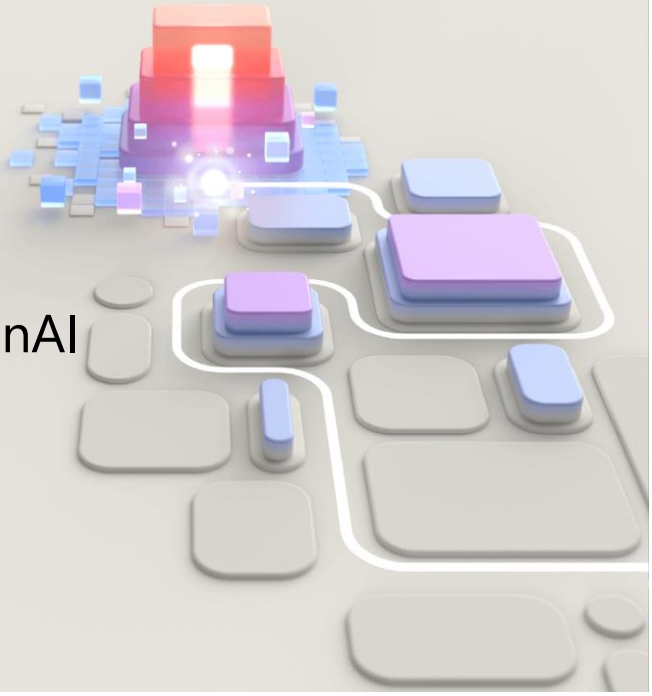
Microsoft

Develop Generative AI Solutions with Azure OpenAI Service [AI-050]



 Date

1

About this course

What we'll cover

In this course, you will learn how to implement generative AI solutions by using Microsoft Azure OpenAI Service. Specifically, you'll learn how to:

- Provision Azure OpenAI resources and deploy OpenAI models for generative AI tasks
- Use APIs and SDKs to consume models from client applications
- Apply prompt engineering to generate relevant results, generate code and images, and use your own data

Intended audience

The primary audience for this course is application developers seeking to include Azure OpenAI models in their applications.

Although Azure OpenAI models operate on natural language prompts, to use the APIs and SDKs effectively, a basic knowledge of Microsoft C# or Python is recommended.

2

Course schedule

	Day 1
Early morning	Introduction Get started with Azure OpenAI Service
Break	
Late morning	Build natural language solutions Apply prompt engineering
Lunch	
Early afternoon	Generate code Generate images
Break	
Late afternoon	Use your own data Q & A

Trainers:

Find the proposed schedule in the Trainer Prep Guide (PDF) and customized if needed. Delete this note before publishing.

3

Get the most out of your Microsoft Learn profile

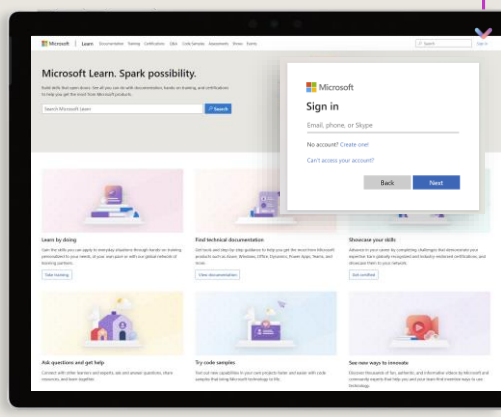
Verify, track, and share your training and certification progress and accomplishments—*all on one platform*

- Claim your achievement code for this course and share you have completed it.
- Access your course material and track progress on your learning activities.
- Share and verify your Microsoft Certifications via email, on social networking platforms, and on your résumé.
- Download and print transcripts and certificates.
- Manage your upcoming activities and certification exam appointments.

www.aka.ms/MyMicrosoftLearnProfile

Create your Microsoft Learn profile at learn.microsoft.com

- Select *Sign in* at the top, right corner of any Microsoft Learn page.
- Follow the Microsoft account authentication process.
- If the account that you have chosen to sign-in with doesn't already have a Microsoft Learn profile, you'll be guided to create one.



4

Access your course material

All course content is available on Microsoft Learn

learn.microsoft.com/training/courses/browse

- We'll go through this content together and as the course progresses, I will advise you on which modules to review.
- You can provide feedback for modules on Microsoft Learn. Find how at the bottom of each page.

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue.

5

This course includes labs:

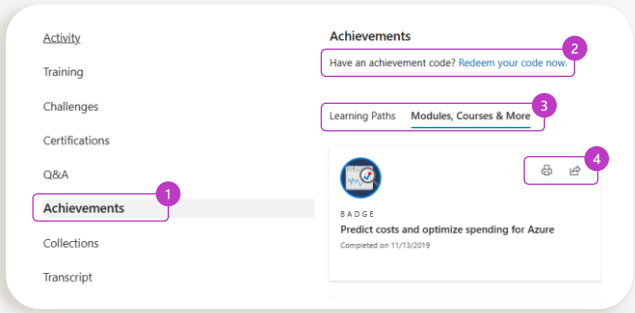
- Detailed lab instructions are included in your lab environment.
- Each exercise is standalone and requires:
  - A Microsoft Azure subscription
  - Approved access to the Azure OpenAI service

Access to the Azure OpenAI service can be requested through the form at aka.ms/oaiapply

Celebrate your accomplishments and feel empowered

Get your achievement recognized and earn your badge for completing this course

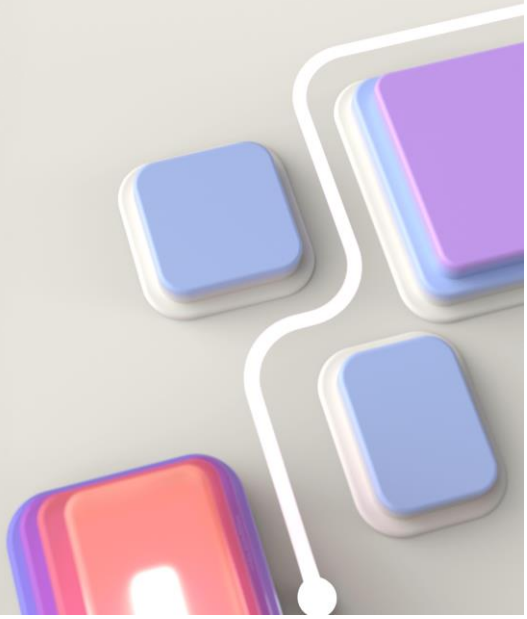
aka.ms/MyMicrosoftLearnProfile



1. Go to **Achievements** on your Microsoft Learn profile
2. Redeem the code provided by your trainer
3. Find your new badge **on Modules, Courses & More**
4. Share your new achievement with your professional network. You can also download and print your certificate.

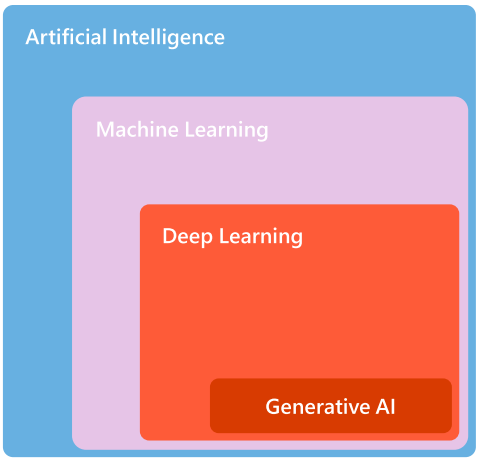
6

Get started with Azure OpenAI Service



7

What is generative AI?



1950s

Artificial Intelligence
the field of computer science that seeks to create intelligent machines that can replicate or exceed human intelligence

1990s

Machine Learning
subset of AI that enables machines to learn from existing data and improve upon that data to make decisions or predictions

2010s

Deep Learning
a machine learning technique in which layers of neural networks are used to process data and make decisions

2021

Generative AI
Create new written, visual, and auditory content given prompts or existing data.

8

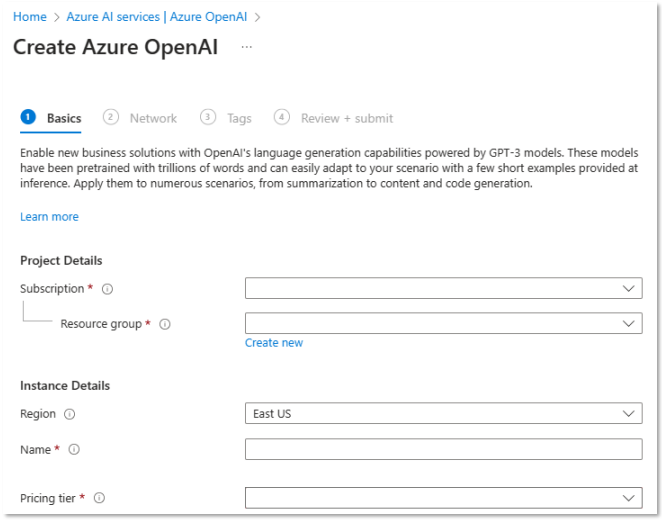
Provision an Azure OpenAI resource in Azure

Deploy a model in Azure OpenAI Studio to use it

1. Apply for access to the Azure OpenAI service: https://aka.ms/oaiapply
2. Create an **Azure OpenAI** resource in the Azure portal

Alternatively, use the Azure CLI

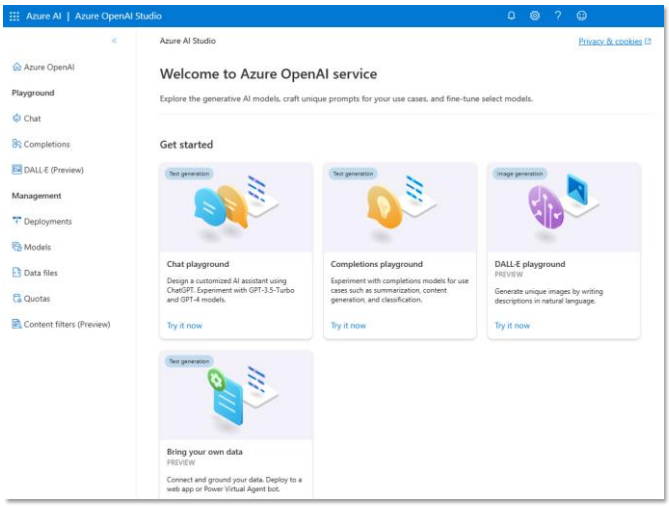
```
az cognitiveservices account create \
-n MyOpenAIResource \
-g MyResourceGroup \
-l eastus \
--kind OpenAI \
--sku s0 \
--subscription subscriptionID
```



9

Azure OpenAI Studio

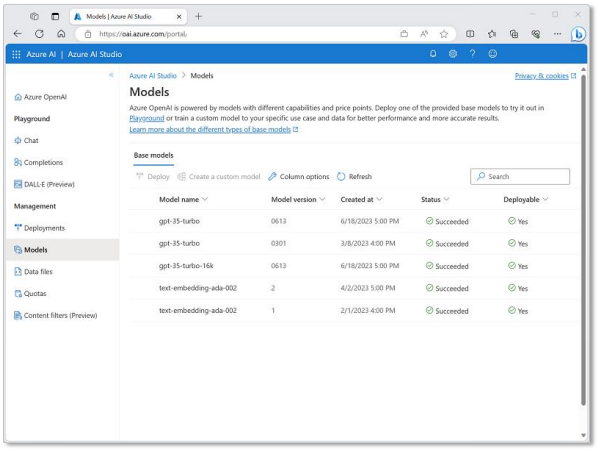
- Web portal for working with Azure OpenAI models: https://oai.azure.com/
- View and deploy base models
- Connect your own data source
- Manage fine tuning and data files for custom models
- Test models in visual playgrounds:
  - **Completions** (GPT-3 and earlier models)
  - **Chat** (GPT-3.5-Turbo and later models)



10

Types of generative AI model

Model Family	Description
GPT-4	Newest, most capable chat-based models for language and code generation (*restricted*)
GPT-3.5	Natural language and code-generation models
Embeddings	Models that use embeddings for specific tasks (similarity, text search, and code search)
DALL-E	Image-generation model (*restricted preview*)



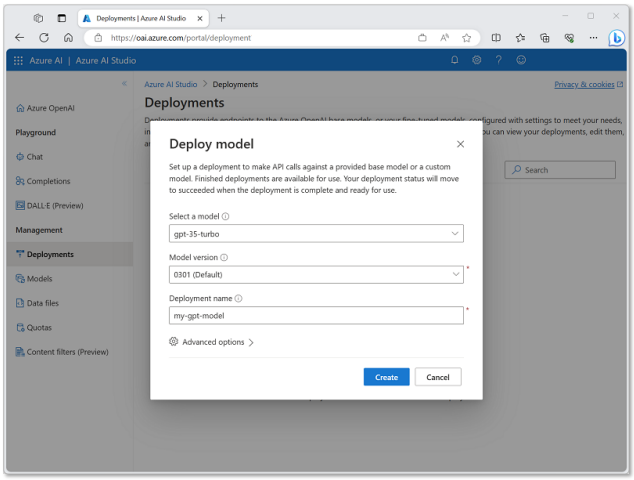
11

Deploying generative AI models

Deploy a model in Azure OpenAI Studio to use it

- You can deploy one or more instances of each available model
- The number of deployments depends on your quota, which you can see in the portal
- Alternatively, use the Azure CLI

```
az cognitiveservices account deployment create \
   -g myResourceGroupName \
   -n MyOpenAIResource \
   --deployment-name my-gpt-model \
   --model-name gpt-35-turbo \
   --model-version "0301"  \
   --model-format OpenAI \
   --scale-settings-scale-type "Standard"
```



12

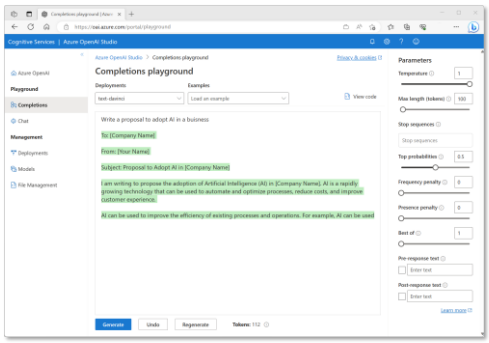
Using prompts to get completions from models

Task	Prompt	Completion
Classifying content	Tweet: I enjoyed the training course. Sentiment:	Positive
Generating new content	Write a poem about databases	Databases, oh databases, You keep our information safe, From the small to the large, You store our data in a place.
Transformation/Translation	English: Hello French:	Bonjour
Summarization	Scotland is [*long description of Scotland…*] Summarize the previous text	Scotland is [*summarized description…*]
Continuation	One way to grow tomatoes is to	start with seeds…
Question answering	How many moons does Earth have?	Earth has one moon.
Chat	*Setup, followed by messages…*	*A sequence of relevant responses*

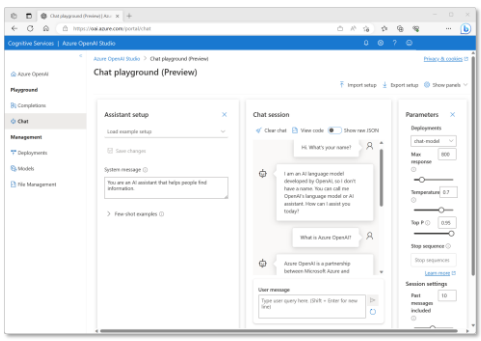
13

Testing models in Azure OpenAI Studio playgrounds

Completions playground



Chat playground



Azure OpenAI Studio includes multiple playgrounds for specific types of model. For natural language generation models, the available playgrounds are:

- The Completions playground – used for content generation tasks with GTP-3 family models
- The Chat playground – used for chat interactions with GPT-35-Turbo and later models.

14

Exercise: Get started with Azure OpenAI Service

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-get-started-azure-openai

15

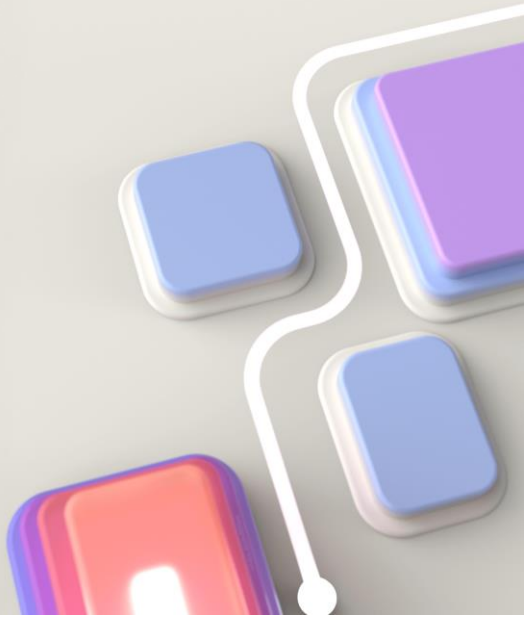
Resources

Get started with Azure OpenAI Service

https://aka.ms/mslearn-start-azure-openai

16

Build natural language solutions with Azure OpenAI Service



17

Integrating Azure OpenAI into your app

Applications submit prompts to models. Responses are completions.
Three REST API endpoints:

- **Completion** - model takes an input prompt, and generates one or more predicted completions
- **Embeddings** - model takes input and returns a vector representation of that input
- **ChatCompletion** - model takes input in the form of a chat conversation (where roles are specified with the message they send), and the next chat completion is generated

ChatCompletion will be the endpoint we focus on for this course

Use **Completion** and **Embeddings** with GPT-3 based models
Use **ChatCompletion** with GPT-35-Turbo and later models

18

Using the Azure OpenAI REST API

Completion Endpoint

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**completions**

Embedding Endpoint

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**embeddings**

ChatCompletion Endpoint

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**chat**/**completions**

19

Using Azure OpenAI SDKs – Microsoft C#

```
using Azure.AI.OpenAI;

OpenAIClient client = new OpenAIClient(new Uri(yourEndpoint), new AzureKeyCredential(yourKey));

// Build completion options object
ChatCompletionsOptions chatCompletionsOptions = new ChatCompletionsOptions()
{
    Messages =
    {
        new ChatMessage(ChatRole.System, "You are a helpful assistant."),
        new ChatMessage(ChatRole.User, "What is Azure OpenAI?"),
    },
    MaxTokens = 120,
    Temperature = 0.7f,
    DeploymentName = oaiModelName
};
// Send request to Azure OpenAI model
ChatCompletions response = client.GetChatCompletions(chatCompletionsOptions);
string completion = response.Choices[0].Message.Content;
Console.WriteLine($"Answer: {completion}");
```


20

Using Azure OpenAI SDKs – Python

```
from openai import AzureOpenAI

# Initialize the Azure OpenAI client
client = AzureOpenAI(
        azure_endpoint = your_endpoint,
        api_key=your_key,
        api_version="2023-05-15")

# Send request to Azure OpenAI model
response = client.chat.completions.create(
   model=azure_oai_model,
   temperature=0.7,
   max_tokens=120,
   messages=[
       {"role": "system", "content": "You are a helpful assistant."},
       {"role": "user", "content": "Summarize the following text in 20 words or less:\n" + text}
   ]
)

print("Summary: " + response.choices[0].message.content + "\n")
```


21

Exercise: Integrate Azure OpenAI into your app

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-build-language-azure-openai

22

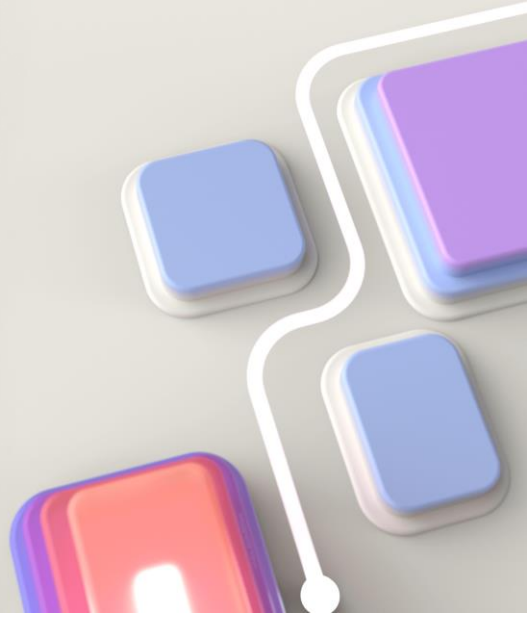
Resources

Build natural language solution with Azure OpenAI Service

https://aka.ms/mslearn-build-openai

23

Apply prompt engineering with Azure OpenAI Service

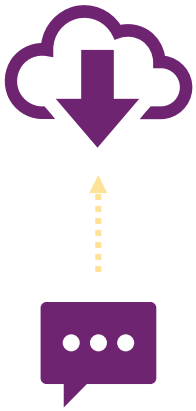


24

What is Prompt Engineering?

Constructing prompts to:

- Maximize relevancy and accuracy of completions
- Specify formatting and style of completions
- Provide conversational context
- Mitigate bias and improve fairness



25

Considerations for API endpoints

Completions API (*GPT-35-Turbo and earlier*)

Include context in the prompt (as a *meta prompt*)

```
{
  prompt: "You are a professional events
           planner. Write an invitation
           for a party to celebrate the
           launch of a new product."
}
```

ChatCompletions API (*GPT-35-Turbo and later*)

Set context in a system message

```
{
  "messages":[
    {"role": "system", "content": "You are a
        professional events planner."},
    {"role": "user", "content": "Write an
        invitation for a party to celebrate
        the launch of a new product."}
  ]
}
```


26

Providing clear instructions

Write a product description for a new water bottle

Introducing the latest addition to our product line - the innovative and eco-friendly water bottle. Made from high-quality, BPA-free materials, this bottle is designed to keep your beverages at the perfect temperature for hours. Its sleek and stylish design makes it the perfect accessory for any outdoor activity or workout session. With its convenient carrying handle and leak-proof lid, you can take it with you wherever you go, without worrying about spills or leaks. Plus, its easy-to-clean design ensures that you can use it again and again, making it the perfect choice for anyone looking for a reliable and sustainable water bottle.

Write a product description for a new water bottle that is 100% recycled. Be sure to include that it comes in natural colors with no dyes, and each purchase removes 10 pounds of plastic from our oceans

Introducing our newest water bottle, the eco-friendly choice for the conscious consumer. Made from 100% recycled materials and available in natural colors with no dyes, this bottle is not only stylish but also sustainable. Each purchase of this bottle removes 10 pounds of plastic from our oceans, making it a responsible choice for those who care about the environment. Its durable and leak-proof design makes it perfect for outdoor activities, while its easy-to-carry handle makes it convenient to take on the go. Join us in our mission to reduce plastic waste and choose this eco-friendly water bottle for your daily hydration needs.

27

Primary, *supporting*, and *grounding* content

Section markers (--- or ###)

Primary content to be summarized, translated, etc.

Supporting content to provide clarity / specificity

Grounding content to define scope for questions

Thanks for your interest in our summer school program.

We're looking for your ideas for webinars on topics such as AI, web development, *... <more...>*

Topics I'm very interested in: AI, webinar dates, submission deadlines

Extract the key points from the above email, and put them in a bulleted list

---

Artificial Intelligence (AI) has evolved over many years *... <more...>*

Where and when did the field of AI start?

28

Cues

- Cues provide a starting point on which the completion builds
- Use cues to set the scope and format for the output
- Cues are particularly useful when generating code. For example:
  - Provide the first keyword in a SQL SELECT statement
  - Define a name and parameters for a Python function

```
---
Review: The writing was good but the movie was too long.
Review: The special effects were terrible.
<more>...
---

Summarize the reviews above.
Most common complaints:
-
```

- The movie was too long.
- The special effects were terrible.

29

Requesting output composition

```
Write a table in markdown with 6 animals in it, with their genus and species
```

```
| Animal           | Genus          | Species        |
| ---------------- |:--------------:|:--------------:|
| Red Panda        | Ailurus        | fulgens        |
| African Elephant | Loxodonta      | africana       |
| Snow Leopard     | Panthera       | uncia          |
| Arctic Fox       | Vulpes         | lagopus        |
| Chimpanzee       | Pan            | troglodytes    |
| Siberian Tiger   | Panthera       | tigris altaica |
```


30

Using a system message

When using the ChatCompletions API, a system message sets context that can affect completions

```
{"role": "system", "content": "You are a casual, helpful assistant.
                               You will talk like an American old western film character."},
{"role": "user", "content": "Can you direct me to the library?"}
```

```
{"role": "assistant", "content": "Well howdy there, stranger! The library, huh?
                                  Y'all just head down the main road till you hit the town
                                  square. Once you're there, take a left and follow the street
                                  for a spell. You'll see the library on your right, can’t
                                  miss it. Happy trails!"}
```


31

Conversation history and *few shot* learning

- Including previous messages helps retain the context and style of a conversation
- Starting a conversation with user-defined messages and responses (*few shot* learning) sets examples for the expected responses

```
{"role": "system", "content": "You are an assistant that evaluates the sentiment of customer
feedback."},
{"role": "user", "content": "That was an awesome experience"},
{"role": "assistant", "content": "positive"},
{"role": "user", "content": "I won't do that again"},
{"role": "assistant", "content": "negative"},
{"role": "user", "content": "That was not worth my time"},
{"role": "assistant", "content": "negative"},
{"role": "user", "content": "You can't miss this"}
```

```
{"role": "assistant", "content": "positive"},
```


32

Chain of thought

Ask a model to break down its response and explain its reasoning

What sport is easiest to learn but hardest to master? Give a step by step approach of your thoughts, ending in your answer

Step 1: Identify the criteria for "easy to learn" and "hard to master" sports.
For a sport to be considered easy to learn, it should have simple rules and require minimal equipment For a sport to be considered hard to master, it should require years of practice to perfect and have a large variety of techniques and strategies...

Step 2: Consider different sports that fit these criteria.
Some sports that could be considered easy to learn but hard to master include:
- Tennis: Simple rules, easy to pick up, but requires years of practice to master all the different shots and strategies.
- Golf: Basic swing mechanics are easy to learn but perfecting them takes a lifetime of practice.
- ...

Step 3: Evaluate each sport based on the criteria.
Tennis and golf both require expensive equipment, which could make them less accessible to beginners. In terms of skill ceiling and time required to master, tennis and golf both have a high skill ceiling and require years of practice to perfect....

Step 4: Make a decision based on the evaluation.
Based on the above criteria and evaluation, I would say that tennis is the sport that is easiest to learn but hardest to master. While it may require expensive equipment, it is still accessible to many people. The basic rules and techniques are easy to learn, but mastering all the different shots and strategies takes years of practice...

33

Exercise: Utilize prompt engineering in your app

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/openai-prompt-engineering-lab

34

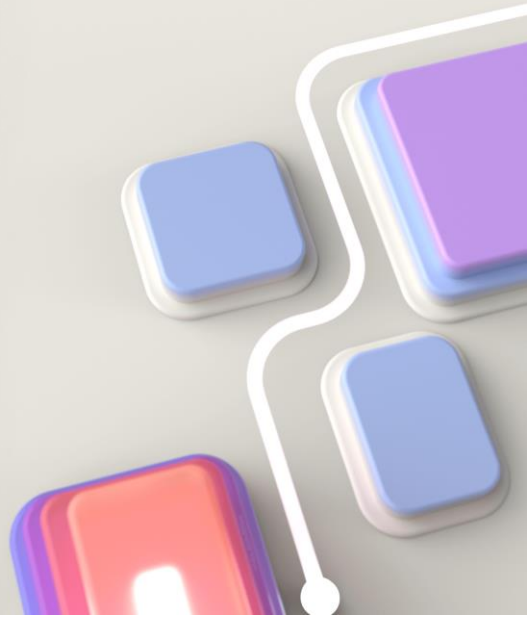
Resources

Apply prompt engineering with Azure OpenAI Service

https://aka.ms/mslearn-prompt-engineering-openai

35

Generate code with Azure OpenAI Service



36

Use natural language prompts to generate code

- Use prompt engineering to describe what you want clearly
- Break down complex tasks into smaller pieces

Prompt:

```
write a function for binary search in python
```

```
def binary_search(array, target):
    low = 0
    high = len(array) - 1

    while low <= high:
        mid = (low + high) // 2
        if array[mid] == target:
            return mid
        elif array[mid] < target:
            low = mid + 1
        else:
            high = mid - 1

    return -1
```


37

Change programming language and understand unknown code

Change programming language

```
Convert the following code to C#:
def print_squares(n):
    for i in range(1, n+1):
        print(i**2)
```

```
public void PrintSquares(int n)
{
    for (int i = 1; i <= n; i++)
    {
        Console.WriteLine(i * i);
    }
}
```

Understand unknown code

```
Could you explain what this code is doing?:
fn qzplv(n: i32, m: i32) -> i32 {
    if n == 0 {
        return m + 1;
    } else if m == 0 {
        return qzplv(n - 1, 1);
    } else {
        return qzplv(n - 1, qzplv(n, m - 1));
    }
}
```

```
Sure, I'd be happy to explain what this code is doing!

This code is implementing the Ackermann function,...
```


38

Complete partial code

- Writing your prompt with the most amount of information available will be the most accurate
- If enough code is given to determine the programming language, the model will use that language

Prompt:

```
complete the following function
"""
# calculate the average of the numbers
# in an array, but only if they're even
def
```

```
# calculate the average of the even
# numbers in an array
def calculate_even_average(numbers):
    total = 0
    count = 0
    for number in numbers:
        if number % 2 == 0:
            total += number
            count += 1
    if count == 0:
        return 0  # avoid division by zero
    else:
        return total / count
```


39

Write unit tests

Prompt:

```
write three unit tests for this function
"""
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```

```
def test_binary_search():
    # Test case 1: Target element is in the middle of the
    # array
    arr1 = [1, 2, 3, 4, 5, 6]
    target1 = 4
    assert binary_search(arr1, target1) == 3

    # Test case 2: Target element is at the beginning of
    # the array
    arr2 = [1, 2, 3, 4, 5, 6]
    target2 = 1
    assert binary_search(arr2, target2) == 0

    # Test case 3: Target element is not in the array
    arr3 = [1, 2, 3, 4, 5, 6]
    target3 = 7
    assert binary_search(arr3, target3) == -1
```


40

Fix bugs in your code

- Help identify and fix bugs quickly
- Provide an explanation of the issues

Prompt:

```
This function isn't working for me. Fix the bugs in this function
"""
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        number = numbers[i]
        total += number
    average = total
    rerun averave
```

```
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        number = numbers[i]
        total += number
    average = total / len(numbers)
    return average

There were a few issues with the original code:

1. The variable "average" was assigned the value of "total" instead of the actual average.
2. There was a typo in the last line - "rerun" should be "return".
3. The average was not actually calculated - the corrected code divides the total by the length of
the "numbers" list to get the average.
```


41

Improve performance

- Identify where improvements can be made
- Models able to optimize and refactor

Prompt:

```
Can this function be more efficient?
"""
def sum_of_n(n):
    result = 0
    for i in range(1, n+1):
        result += i
    return result
```

```
We can improve this function it by using a mathematical formula instead.

Here's the more efficient version:

def sum_of_n(n):
    return n * (n + 1) // 2
```


42

Exercise: Generate and improve code with Azure OpenAI Service

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-openai-code-gen

43

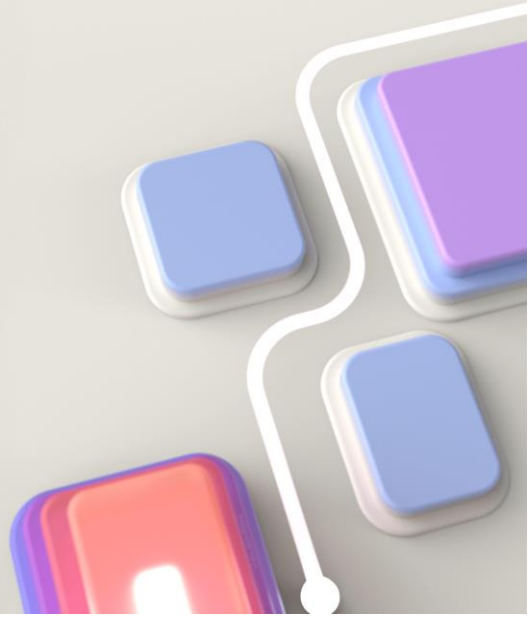
Resources

Generate code with Azure OpenAI Service

https://aka.ms/mslearn-code-openai

44

Generate images with Azure OpenAI Service



45

What is DALL-E?

Generate images with a description

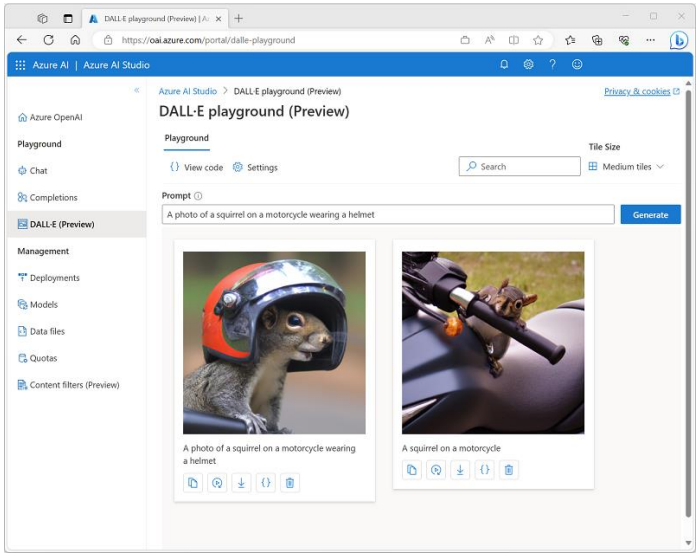
- Neural network based model for generating images
- Use natural language to describe what the image should be
- Specify content and style

Prompt:

```
a highland cow in a field on the coast of
Scotland, digital art style
```

46

Using DALL-E in Azure OpenAI Studio



47

Using the Azure OpenAI REST API

DALL-E Endpoint

https://*endpoint*.openai.azure.com/openai/images/generations:submit?api-version=*api_version*

48

Using Azure OpenAI SDKs for DALL-E – Microsoft C#

```
using Azure.AI.OpenAI;

OpenAIClient client = new OpenAIClient(new Uri(yourEndpoint), new AzureKeyCredential(yourKey));

ImageGenerations response = client.GetImageGenerations(
    new ImageGenerationOptions()
    {
          Prompt = "A badger wearing a tuxedo",
          Size = ImageSize. Size1024x1024,
    });

Uri imageUri = response.Data[0].Url;
Console.WriteLine("Image URL: " + imageUri + "\n");
```


49

Using Azure OpenAI SDKs – Python

```
import openai

openai.api_type = "azure"
openai.api_base = your_endpoint
openai.api_version = "2023-06-01-preview “ # be sure to use a version that supports dall-e
openai.api_key = your_key

response = openai.Image.create(
    prompt: ‘A badger wearing a tuxedo’,
    size: ‘1024x1024’,
    n: 1
}

print("...Image generation complete...\n")
image_url = response["data"][0]["url"]
print("Image URL: " + image_url + "\n")
```


50

Exercise - Generate images with a DALL-E model

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-openai-dall-e

51

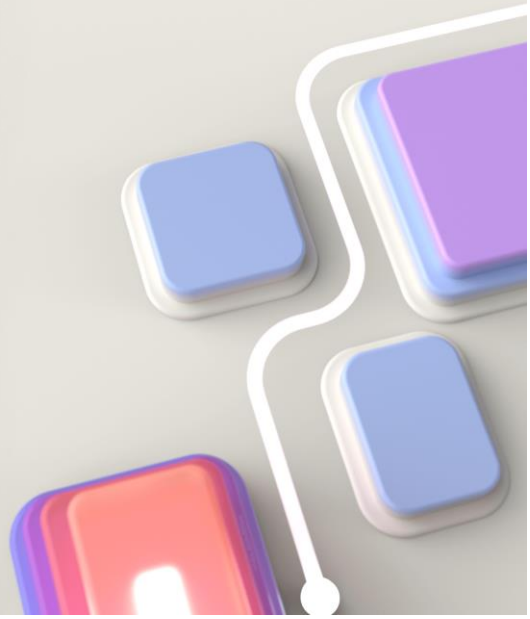
Resources

Build natural language solution with Azure OpenAI Service

https://aka.ms/mslearn-image-gen-module

52

Use your own data with Azure OpenAI Service



53

How Azure OpenAI can use your data

Set up your data source

- Use an existing data source, such as an Azure search resource
- Use the Azure OpenAI studio to create that data source, if you don't already have one
- When creating the data source, you can use data already in your account such as blob storage

Configure the studio or your app to connect to that data source

- In the studio, set up the connection by pointing it to the data source
- In your app, specify the data source in the prompt parameters

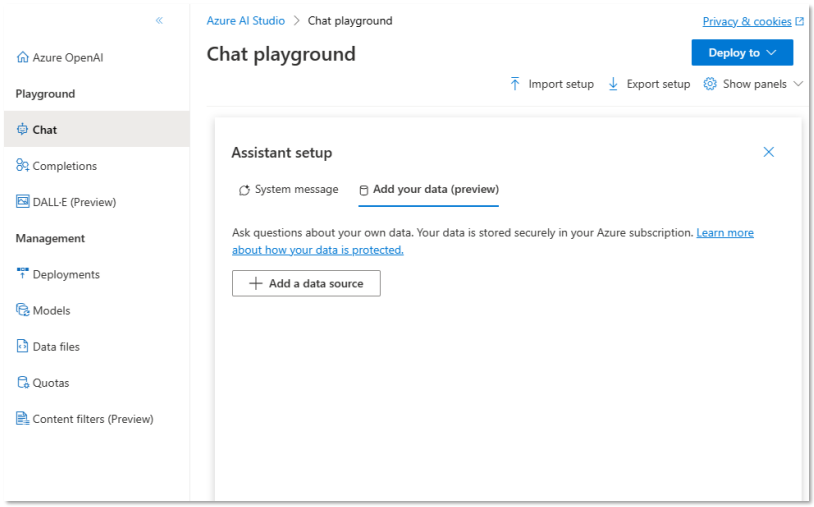
Use the Azure OpenAI model, which now uses your data for grounding

- Chat with the AI models like normal
- If the data source has relevant information about the prompt, it will use that data
- You can specify if the AI model is limited to just your data source

54

Connect to your data source

- Add your data source in the Chat playground, under Assistant setup
- Use an existing data source, or use that wizard to create a new one
- Once connected, a new chat session will start. Chat like normal, and see how the AI model references that data



55

Using the Azure OpenAI REST API

Using your own data

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**extensions/chat/completions**?api-version=*version*

- With each call, you need to specify the data source values, along with the messages array and any other parameters
- The REST endpoint will be different than normal chats

```
{
    "dataSources": [
        {
            "type": "AzureCognitiveSearch",
            "parameters": {
                "endpoint": "<your_search_endpoint>",
                "key": "<your_search_endpoint>",
                "indexName": "<your_search_index>"
            }
        }
    ],
    "messages":[
                ...
    ]
}
```


56

Using Azure OpenAI SDKs – Microsoft C#

```
AzureCognitiveSearchChatExtensionConfiguration ownDataConfig = new()
{
        SearchEndpoint = new Uri(searchEndpoint),
        IndexName = searchIndex
};
ownDataConfig.SetSearchKey(searchKey);

ChatCompletionsOptions chatCompletionsOptions = new ChatCompletionsOptions()
{
    Messages =
    {
        new ChatMessage(ChatRole.User, “I want to go to New York. Where should I stay?”)
    },
    MaxTokens = 600,
    Temperature = 0.9f,
    DeploymentName = modelName,
    AzureExtensionsOptions = new AzureChatExtensionsOptions()
    {
        Extensions = {ownDataConfig}
    }
};
```


57

Using Azure OpenAI SDKs – Python

```
# Create extension config for own data
extension_config = dict(dataSources = [
  {
    "type": "AzureCognitiveSearch",
    "parameters": {
      "endpoint":search_endpoint,
      "key": search_key,
      "indexName": search_index,
    }
  }])
response = client.chat.completions.create(
  model = azure_oai_model,
  temperature = 0.5,
  max_tokens = 1000,
  messages = [
    {"role": "user", "content": "I want to go to New York. Where should I stay?"}}
  ],
  extra_body = extension_config
)
```


58

Exercise: Use your own data with Azure OpenAI Service

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-openai-own-data

59

Resources

Use your own data with Azure OpenAI Service

https://aka.ms/mslearn-openai-own-data-module

60

Microsoft

Thank You

61