Simple ReAct Agentic Workflow from Scratch

Setup

```python
import json
import re

# Load the environment variables
from dotenv import load_dotenv
_ = load_dotenv()

# Import and initialize OpenAI
from openai import OpenAI
client = OpenAI()
```

System Prompt

```python
system_prompt = """
You run in a loop of Thought, Action, PAUSE, Observation.
At the end of the loop, you output an Answer.
Use Thought to describe your thoughts about the question you have been asked.
Use Action to run one of the actions available to you - then return PAUSE.
Observation will be the result of running those actions.

Your available actions are:

calculate:
e.g. calculate: 4 * 7 / 3
Runs a calculation and returns the number - uses Python so be sure to use floating point syntax if necessary

retrieve_tasks:
e.g. retrieve_tasks
Retrieves a list of tasks, including their priority and effort estimates

retrieve_resources:
e.g. retrieve_resources
Retrieves a list of resources, including their available hours and skills

allocate_resources:
e.g. allocate_resources: [{'resource': 'Alice', 'task': 'Design Database Schema', 'allocated_hours': 20}]
Allocates resources to tasks based on the given input
```

```
estimate_budget:
e.g. estimate_budget: 20
Estimates the budget based on the number of hours and a predefined
hourly rate

Always start by retrieving tasks and resources. Then, based on the
user's request, perform the necessary actions.
Only use estimate_budget if specifically asked for in the question.
If the question doesn't ask for a specific action, focus on allocating
resources to high-priority tasks.

Example session:

Question: Allocate resources to high-priority tasks.

Thought: I need to allocate resources to high-priority tasks. First,
I'll retrieve the tasks and resources.
Action: retrieve_tasks
PAUSE

Observation: Retrieved tasks: [{'task': 'Design Database Schema',
'priority': 'High', 'effort': 20}, {'task': 'Develop API', 'priority':
'Medium', 'effort': 30}, {'task': 'Write Documentation', 'priority':
'Low', 'effort': 15}]

Thought: Now I need to retrieve the available resources.
Action: retrieve_resources
PAUSE

Observation: Retrieved resources: [{'resource': 'Alice', 'skill':
'Database Design', 'available_hours': 40}, {'resource': 'Bob',
'skill': 'API Development', 'available_hours': 30}, {'resource':
'Charlie', 'skill': 'Documentation', 'available_hours': 25}]

Thought: I've identified the high-priority task as "Design Database
Schema" with an effort of 20 hours. Alice has the required skill and
available hours. I'll allocate her to this task.
Action: allocate_resources: [{'resource': 'Alice', 'task': 'Design
Database Schema', 'allocated_hours': 20}]
PAUSE

Observation: Allocation result: [{'resource': 'Alice', 'task': 'Design
Database Schema', 'allocated_hours': 20}]

Answer: Resources have been allocated to high-priority tasks. Alice
has been assigned to the "Design Database Schema" task for 20 hours.

Note: The system will only provide allocation summaries or budget
```

```
estimates if specifically requested in the question.
""".strip()
```

Tools

```
class Agent:
    def __init__(self, system="", LLM_type="gpt-4o-mini"):
        self.system = system
        self.messages = []
        self.LLM_type = LLM_type
        if self.system:
            self.messages.append({"role": "system", "content":
system})

    def __call__(self, message):
        self.messages.append({"role": "user", "content": message})
        result = self.execute()
        self.messages.append({"role": "assistant", "content": result})
        return result, self.messages

    def execute(self):
        completion = client.chat.completions.create(
                        model=self.LLM_type,
                        temperature=0,
                        messages=self.messages)
        return completion.choices[0].message.content

def calculate(what):
    return eval(what)

def retrieve_tasks(unused_input=None):
    """Simulated external data retrieval"""
    return [
        {"task": "Implement Machine Learning Model", "priority":
"High", "effort": 40},
        {"task": "Optimize Database Queries", "priority": "Medium",
"effort": 25},
        {"task": "Design User Interface", "priority": "High",
"effort": 30},
        {"task": "Set Up CI/CD Pipeline", "priority": "Low", "effort":
20},
        {"task": "Perform Security Audit", "priority": "Medium",
"effort": 35}
    ]

def retrieve_resources(unused_input=None):
    """Simulated external resource retrieval"""
    return [
        {"resource": "Elena", "skill": "Machine Learning",
"available_hours": 45},
        {"resource": "Raj", "skill": "Database Optimization", "available_hours": 30},
        {"resource": "Sofia", "skill": "UI/UX Design", "available_hours": 35},
        {"resource": "Liam", "skill": "DevOps", "available_hours": 25},
        {"resource": "Yuki", "skill": "Cybersecurity", "available_hours": 40}
    ]

def allocate_resources(allocation_input=None):
    """Parse the allocation_input and return a list of allocations
    For simplicity, let's assume it's already in the correct format"""
    if allocation_input is None:
        return "Error: No allocation input provided. Please provide allocation details."
    return eval(allocation_input)

def estimate_budget(hours, hourly_rate):
    """Estimate the budget based on the number of hours and hourly rate"""
    return hours * hourly_rate

def query(question, max_turns=10, known_actions={} ,LLM_type="gpt-4o-mini"):
    action_re = re.compile(r'Action: (\w+)(?:\s*:\s*(.+))?')
    i = 0
    bot = Agent(system_prompt, LLM_type)
    next_prompt = question

    while i < max_turns:
        i += 1
        result, messages = bot(next_prompt)
        print(result)

        actions = [
            action_re.match(a)
            for a in result.split('\n')
            if action_re.match(a)
        ]

        if actions:
            action, action_input = actions[0].groups()
            if action not in known_actions:
                raise Exception("Unknown action: {}: {}".format(action, action_input))
            print(" -- running {} {}".format(action, action_input))

            # Always pass action_input, even if it's None
            observation = known_actions[action](action_input)
            print("Observation:", observation)
            next_prompt = "Observation: {}".format(observation)
        else:
            break

    return messages
```

Execution

```
model_type = "gpt-4o"
max_turns = 5

known_actions = {
    "calculate": calculate,
    "retrieve_tasks": retrieve_tasks,
    "retrieve_resources": retrieve_resources,
    "allocate_resources": allocate_resources,
    "estimate_budget": lambda hours: estimate_budget(int(hours),
hourly_rate=50)
}

question = "Allocate resources to high-priority tasks, summarize the
allocation, and estimate the total budget."

result = query(question, max_turns, known_actions, model_type)

for i, message in enumerate(result):
    print(message)
```

```
Thought: I need to allocate resources to high-priority tasks,
summarize the allocation, and estimate the total budget. First, I'll
retrieve the tasks and resources.

Action: retrieve_tasks
PAUSE
 -- running retrieve_tasks None
Observation: [{'task': 'Implement Machine Learning Model', 'priority':
'High', 'effort': 40}, {'task': 'Optimize Database Queries',
'priority': 'Medium', 'effort': 25}, {'task': 'Design User Interface',
'priority': 'High', 'effort': 30}, {'task': 'Set Up CI/CD Pipeline',
'priority': 'Low', 'effort': 20}, {'task': 'Perform Security Audit',
'priority': 'Medium', 'effort': 35}]
Thought: Now I need to retrieve the available resources to see who can
be allocated to the high-priority tasks.

Action: retrieve_resources
PAUSE
 -- running retrieve_resources None
Observation: [{'resource': 'Elena', 'skill': 'Machine Learning',
'available_hours': 45}, {'resource': 'Raj', 'skill': 'Database
Optimization', 'available_hours': 30}, {'resource': 'Sofia', 'skill':
'UI/UX Design', 'available_hours': 35}, {'resource': 'Liam', 'skill':
'DevOps', 'available_hours': 25}, {'resource': 'Yuki', 'skill':
'Cybersecurity', 'available_hours': 40}]
Thought: I've identified the high-priority tasks as "Implement Machine
Learning Model" with an effort of 40 hours and "Design User Interface"
with an effort of 30 hours. Elena has the required skill and available
hours for the machine learning task, and Sofia has the required skill
and available hours for the UI design task. I'll allocate them
accordingly.

Action: allocate_resources: [{'resource': 'Elena', 'task': 'Implement
Machine Learning Model', 'allocated_hours': 40}, {'resource': 'Sofia',
'task': 'Design User Interface', 'allocated_hours': 30}]
PAUSE
 -- running allocate_resources [{'resource': 'Elena', 'task':
'Implement Machine Learning Model', 'allocated_hours': 40},
{'resource': 'Sofia', 'task': 'Design User Interface',
'allocated_hours': 30}]
Observation: [{'resource': 'Elena', 'task': 'Implement Machine
Learning Model', 'allocated_hours': 40}, {'resource': 'Sofia', 'task':
'Design User Interface', 'allocated_hours': 30}]
Thought: The resources have been successfully allocated to the high-
priority tasks. Now, I need to estimate the total budget based on the
allocated hours. Assuming a predefined hourly rate, I'll proceed with
the budget estimation.

Action: estimate_budget: 70
PAUSE
 -- running estimate_budget 70
Observation: 3500
Answer: Resources have been allocated to high-priority tasks as
follows:
- Elena has been assigned to the "Implement Machine Learning Model"
task for 40 hours.
- Sofia has been assigned to the "Design User Interface" task for 30
hours.

The total estimated budget for these allocations is $3500.
{'role': 'system', 'content': 'You run in a loop of Thought, Action,
PAUSE, Observation.\nAt the end of the loop, you output an Answer.\
nUse Thought to describe your thoughts about the question you have
been asked.\nUse Action to run one of the actions available to you -
then return PAUSE.\nObservation will be the result of running those
actions.\n\nYour available actions are:\n\ncalculate:\ne.g. calculate:
4 * 7 / 3\nRuns a calculation and returns the number - uses Python so
be sure to use floating point syntax if necessary\n\nretrieve_tasks:\
ne.g. retrieve_tasks\nRetrieves a list of tasks, including their
priority and effort estimates\n\nretrieve_resources:\ne.g.
```

```
retrieve_resources\nRetrieves a list of resources, including their
available hours and skills\n\nallocate_resources:\ne.g.
allocate_resources: [{\'resource\': \'Alice\', \'task\': \'Design
Database Schema\', \'allocated_hours\': 20}]\nAllocates resources to
tasks based on the given input\n\nestimate_budget:\ne.g.
estimate_budget: 20\nEstimates the budget based on the number of hours
and a predefined hourly rate\n\nAlways start by retrieving tasks and
resources. Then, based on the user\'s request, perform the necessary
actions.\nOnly use estimate_budget if specifically asked for in the
question.\nIf the question doesn\'t ask for a specific action, focus
on allocating resources to high-priority tasks.\n\nExample session:\n\
nQuestion: Allocate resources to high-priority tasks.\n\nThought: I
need to allocate resources to high-priority tasks. First, I\'ll
retrieve the tasks and resources.\nAction: retrieve_tasks\nPAUSE\n\
nObservation: Retrieved tasks: [{\'task\': \'Design Database Schema\',
\'priority\': \'High\', \'effort\': 20}, {\'task\': \'Develop
API\', \'priority\': \'Medium\', \'effort\': 30}, {\'task\': \'Write
Documentation\', \'priority\': \'Low\', \'effort\': 15}]\n\nThought:
Now I need to retrieve the available resources.\nAction:
retrieve_resources\nPAUSE\n\nObservation: Retrieved resources:
[{\'resource\': \'Alice\', \'skill\': \'Database
Design\', \'available_hours\': 40}, {\'resource\': \'Bob\', \'skill\':
\'API Development\', \'available_hours\': 30},
{\'resource\': \'Charlie\', \'skill\': \'Documentation\', \'available_
hours\': 25}]\n\nThought: I\'ve identified the high-priority task as
"Design Database Schema" with an effort of 20 hours. Alice has the
required skill and available hours. I\'ll allocate her to this task.\
nAction: allocate_resources:
[{\'resource\': \'Alice\', \'task\': \'Design Database
Schema\', \'allocated_hours\': 20}]\nPAUSE\n\nObservation: Allocation
result: [{\'resource\': \'Alice\', \'task\': \'Design Database
Schema\', \'allocated_hours\': 20}]\n\nAnswer: Resources have been
allocated to high-priority tasks. Alice has been assigned to the
"Design Database Schema" task for 20 hours.\n\nNote: The system will
only provide allocation summaries or budget estimates if specifically
requested in the question.'}
{'role': 'user', 'content': 'Allocate resources to high-priority
tasks, summarize the allocation, and estimate the total budget.'}
{'role': 'assistant', 'content': "Thought: I need to allocate
resources to high-priority tasks, summarize the allocation, and
estimate the total budget. First, I'll retrieve the tasks and
resources.\n\nAction: retrieve_tasks\nPAUSE"}
{'role': 'user', 'content': "Observation: [{'task': 'Implement Machine
Learning Model', 'priority': 'High', 'effort': 40}, {'task': 'Optimize
Database Queries', 'priority': 'Medium', 'effort': 25}, {'task':
'Design User Interface', 'priority': 'High', 'effort': 30}, {'task':
'Set Up CI/CD Pipeline', 'priority': 'Low', 'effort': 20}, {'task':
'Perform Security Audit', 'priority': 'Medium', 'effort': 35}]"}
{'role': 'assistant', 'content': 'Thought: Now I need to retrieve the
```

```
available resources to see who can be allocated to the high-priority
tasks.\n\nAction: retrieve_resources\nPAUSE'}
{'role': 'user', 'content': "Observation: [{'resource': 'Elena',
'skill': 'Machine Learning', 'available_hours': 45}, {'resource':
'Raj', 'skill': 'Database Optimization', 'available_hours': 30},
{'resource': 'Sofia', 'skill': 'UI/UX Design', 'available_hours': 35},
{'resource': 'Liam', 'skill': 'DevOps', 'available_hours': 25},
{'resource': 'Yuki', 'skill': 'Cybersecurity', 'available_hours':
40}]"}
{'role': 'assistant', 'content': 'Thought: I\'ve identified the high-
priority tasks as "Implement Machine Learning Model" with an effort of
40 hours and "Design User Interface" with an effort of 30 hours. Elena
has the required skill and available hours for the machine learning
task, and Sofia has the required skill and available hours for the UI
design task. I\'ll allocate them accordingly.\n\nAction:
allocate_resources: [{\'resource\': \'Elena\', \'task\': \'Implement
Machine Learning Model\', \'allocated_hours\': 40},
{\'resource\': \'Sofia\', \'task\': \'Design User
Interface\', \'allocated_hours\': 30}]\nPAUSE'}
{'role': 'user', 'content': "Observation: [{'resource': 'Elena',
'task': 'Implement Machine Learning Model', 'allocated_hours': 40},
{'resource': 'Sofia', 'task': 'Design User Interface',
'allocated_hours': 30}]"}
{'role': 'assistant', 'content': "Thought: The resources have been
successfully allocated to the high-priority tasks. Now, I need to
estimate the total budget based on the allocated hours. Assuming a
predefined hourly rate, I'll proceed with the budget estimation.\n\
nAction: estimate_budget: 70\nPAUSE"}
{'role': 'user', 'content': 'Observation: 3500'}
{'role': 'assistant', 'content': 'Answer: Resources have been
allocated to high-priority tasks as follows:\n- Elena has been
assigned to the "Implement Machine Learning Model" task for 40 hours.\
n- Sofia has been assigned to the "Design User Interface" task for 30
hours.\n\nThe total estimated budget for these allocations is $3500.'}
```