

OpenAI API Quickstart Tutorial

Overview

This tutorial is to help you set up your local OpenAI API development environment and send your API requests.

Account setup

First, create an [OpenAI account](#).

Next, navigate to the [API key page](#) and "Create new secret key".

The image shows two screenshots from the OpenAI developer interface. The top screenshot is the 'API keys' management page. It features a sidebar with navigation links: Playground, Assistants, Fine-tuning, API keys (highlighted), Storage, Usage, and Settings. The main content area is titled 'API keys' and contains instructions about secret API keys, a warning not to share them, and a link to the 'Usage page' to enable tracking. Below this is a table with columns: NAME, SECRET KEY, TRACKING, CREATED, LAST USED, and PERMISSIONS. The table is currently empty, and a '+ Create new secret key' button is visible at the bottom. The bottom screenshot is the 'Create new secret key' dialog. It has a title 'Create new secret key' and a section for 'Name' with the label 'Optional'. A text input field contains the placeholder text 'Name it as you like'. Below this is a 'Permissions' section with three buttons: 'All' (selected), 'Restricted', and 'Read Only'. At the bottom right, there are two buttons: 'Cancel' and 'Create secret key'.

API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that we've found has leaked publicly.

Enable tracking to see usage per API key on the [Usage page](#).

NAME	SECRET KEY	TRACKING	CREATED	LAST USED	PERMISSIONS
------	------------	----------	---------	-----------	-------------

+ Create new secret key

Create new secret key

Name Optional

Name it as you like

Permissions

All Restricted Read Only

Cancel Create secret key

Save your key

Please save this secret key somewhere safe and accessible. For security reasons, you won't be able to view it again through your OpenAI account. If you lose this secret key, you'll need to generate a new one.

Copy

Make sure to save it to a safe place!

Permissions

Read and write API resources

Done

Make sure to save the key and do not share it with anyone.

Step 1: Setting up Python

You can refer to [this tutorial](#) if you want to learn how to set up Python.

Once you have Python installed, you need to install the OpenAI Python library. From the terminal / command line, run:

```
pip install --upgrade openai
```

Step 2: Set up your API key

MacOS (From OpenAI official tutorial)

- 1. Open Terminal:** You can find it in the Applications folder or search for it using Spotlight (Command + Space).
- 2. Edit Bash Profile:** Use the command `nano ~/.bash_profile` or `nano ~/.zshrc` (for newer MacOS versions) to open the profile file in a text editor.
- 3. Add Environment Variable:** In the editor, add the line below, replacing `your-api-key-here` with your actual API key:

```
export OPENAI_API_KEY='your-api-key-here'
```

4. **Save and Exit:** Press Ctrl+O to write the changes, followed by Ctrl+X to close the editor.

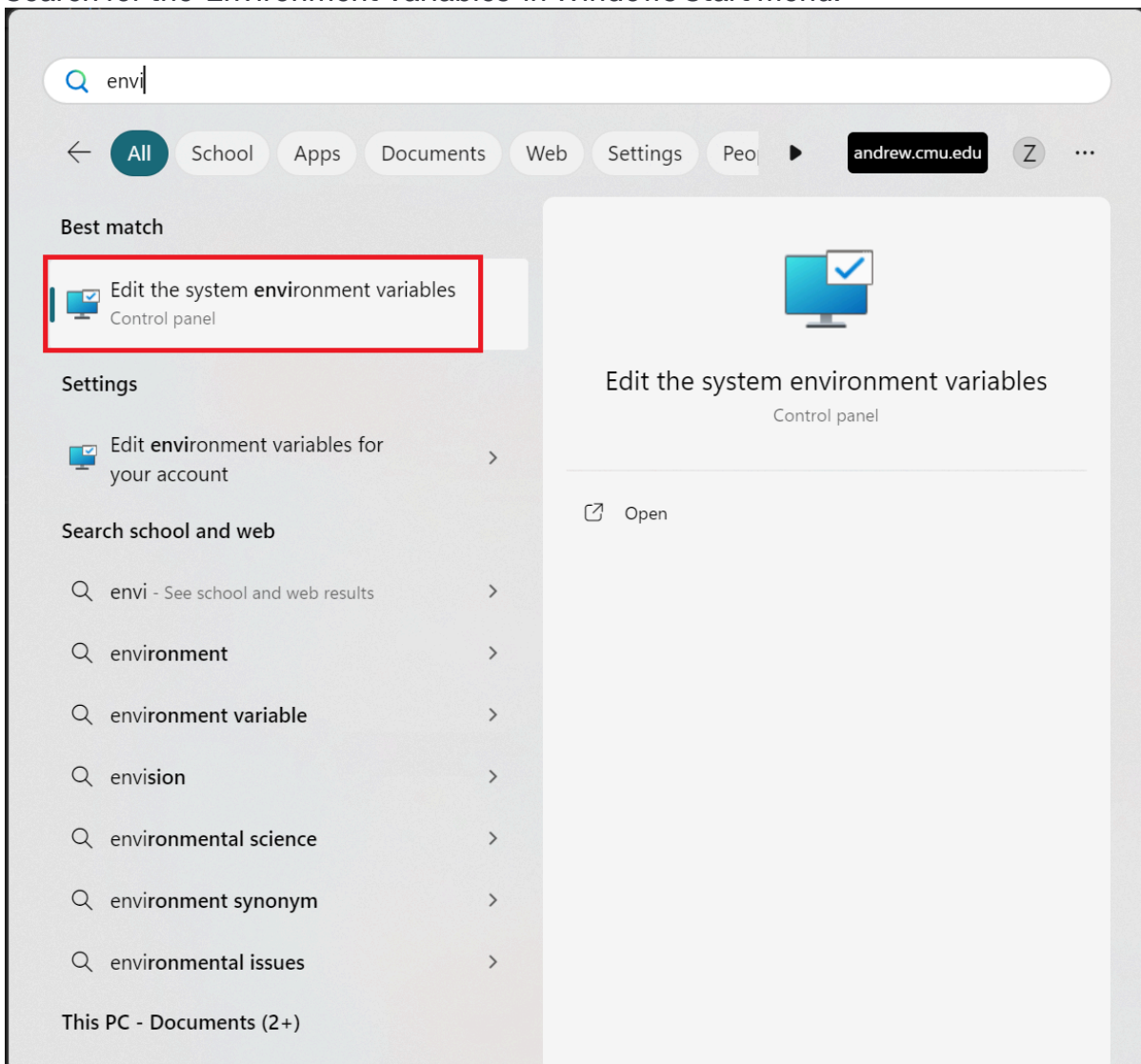
5. **Load Your Profile:** Use the command `source ~/.bash_profile` or `source ~/.zshrc` to load the updated profile.

6. **Verification:** Verify the setup by typing `echo $OPENAI_API_KEY` in the terminal. It should display your API key.

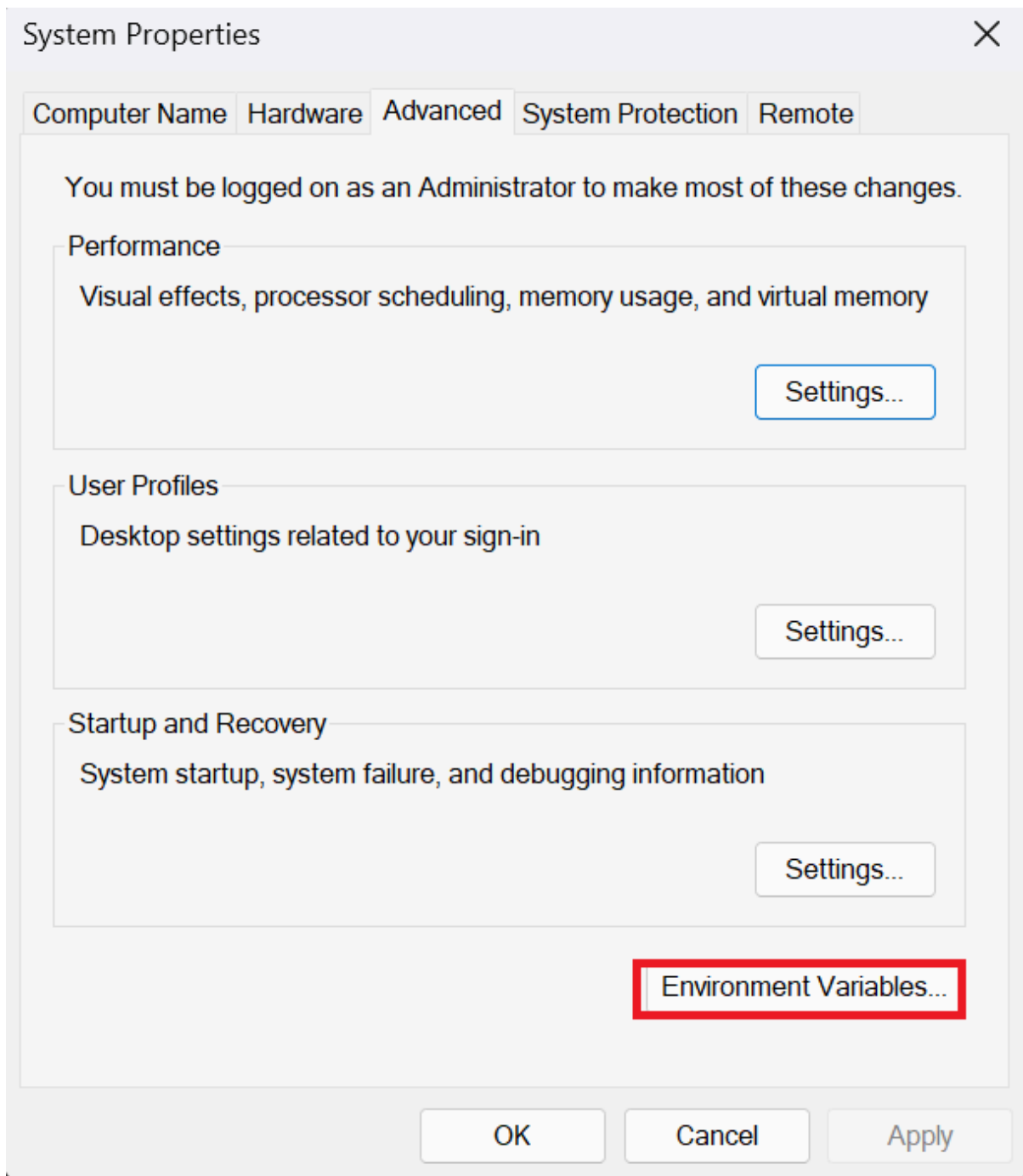
Windows

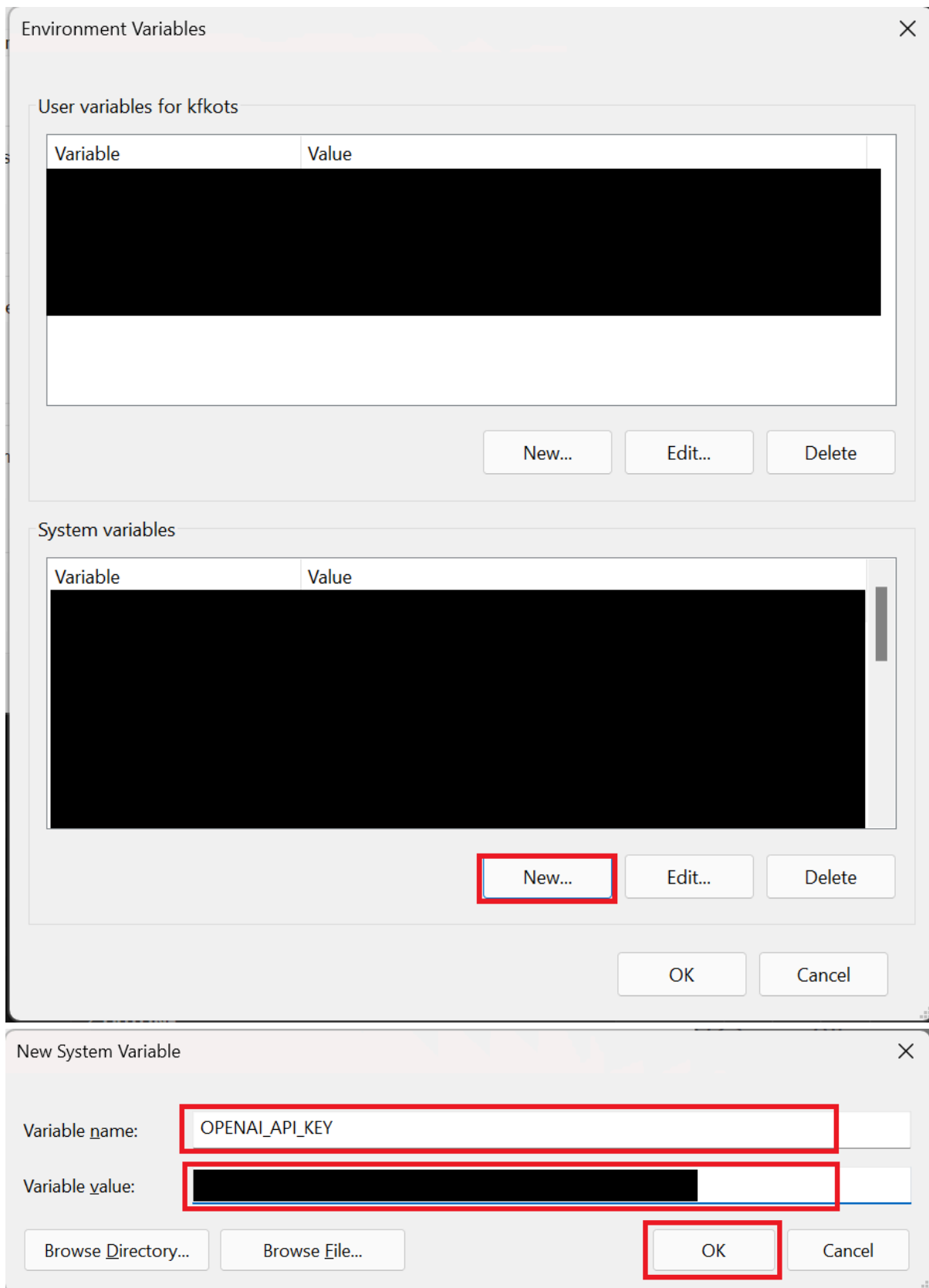
1. Permanently set up environment variable:

- Search for the 'Environment Variables' in Windows Start Menu.



- In the 'System variables' section, click 'New...' and enter OPENAI_API_KEY as the variable name and your API key as the variable value.





Step 3: Sending your first API request

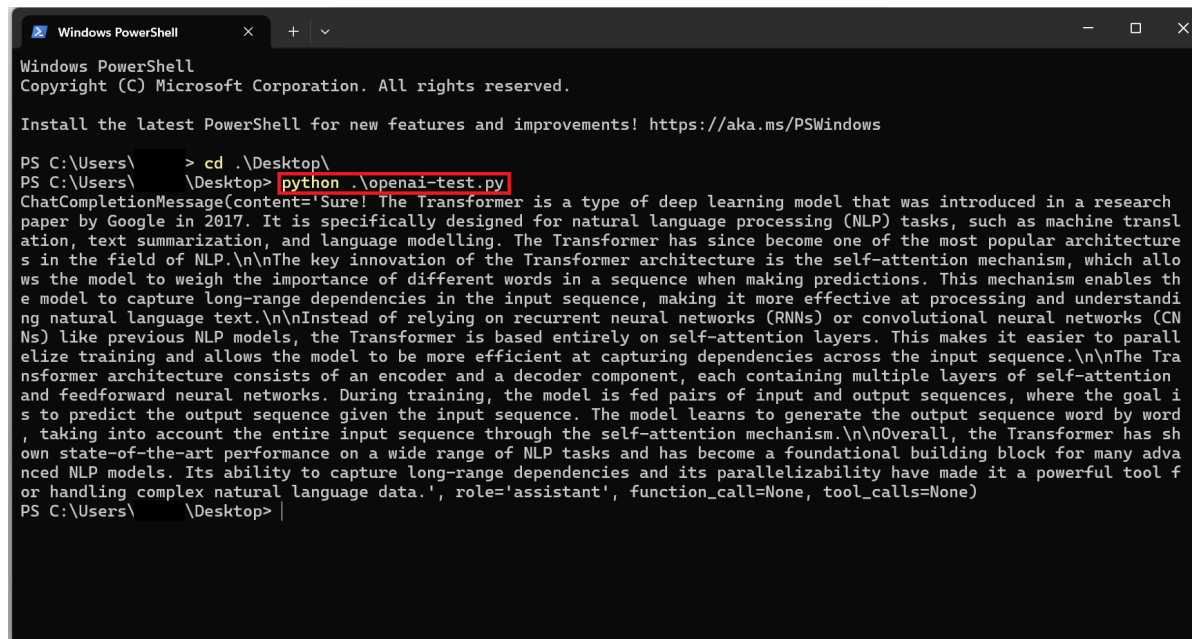
Making an API request

After you finished all that, you can finally send a request to the OpenAI API using the Python library. To do this, create a file named `openai-test.py` using the terminal or an IDE.

Inside the file, copy and paste the example below:

```
1 from openai import OpenAI
2 client = OpenAI()
3
4 completion = client.chat.completions.create(
5     model="gpt-3.5-turbo",
6     messages=[
7         {"role": "system", "content": "You are a machine learning specialist, skilled in expl
8         {"role": "user", "content": "Explain Transformer in machine learning to me."}
9     ]
10 )
11
12 print(completion.choices[0].message)
```

To run the code, enter `python openai-test.py` into the terminal / command line.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ > cd .\Desktop\
PS C:\Users\ \Desktop> python .\openai-test.py
ChatCompletionMessage(content='Sure! The Transformer is a type of deep learning model that was introduced in a research paper by Google in 2017. It is specifically designed for natural language processing (NLP) tasks, such as machine translation, text summarization, and language modelling. The Transformer has since become one of the most popular architectures in the field of NLP.\n\nThe key innovation of the Transformer architecture is the self-attention mechanism, which allows the model to weigh the importance of different words in a sequence when making predictions. This mechanism enables the model to capture long-range dependencies in the input sequence, making it more effective at processing and understanding natural language text.\n\nInstead of relying on recurrent neural networks (RNNs) or convolutional neural networks (CNNs) like previous NLP models, the Transformer is based entirely on self-attention layers. This makes it easier to parallelize training and allows the model to be more efficient at capturing dependencies across the input sequence.\n\nThe Transformer architecture consists of an encoder and a decoder component, each containing multiple layers of self-attention and feedforward neural networks. During training, the model is fed pairs of input and output sequences, where the goal is to predict the output sequence given the input sequence. The model learns to generate the output sequence word by word, taking into account the entire input sequence through the self-attention mechanism.\n\nOverall, the Transformer has shown state-of-the-art performance on a wide range of NLP tasks and has become a foundational building block for many advanced NLP models. Its ability to capture long-range dependencies and its parallelizability have made it a powerful tool for handling complex natural language data.', role='assistant', function_call=None, tool_calls=None)
PS C:\Users\ \Desktop> |
```