**LangFlow Assignment**

**Blog writer**

Build a Blog Writer flow for a one-shot application using OpenAI.

This flow extends the Basic Prompting flow with the **URL** and **Parse data** components that fetch content from multiple URLs and convert the loaded data into plain text.

OpenAI uses this loaded data to generate a blog post, as instructed by the **Text input** component.

**Prerequisites**[**​**](https://docs.langflow.org/tutorials-blog-writer#899268e6c12c49b59215373a38287507)

* [Langflow installed and running](https://docs.langflow.org/get-started-installation)
* [OpenAI API key created](https://platform.openai.com/)

**Create the blog writer flow**[**​**](https://docs.langflow.org/tutorials-blog-writer#0c1a9c65b7d640f693ec3aad963416ff)

1. From the Langflow dashboard, click **New Flow**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Go to All Templates and select **Blog Writer**.

A screenshot of a computer

AI-generated content may be incorrect.

1. The **Blog Writer** flow is created.

A screenshot of a chat

AI-generated content may be incorrect.

First: go to the OpenAI and use this OpenAI Key: **sk-proj-2xEPooTkDahEJeK0Cv51d3\_CRl0dm-HGc9v242z7-dAwWYHILFDDDYrvzTCVPIm40A\_G\_WvTMDT3BlbkFJKWlY7lyq3DWtUWUm1NwJ1iERBDFdR3NmaXt8os4XRK4bln6hcIKINSja9BUmNAHo\_0Qqz5ib8A**

This flow creates a one-shot article generator with **Prompt**, **OpenAI**, and **Chat Output** components, augmented with reference content and instructions from the **URL** and **Text Input** components.

The **URL** component extracts raw text and metadata from one or more web links. The **Parse Data** component converts the data coming from the **URL** component into plain text to feed the prompt.

To examine the flow's prompt, click the **Template** field of the **Prompt** component.

Blog:

The {instructions} value is received from the **Text input** component, and one or more {references} are received from a list of URLs parsed from the **URL** component.

**Run the blog writer flow**[**​**](https://docs.langflow.org/tutorials-blog-writer#b93be7a567f5400293693b31b8d0f81a)

1. Click the **Playground** button. Here you can chat with the AI that has access to the **URL** content.
2. Click the **Lighting Bolt** icon to run it.
3. To write about something different, change the values in the **URL** component and adjust the instructions on the left side bar of the **Playground**. Try again and see what the LLM constructs.

Part II:

**Document QA**

Build a question-and-answer chatbot with a document loaded from local memory.

**Prerequisites**[**​**](https://docs.langflow.org/tutorials-document-qa#6555c100a30e4a21954af25e2e05403a)

* [Langflow installed and running](https://docs.langflow.org/get-started-installation)
* [OpenAI API key created](https://platform.openai.com/)

**Create the document QA flow**[**​**](https://docs.langflow.org/tutorials-document-qa#204500104f024553aab2b633bb99f603)

1. From the Langflow dashboard, click **New Flow**.
2. Select **Document QA**.
3. The **Document QA** flow is created.

A screenshot of a computer

AI-generated content may be incorrect.

This flow is composed of a standard chatbot with the **Chat Input**, **Prompt**, **OpenAI**, and **Chat Output** components, but it also incorporates a **File** component, which loads a file from your local machine. **Parse Data** is used to convert the data from **File** into the **Prompt** component as {Document}. The **Prompt** component is instructed to answer questions based on the contents of {Document}. This gives the **OpenAI** component context it would not otherwise have access to.

**Run the document QA flow**[**​**](https://docs.langflow.org/tutorials-document-qa#f58fcc2b9e594156a829b1772b6a7191)

1. To select a document to load, in the **File** component, click the **Path** field. Select a local file, and then click **Open**. The file name appears in the field.
2. Click the **Playground** button. Here you can chat with the AI that has access to your document's content.
3. Type in a question about the document content and press Enter. You should see a contextual response.