了一个自定义的Agent类FakeAgent，这个类从BaseSingleActionAgent继承。该类定义了两个方法plan和aplan，这两个方法是Agent根据给定的输入和中间步骤来决定下一步要做什么的核心逻辑。

from typing import List, Tuple, Any, Union  
from langchain.schema import AgentAction, AgentFinish  
  
  
class FakeAgent(BaseSingleActionAgent):  
 """Fake Custom Agent."""  
  
 @property  
 def input\_keys(self):  
 return ["input"]  
  
 def plan(  
 self, intermediate\_steps: List[Tuple[AgentAction, str]], \*\*kwargs: Any  
 ) -> Union[AgentAction, AgentFinish]:  
 """Given input, decided what to do.  
  
 Args:  
 intermediate\_steps: Steps the LLM has taken to date,  
 along with observations  
 \*\*kwargs: User inputs.  
  
 Returns:  
 Action specifying what tool to use.  
 """  
 return AgentAction(tool="Search", tool\_input=kwargs["input"], log="")  
  
 async def aplan(  
 self, intermediate\_steps: List[Tuple[AgentAction, str]], \*\*kwargs: Any  
 ) -> Union[AgentAction, AgentFinish]:  
 """Given input, decided what to do.  
  
 Args:  
 intermediate\_steps: Steps the LLM has taken to date,  
 along with observations  
 \*\*kwargs: User inputs.  
  
 Returns:  
 Action specifying what tool to use.  
 """  
 return AgentAction(tool="Search", tool\_input=kwargs["input"], log="")

创建了一个FakeAgent的实例。

agent = FakeAgent()

创建了一个AgentExecutor的实例，该实例将使用前面定义的FakeAgent和tools来执行任务。

agent\_executor = AgentExecutor.from\_agent\_and\_tools(  
 agent=agent, tools=tools, verbose=True  
)

调用AgentExecutor的run方法来执行一个任务，任务是查询”2023年加拿大有多少人口”。

agent\_executor.run("How many people live in canada as of 2023?")

打印最终的结果。

> Entering new AgentExecutor chain...  
 The current population of Canada is 38,669,152 as of Monday, April 24, 2023, based on Worldometer elaboration of the latest United Nations data.  
   
 > Finished chain.  
  
  
  
  
  
 'The current population of Canada is 38,669,152 as of Monday, April 24, 2023, based on Worldometer elaboration of the latest United Nations data.'