Defining function calling

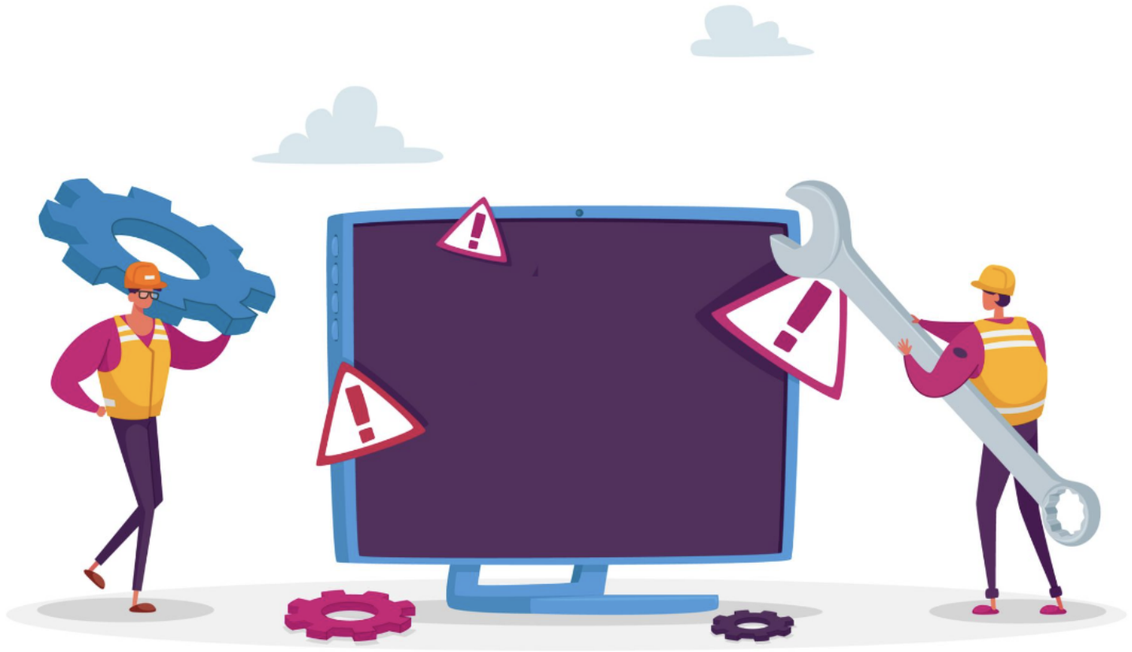
DEVELOPING AI SYSTEMS WITH THE OPENAI API

Francesca Donadoni
Curriculum Manager, DataCamp

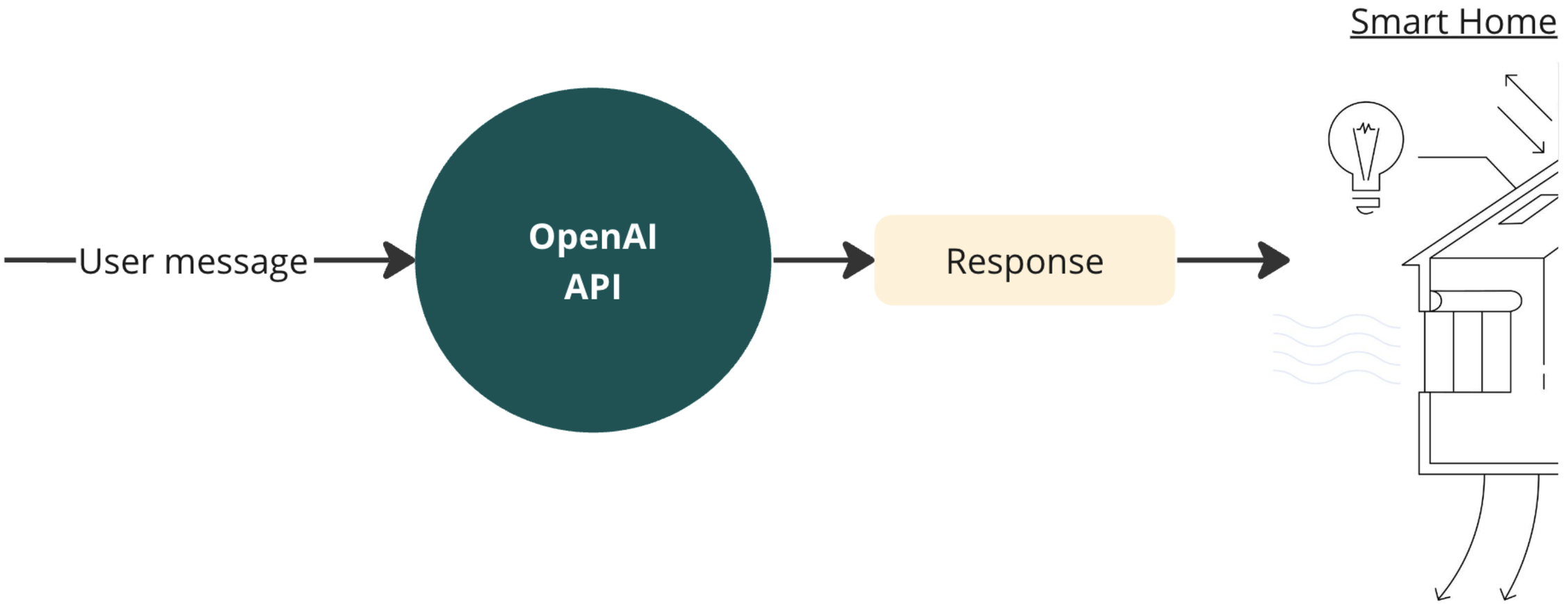
datacamp

OpenAI's tools

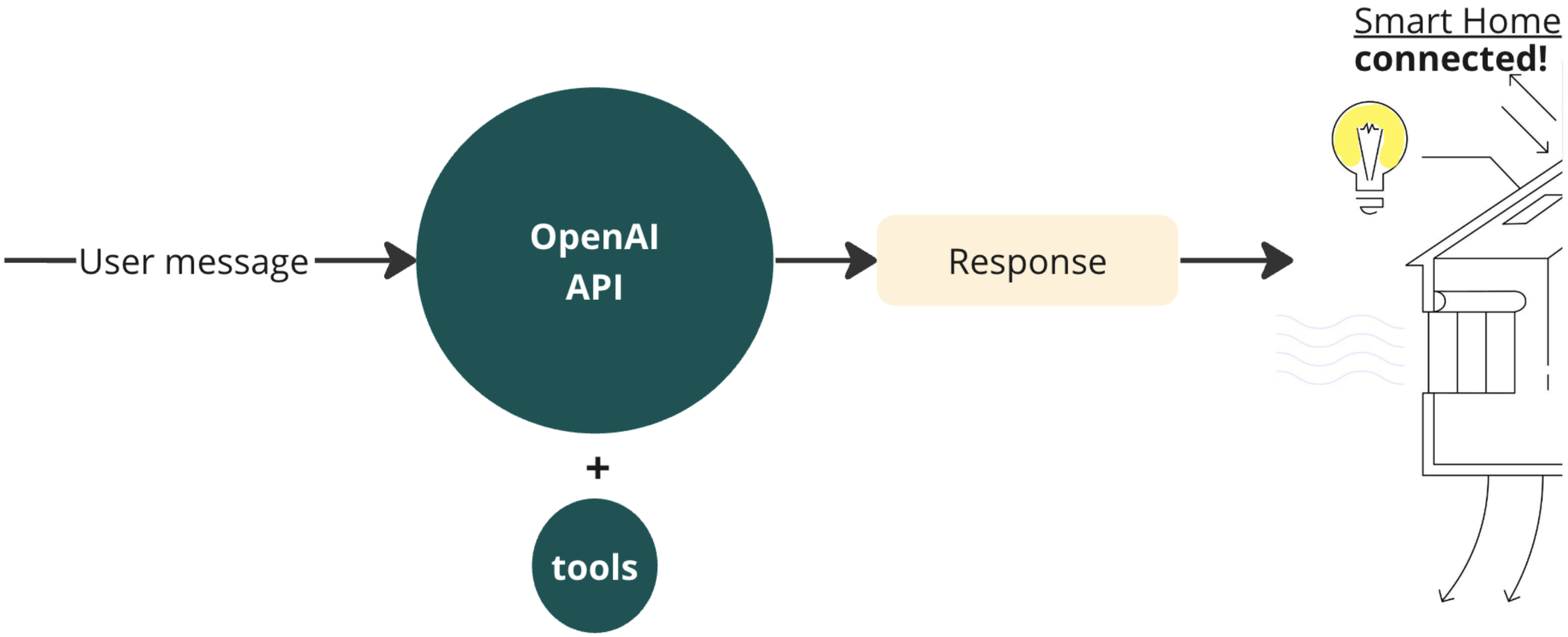
- Can be used to return more specific information
- Can define a more precise structure
- Enhances the capabilities of the API call



OpenAI's tools



OpenAI's tools



What is function calling?

```
response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
   {"role": "user",
   "content": "Please write down four trees with their scientific names in json."}
 ],
  response_format={"type": "json_object"}
)
print(response.choices[0].message.content)
```

```
{"trees": [{"common_name": "Oak", "scientific_name": "Quercus"}, { "common_name":
"Maple", "scientific_name": "Acer"}, {"common_name": "Pine", "scientific_name":
"Pinus"}, {"common_name": "Birch", "scientific_name": "Betula"}]}
```

Why use function calling?

```
{"trees": [
          {"common_name": "Oak",
          "scientific_name": "Quercus"},
          { "common_name": "Maple",
          "scientific_name": "Acer"},
          {"common_name": "Pine",
          "scientific_name": "Pinus"},
          {"common_name": "Birch",
          "scientific_name": "Betula"}
          ]
}
```

```
{"Oak": "Quercus",
 "Maple": "Acer",
 "Pine": "Pinus",
 "Birch": "Betula"}
```

Use cases for function calling

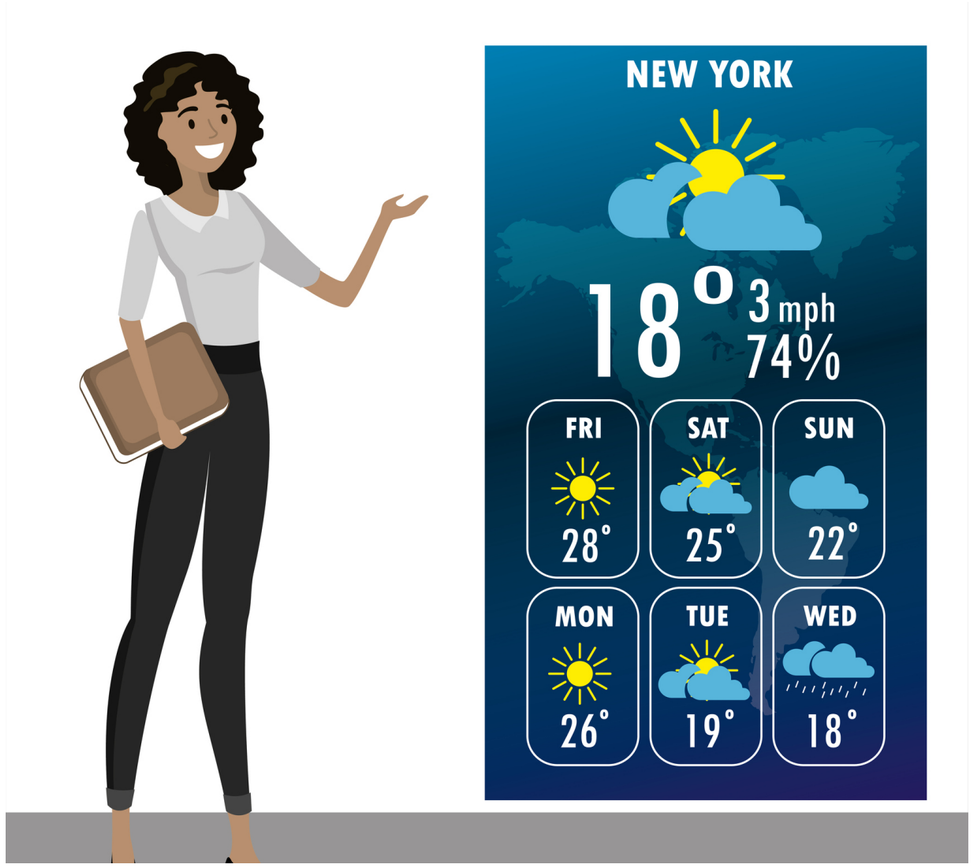
Going from unstructured to consistent structured output

Use cases for function calling

Calling multiple functions to provide complex responses

Use cases for function calling

Calling external APIs



Let's practice!

DEVELOPING AI SYSTEMS WITH THE OPENAI API

Extracting structured data from text

DEVELOPING AI SYSTEMS WITH THE OPENAI API

Francesca Donadoni
Curriculum Manager, DataCamp

datacamp

Implementing function calling

"We are currently seeking a highly skilled Data Scientist to join our innovative team at the company's headquarters in San Francisco, CA. In this role, you will have the opportunity to work on complex data analysis and modeling projects that drive our strategic decisions. Requirements: Minimum 3 years of experience in data science with Python and AWS, Azure or GCP."

```
from openai import OpenAI

client = OpenAI(api_key="ENTER YOUR KEY
                        HERE")

response= client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=messages,
    tools=function_definition,
)
```

Setting up function calling

```
function_definition = [{
        'type': 'function',
        'function': {
          'name': 'extract_job_info',
          'description': 'Get the job information from the body of the input text',
          'parameters': {
            ...
            }
    }]
```

Setting up function calling

```
function_definition = [{
        'type': 'function',
        'function': {
          'name': 'extract_job_info',
          'description': 'Get the job information from the body of the input text',
          'parameters': {
              'job': {'type': 'string',
                      'description': 'Job title'},
              'location': {'type': 'string',
                           'description': 'Office location'},
              ...
           }
    }]
```

Response

```
response= client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=messages,
    tools=function_definition
)
print(response.choices[0].message.tool_calls[0].function.arguments)
```

```
{"job":"Data Scientist","location":"San Francisco, CA"}
```

Let's practice!

DEVELOPING AI SYSTEMS WITH THE OPENAI API

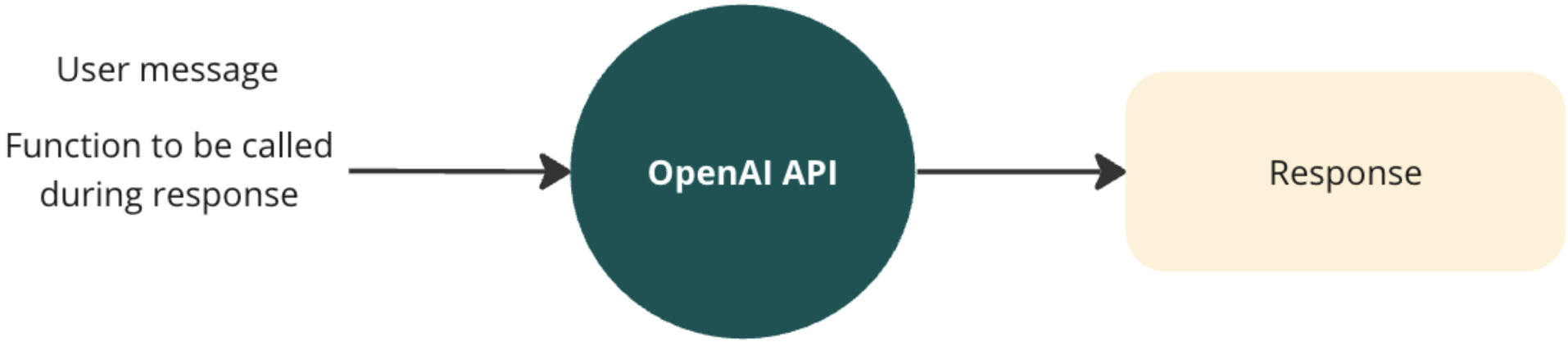
Working with multiple functions

DEVELOPING AI SYSTEMS WITH THE OPENAI API

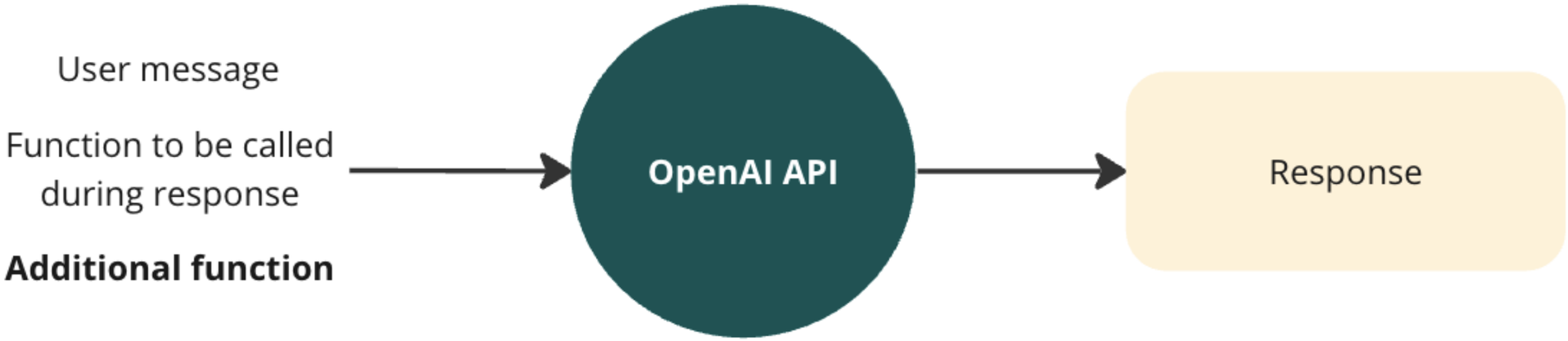
Francesca Donadoni
Curriculum Manager, DataCamp

datacamp

Parallel function calling



Parallel function calling



Example message

```
function_definition = [{'type': 'function','function':
        {'name': 'extract_job_info',
        'description': 'Get the job information from the body of the input text',
        'parameters': {
            'type': 'object',
            'properties': {
                'job': {'type': 'string', 'description': 'Job title'},
                'location': {'type': 'string', 'description': 'Location'}}
        }}
    }]
```

Example message

```
function_definition.append({'type': 'function',
        'function':{
            'name': 'get_timezone',
            'description': 'Return the timezone corresponding to the location in the
                            job advert',
            'parameters': {
            'type': 'object',
            'properties': {
                'timezone': {'type': 'string','description': 'Timezone'}}}
        }
    }
)
```

Example message

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo-1106",
    messages=messages,
    tools=function_definition
)
```

Example response

```
# Print the arguments of the first function
print(response.choices[0].message.tool_calls[0].function.arguments)
```

```
{"job": "Data Scientist", "location": "San Francisco, CA"}
```

```
# Print the arguments of the second function
print(response.choices[0].message.tool_calls[1].function.arguments)
```

```
{"timezone": "America/Los_Angeles"}
```

Setting specific functions

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo-1106",
    messages=messages,
    tools=function_definition,
    tool_choice='auto'
)
```

Setting specific functions

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo-1106",
    messages=messages,
    tools=function_definition,
    tool_choice={'type': 'function',
                 'function': {'name': 'extract_job_info'}
                }
)
```

Double checking the response

```
messages = []
messages.append({"role": "system", "content": "Don't make assumptions about what
values to plug into functions. Don't make up values to fill the response with."})
messages.append({"role": "system", "content": "Ask for clarification if needed."})
messages.append({"role": "user", "content": "What is the starting salary
                                             for the role?"})
```

Double checking the response

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo-1106",
    messages=messages,
    tools=function_definition
)
print(response.choices[0].message.content)
```

```
I don't have information about the starting salary for the role. If you'd like, I
can help you extract the job information from the description and then provide
additional assistance.
```

Let's practice!

DEVELOPING AI SYSTEMS WITH THE OPENAI API

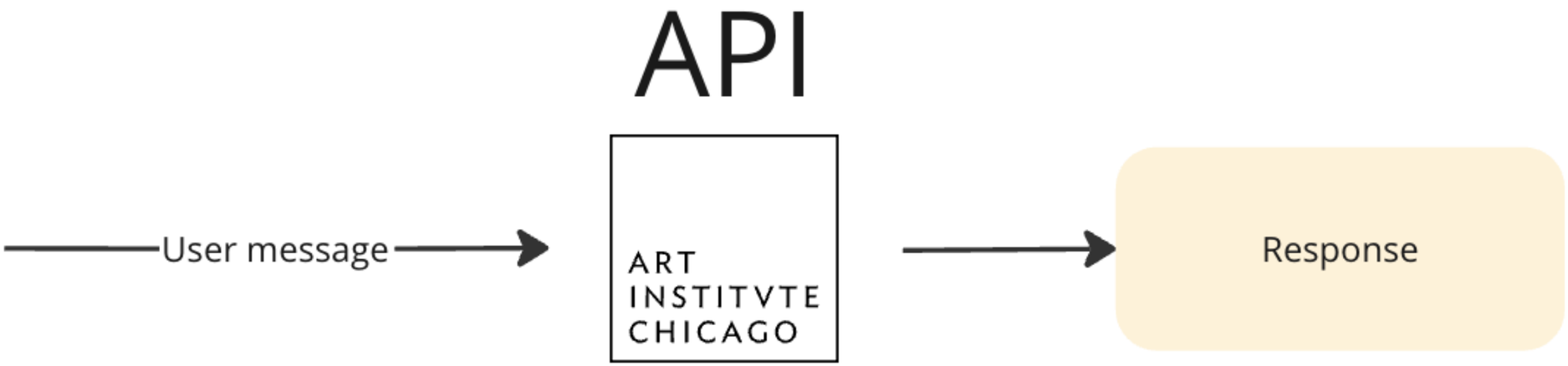
Calling external APIs

DEVELOPING AI SYSTEMS WITH THE OPENAI API

Francesca Donadoni
Curriculum Manager, DataCamp

datacamp

Python requests library for APIs



Python requests library for APIs

- Import the library

```
import requests
```

- Provide a URL

```
url = "https://api.artic.edu/api/v1/artworks/search"
```

- Provide query parameters

```
querystring = {"q":keyword}
```

- Get the response

```
response = requests.request("GET", url, params=querystring)
```

[1] https://api.artic.edu/docs/#quick-start

Packaging the function

```
import requests

def get_artwork(keyword):
    url = "https://api.artic.edu/api/v1/artworks/search"
    querystring = {"q":keyword}
    response = requests.request("GET", url, params=querystring)
    return response.text
```

Adding context

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[
{"role": "system",
 "content": "You are an AI assistant, a specialist in history
    of art. You should interpret the user prompt, and based on it extract one
    keyword for recommending artwork related to their preference."},
{"role": "user",
 "content": "I don't have much time to visit the museum and would
    like some recommendations. I like the seaside and quiet places."}
],
```

Adding the function to tools

```
function_definition=[{"type": "function",
        "function" : {
            "name": "get_artwork",
            "description": "This function calls the Art Institute of Chicago API
                            to find artwork that matches a keyword",
            "parameters": {
                "type": "object",
                "properties": {
                    "artwork keyword": {
                        "type": "string",
                        "description": "The keyword to be passed to the
                                        get_artwork function."}}},
            "result": {"type": "string"}
        }} ] )
```

Bringing it all together

```
import json
if response.choices[0].finish_reason=='tool_calls':
    function_call = response.choices[0].message.tool_calls[0].function
    if function_call.name == "get_artwork":
        artwork_keyword = json.loads(function_call.arguments)["artwork keyword"]
        artwork = get_artwork(artwork_keyword)
        if artwork:
            print(f"Here are some recommendations:
                    {[i['title'] for i in json.loads(artwork)['data']]}")
        else:
            print("Apologies, I couldn't make any recommendations based on the request.")
    else:
        print("Apologies, I couldn't find any artwork.")
else:
    print("I am sorry, but I could not understand your request.")
```

Final response

```
Here are some recommendations: ['Seaside, Port of Honfleur',
'Little Landscape at the Seaside (Kleine Landschaft am Meer)',
'Museum of Contemporary Art, Niterói, Rio de Janeiro, Brazil, Four Sketches',
'A seaside outing', 'Stacks of Wheat (End of Day, Autumn)', 'Stack of Wheat',
'Stacks of Wheat (Sunset, Snow Effect)', 'Stacks of Wheat (End of Summer)',
'Stack of Wheat (Thaw, Sunset)', 'Waitress at a Seaside Teahouse']
```

[1] Seaside, Port of Honfleur | The Art Institute of Chicago

Let's practice!

DEVELOPING AI SYSTEMS WITH THE OPENAI API