

Test the api you plan on using. Note the path of the response you wish to capture. In this case it is `current.temp_f`

The screenshot shows a REST client interface with a GET request to `http://api.weatherapi.com/v1/current.json?key=a79f6538721d4c48beb170058252801&q=London&aqi=no`. The request is sent, and the response is a 200 OK status with a response time of 252 ms and a body size of 1.09 KB. The response body is displayed in JSON format, showing location details for London and current weather information.

Query Params

Key	Value	Description
key	a79f6538721d4c48beb170058252801	
q	London	
aqi	no	

```
1 {
2   "location": {
3     "name": "London",
4     "region": "City of London, Greater London",
5     "country": "United Kingdom",
6     "lat": 51.5171,
7     "lon": -0.1062,
8     "tz_id": "Europe/London",
9     "localtime_epoch": 1738083935,
10    "localtime": "2025-01-28 17:05"
11  },
12  "current": {
13    "last_updated_epoch": 1738083600,
14    "last_updated": "2025-01-28 17:00",
15    "temp_c": 9.2,
16    "temp_f": 48.6,
17    "is_day": 0,
```

Deploy a model that allow function calling.

gpt-4o

Help

Deploy

Fine-tune

Details

Benchmarks

Existing deployments

Code samples

License

gpt-4o offers a shift in how AI models interact with multimodal inputs. By seamlessly combining text, images, and audio, gpt-4o provides a richer, more engaging user experience. Matching the intelligence of gpt-4 turbo, it is remarkably more efficient, delivering text at twice the speed and at half the cost. Additionally, GPT-4o exhibits the highest vision performance and excels in non-English languages compared to previous OpenAI models.

gpt-4o is engineered for speed and efficiency. Its advanced ability to handle complex queries with minimal resources can translate into cost savings and performance.

The introduction of gpt-4o opens numerous possibilities for businesses in various sectors:

- Enhanced customer service:** By integrating diverse data inputs, gpt-4o enables more dynamic and comprehensive customer support interactions.
- Advanced analytics:** Leverage gpt-4o's capability to process and analyze different types of data to enhance decision-making and uncover deeper insights.
- Content innovation:** Use gpt-4o's generative capabilities to create engaging and diverse content formats, catering to a broad range of consumer preferences.

Updates

gpt-4o-2024-11-20: this is the latest version of gpt-4o. Supports all previous output size (16384) and features such as:

- Text, image processing
- JSON Mode
- parallel function calling
- Enhanced accuracy and responsiveness
- Parity with English text and coding tasks compared to GPT-4 Turbo with Vision
- Superior performance in non-English languages and in vision tasks
- Support for enhancements
- Support for complex structured outputs.

Resources

- "Hello gpt-4o" (OpenAI announcement)
- Introducing gpt-4o: OpenAI's new flagship multimodal model now in preview on Azure

See less

Quick facts

OpenAI

gpt-4o

chat-completion

Last trained

September 2023

Pricing

See pricing

Benchmarks

0.85 AI quality

Quality index

4.38 USD per 1M tokens

Estimated cost

Model ID

Reference this model ID when deploying the model in code

azureml://registries/azure-openai/models/gpt-4o/versions/2024-11-20

## Within Prompt flow create a chat flow

### Create a new flow

#### Create by type

Standard flow

Harness the power of Large Language Models, customized Python code, and more to craft your tailored prompt flow. Test the flow using custom datasets and seamlessly deploy as an endpoint for easy integration.

Create

Chat flow

On top of the standard flow, this option provides the chat history support and a user-friendly chat interface in the authoring/debugging UI.

Create

Evaluation flow

Create an evaluation flow to measure how well the output matches the expected criteria and goals.

Create

#### Explore gallery

All Standard flow Chat flow Evaluation flow

Chat

Multi-Round Q&A on Your Data

Create a chatbot that uses LLM and data from your own indexed files to ground multi-round question and answering capabilities in enterprise chat scenarios.

View detail

Clone

Standard

Q&A on Your Data

Use LLM and data from your own indexed files to ground multi-round question and answering capabilities.

View detail

Clone

Standard

Web Classification

Use LLM to classify URLs into multiple categories.

View detail

Clone

Chat

Chat with

Create a chatbot that ground the responses

View detail

Clone

Erase every node but input and output and start the computer sessions. Once started add a prompt. Define the connection and model. Then validate. Configuration should look like below.

Once completed, run the flow without function calling and this should be the output. Note the llm couldn't help with the question.

## Outputs

[Outputs](#) [Logs](#) [Metrics](#) [Trace](#)

Export

Search

Details	#	inputs.question	Status	output
	0	What is the current weather in London?	Completed	I'm sorry, I can't provide real-time data like the current weather in London right now. However, you can check the latest weather information by using a trusted weather service like [BBC Weather](https://www.bbc.co.uk/weather), [AccuWeather](https://www.accuweather.com/), or a weather app on your device.

Add another llm and add the following json below to function block. This json defines the schema for the function. The description is of critical importance.

Function Call-01-28-2025-1... Completed View batch runs View outputs

Flow + LLM + Prompt + Python + More tools Raw file mode Wrap text Diff mode

Inputs Validation and parse input Validation and parsing input completed successfully.

Name	Type	Value
question	string	{inputs.question}

Activate config

Outputs Duration: 0.04s Tokens: 0 Completed View full output

mytemplim Connection ai-ahubdev001815404591378.acai Api chat

deployment\_name gpt-4o temperature 1 stop max\_tokens 100 response\_format Please choose an option

Advanced

Function calling

Name	Type	Value
functions	list	[{"name": "get_temperature", "description": "get temperature of a particular city", "parameters": {"type": "object", "properties": {"city": {"type": "string", "description": "The city e.g. Orlando"}} }}]</td
function_call	string, object	

Prompt Referring to: mytemplim.jinja2

```
1 & system:
2 You are a helpful assistant.
3 & user:
4 {{question}}
```

Inputs Validation and parse input

Input section is empty. This may be due to one of the following reasons:  
 1. You haven't clicked "validate and parse input" button to parse the input.  
 2. Your function or prompt doesn't require any input.

Activate config

Files

Graph

```

graph TD
    inputs --> mytemplim[mytemplim Completed]
    mytemplim --> outputs
  
```

JSON Function Schema. Note the json is inclosed in an list

```

[[
  {
    "name": "get_temperature",
    "description": "get temperature of a particular city",
    "parameters": {
      "type": "object",
      "properties": {
        "city": {
          "type": "string",
          "description": "The city e.g. Orlando"
        }
      }
    }
  }
],
{
  "required": ["city"]
}
]]

```

For the function call field add auto. This tells the llm to decide if and when to use the tool.

mytemplim Show variants Generate variants

Connection ai-ahubdev001815404591378.acai Api chat

deployment\_name gpt-4o temperature 1 stop max\_tokens 100 response\_format Please choose an option

Advanced

Function calling

Name	Type	Value
functions	list	[{"name": "get_temperature", "description": "get temperature of a particular city", "parameters": {"type": "object", "properties": {"city": {"type": "string", "description": "The city e.g. Orlando"}}}]
function_call	string, object	auto

Within the flow add a custom python node and delete the content.

The screenshot shows the Flow editor interface. On the left, the 'Function calling' node is expanded, showing a table with columns 'Name', 'Type', and 'Value'. The 'functions' row has a value of '["name": "get\_temperature", "description": "get temperature of a particular city", "parameters": {"type": "object", "properties": {"city": {"type": "string", "description": "The city to get the temperature of"}}, "required": ["city"]}'. Below this, the 'Prompt' section shows a system prompt and a user prompt. The 'Inputs' section shows a 'question' input. On the right, the 'Graph' panel shows a flow diagram with nodes: 'input' -> 'myllm' (Completed) -> 'mytemppllm' -> 'gettempcall' -> 'output'.

Add the snippet of python below to the code block and validate. Once valid update `response_message` to `mytemppllm.output`

The screenshot shows the Flow editor interface. The 'gettempcall' node is expanded, showing a code block with the following Python code:

```
1 from promptflow.core import tool
2 import requests
3 import json
4
5 # api_key
6 api_key = "9790538723464486b170050252891"
7 # url of api
8 url = "https://api.weatherapi.com/v2/current.json?u="
9
10 # a dot name should match description and import should match
11 def get_temperature(city: str) -> str:
12     url = url + city + "&key=" + api_key
13     response = requests.get(url)
14     # json decoding Get response to url
15     return "Temperature: " + str(response.json()["temp_c"])
16
17 @tool
18 def run_function(response_message: dict) -> str:
19     function_call = response_message.get("function_call", None)
20     if function_call and "name" in function_call and "arguments" in function_call:
21         function_name = function_call["name"]
22         function_args = json.loads(function_call["arguments"])
23         print(function_args)
24         result = globals()[function_name](**function_args)
25     else:
26         print("No function call")
27     if isinstance(response_message, dict):
28         result = response_message.get("content", "")
29     else:
30         result = response_message
31     return result
```

The 'Inputs' section shows a 'response\_message' input of type 'object' with a value of '["mytemppllm.output"]'. On the right, the 'Graph' panel shows a flow diagram with nodes: 'mytemppllm' -> 'gettempcall' -> 'myllm' (Completed) -> 'output'.

Python snippet. Update with your api key

```
from promptflow.core import tool
import requests
import json
```

```
# api_key
```

```

api_key="xxxxxxxxxxxxxxxxxxxx"
# url of api
api_url="http://api.weatherapi.com/v1/current.json?q="

# def name should match description and import should match
def get_temperature(city: str)-> str:
    url=str(api_url) + str(city) + "&key=" + str(api_key)

    response = requests.get(url).json() #sending Get response to url

    return "Temperature: " + str(response["current"]["temp_f"])

@tool
def run_functions(response_message: dict)->str:
    function_call = response_message.get("function_call", None)
    if function_call and "name" in function_call and "arguments" in function_call:
        function_name = function_call["name"]
        function_args = json.loads(function_call["arguments"])
        print(function_args)
        result = globals()[function_name](**function_args)
    else:
        print("No function call")
        if isinstance(response_message, dict):
            result = response_message.get("content", "")
        else:
            result = response_message

    return result

```

Add another llm to parse the response from the function call and compare to the original question. Use the snippet below for the prompt.

myfinalllm

Connection: ai-ahubdev001815404591378\_acai | Api: chat

deployment\_name: gpt-4o | temperature: 1 | stop: | max\_tokens: 100 | response\_format: Please choose an option

> Advanced

> Function calling

> Prompt ☐ Referring to: myfinalllm.jinja2

```

1 # system:
2 You are a helpful AI assistant
3 You will be provided with the user question and a response to that user question
4 Make sure the response is relevant to the user's question
5 Finally present it as an answer to the user's question in the best format possible
6
7 # user:
8 {{question}}
9 {{llm_response}}
10

```

> Inputs ☐ Validate and parse input ☒ Validation and parsing input completed successfully.

Name	Type	Value
llm_response	string	
question	string	

> Activate config

# system:

You are a helpful AI assistant

You will be provided with the user question and a response to that user question

Make sure the response is relevant to the user's question

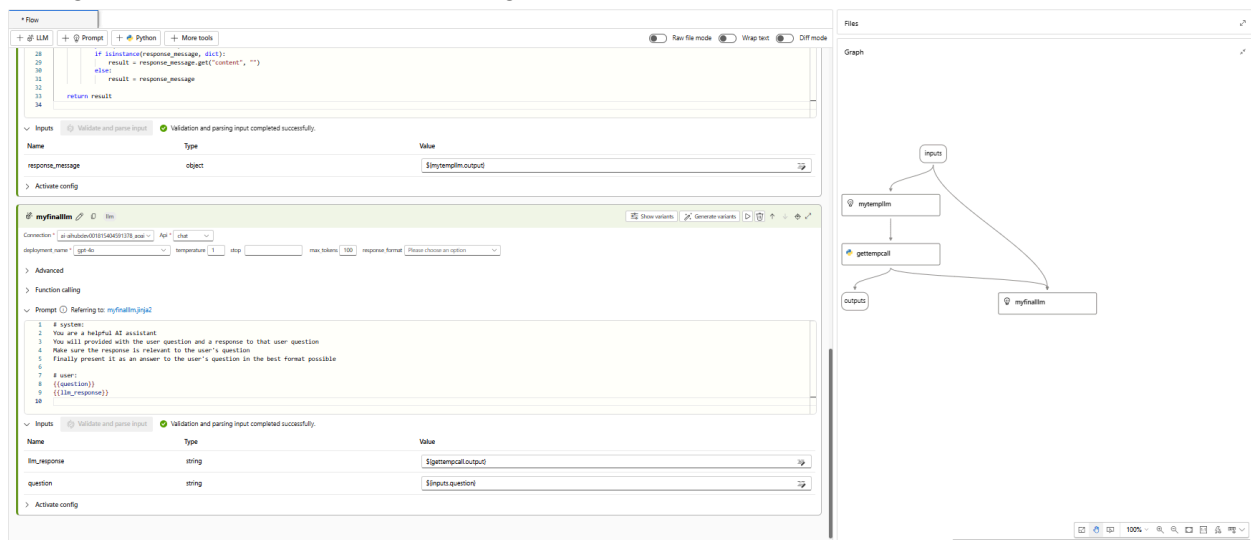
Finally present it as an answer to the user's question in the best format possible

# user:

{{question}}

{{llm\_response}}

Configure the last llm to receive the original question and the output from the function call



Run and see if successful

variant\_0 Run completed.

Function Call-01-28-2025-1... Completed View batch runs View outputs Clone Save Deploy Evaluate Compute session running Run

Flow

+ LLM + Prompt + Python + More tools Raw file mode

Outputs Duration 0.26s Completed View full output

Input Output Trace Logs

```
[{"system_metrics": {"duration": 0.259592}, "output": "Temperature: 46.9"}]
```

myfinalllm llm Show variants Generate variants ⏮ ⏪ ⏩ ⏭

Connection ai-aihubdev001815404591378\_aiai Api chat

deployment\_name gpt-4o temperature 1 stop

max\_tokens 1000 response\_format Please choose an option

Advanced

Function calling

Prompt Referring to: myfinalllm.jinja2

```
1 # system:
2 You are a helpful AI assistant
```

Files

Graph

inputs

mytempilim Completed

gettempcall Completed

myfinalllm Completed

outputs

100%

If successful you can see the question and output

Outputs ×

Outputs Logs Metrics Trace

Export Expand all cells

Details	#	inputs.question	Status	output
<span>0</span>	0	What is the current weather in London?	<span>Completed</span>	Temperature: 46.9