Before You Start

Tip: As you follow the instructions in this pane, whenever you see a icon, you can use it to copy text from the instruction pane into the virtual machine interface. This is particularly useful to copy code; but bear in mind you may need to modify the pasted code to fix indent levels or formatting before running it!

1. If prompted, log into the **Student** account with the password Pa55w.rd. If prompted to allow your PC to be discoverable, select **No**.
2. If you do not already have an Azure subscription, sign up for a free trial at https://azure.microsoft.com/free/.

Note: This hosted virtual machine environment supports redirection of local speaker and microphone hardware.

Get Started with Cognitive Services

In this exercise, you'll get started with Cognitive Services by creating a **Cognitive Services** resource in your Azure subscription and using it from a client application. The goal of the exercise is not to gain expertise in any particular service, but rather to become familiar with a general pattern for provisioning and working with cognitive services as a developer.

Clone the repository for this course

If you have not already cloned **AI-102-AIEngineer** code repository to the environment where you're working on this lab, follow these steps to do so. Otherwise, open the cloned folder in Visual Studio Code.

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/MicrosoftLearning/AI-102-AIEngineer` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the C# code projects in the repo.

Note: If you are prompted to add required assets to build and debug, select **Not Now**.

Provision a Cognitive Services resource

Azure Cognitive Services are cloud-based services that encapsulate artificial intelligence capabilities you can incorporate into your applications. You can provision individual cognitive services resources for specific APIs (for example, **Language** or **Computer Vision**), or you can provision a general **Cognitive Services** resource that provides access to multiple cognitive services APIs through a single endpoint and key. In this case, you'll use a single **Cognitive Services** resource.

1. Open the Azure portal at `https://portal.azure.com` , and sign in using the Microsoft account associated with your Azure subscription.

2. Select the **+ Create a resource** button, search for *cognitive services*, and create a **Cognitive Services** resource with the following settings:
    - **Subscription**: *Your Azure subscription*
    - **Resource group**: *Choose or create a resource group (if you are using a restricted subscription, you may not have permission to create a new resource group - use the one provided)*
    - **Region**: *Choose any available region*
    - **Name**: *Enter a unique name*
    - **Pricing tier**: Standard S0
3. Select the required checkboxes and create the resource.
4. Wait for deployment to complete, and then view the deployment details.
5. Go to the resource and view its **Keys and Endpoint** page. This page contains the information that you will need to connect to your resource and use it from applications you develop. Specifically:
    - An HTTP *endpoint* to which client applications can send requests.
    - Two *keys* that can be used for authentication (client applications can use either key to authenticate).
    - The *location* where the resource is hosted. This is required for requests to some (but not all) APIs.

Use a REST Interface

The cognitive services APIs are REST-based, so you can consume them by submitting JSON requests over HTTP. In this example, you'll explore a console application that uses the **Language** REST API to perform language detection; but the basic principle is the same for all of the APIs supported by the Cognitive Services resource.

> **Note**: In this exercise, you can choose to use the REST API from either **C#** or **Python**. In the steps below, perform the actions appropriate for your preferred language.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **01-getting-started** folder and expand the **C-Sharp** or **Python** folder depending on your language preference.
2. View the contents of the **rest-client** folder, and note that it contains a file for configuration settings:
    - **C#**: appsettings.json
    - **Python**: .env

    Open the configuration file and update the configuration values it contains to reflect the **endpoint** and an authentication **key** for your cognitive services resource. Save your changes.
3. Note that the **rest-client** folder contains a code file for the client application:
    - **C#**: Program.cs
    - **Python**: rest-client.py

    Open the code file and review the code it contains, noting the following details:
    - Various namespaces are imported to enable HTTP communication
    - Code in the **Main** function retrieves the endpoint and key for your cognitive services resource - these will be used to send REST requests to the Text Analytics service.
    - The program accepts user input, and uses the **GetLanguage** function to call the Text Analytics language detection REST API for your cognitive services endpoint to detect the language of the text that was entered.
    - The request sent to the API consists of a JSON object containing the input data - in this case, a collection of **document** objects, each of which has an **id** and **text**.

- The key for your service is included in the request header to authenticate your client application.
- The response from the service is a JSON object, which the client application can parse.

4. Right-click the **rest-client** folder and open an integrated terminal. Then enter the following language-specific command to run the program:

   C#

   ```
   dotnet run
   ```

   Python

   ```
   python rest-client.py
   ```

5. When prompted, enter some text and review the language that is detected by the service, which is returned in the JSON response. For example, try entering "Hello", "Bonjour", and "Hola".

6. When you have finished testing the application, enter "quit" to stop the program.

Use an SDK

You can write code that consumes cognitive services REST APIs directly, but there are software development kits (SDKs) for many popular programming languages, including Microsoft C#, Python, and Node.js. Using an SDK can greatly simplify development of applications that consume cognitive services.

1. In Visual Studio Code, in the **Explorer** pane, in the **01-getting-started** folder, expand the **C-Sharp** or **Python** folder depending on your language preference.

2. Right-click the **sdk-client** folder and open an integrated terminal. Then install the Text Analytics SDK package by running the appropriate command for your language preference:

   C#

   ```
   dotnet add package Azure.AI.TextAnalytics --version 5.1.0
   ```

   Python

   ```
   pip install azure-ai-textanalytics==5.1.0
   ```

3. View the contents of the **sdk-client** folder, and note that it contains a file for configuration settings:

   - **C#**: appsettings.json
   - **Python**: .env

   Open the configuration file and update the configuration values it contains to reflect the **endpoint** and an authentication **key** for your cognitive services resource. Save your changes.

4. Note that the **sdk-client** folder contains a code file for the client application:

   - **C#**: Program.cs
   - **Python**: sdk-client.py

   Open the code file and review the code it contains, noting the following details:

   - The namespace for the SDK you installed is imported

- Code in the **Main** function retrieves the endpoint and key for your cognitive services resource - these will be used with the SDK to create a client for the Text Analytics service.
- The **GetLanguage** function uses the SDK to create a client for the service, and then uses the client to detect the language of the text that was entered.

5. Return to the integrated terminal for the **sdk-client** folder, and enter the following command to run the program:

   C#

   ```
   dotnet run
   ```

   Python

   ```
   python sdk-client.py
   ```

6. When prompted, enter some text and review the language that is detected by the service. For example, try entering "Goodbye", "Au revoir", and "Hasta la vista".
7. When you have finished testing the application, enter "quit" to stop the program.

Note: Some languages that require Unicode character sets may not be recognized in this simple console application.

More information

For more information about Azure Cognitive Services, see the Cognitive Services documentation.

Return to Microsoft Learn

Now that you have completed the exercise, return to Microsoft Learn to complete the knowledge check and earn points for completing this module.