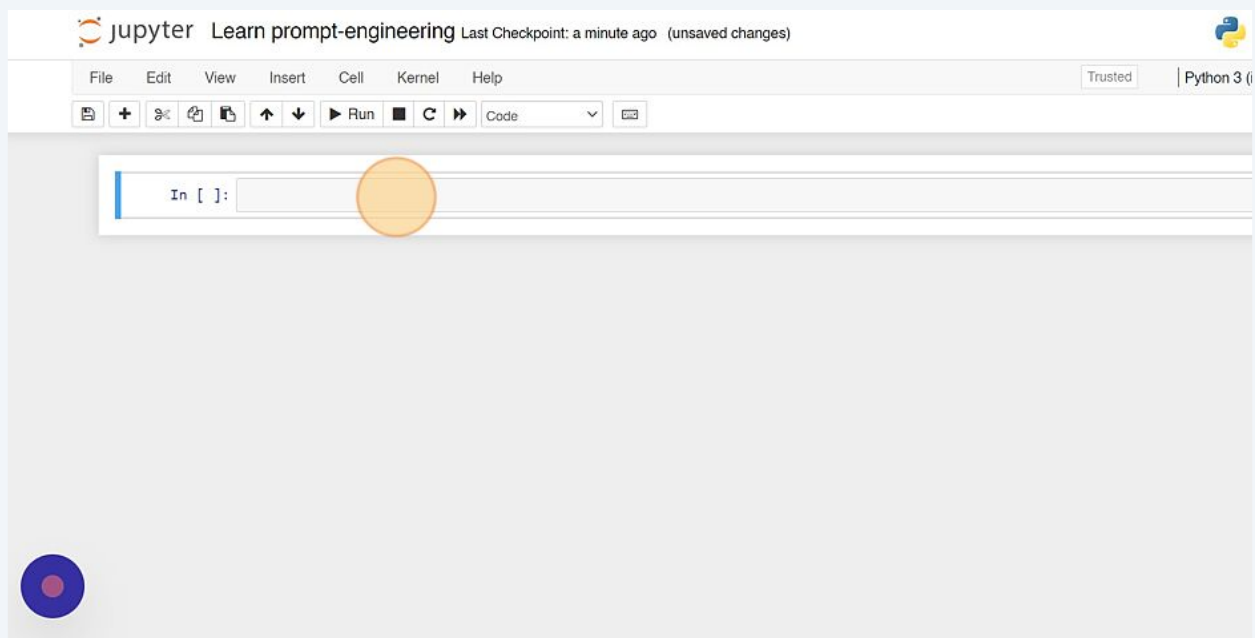


# How to use OpenAi api in jupyter

Note: In this attached link OpenAi (token/key) is added by default, so you do not need to add them manually.

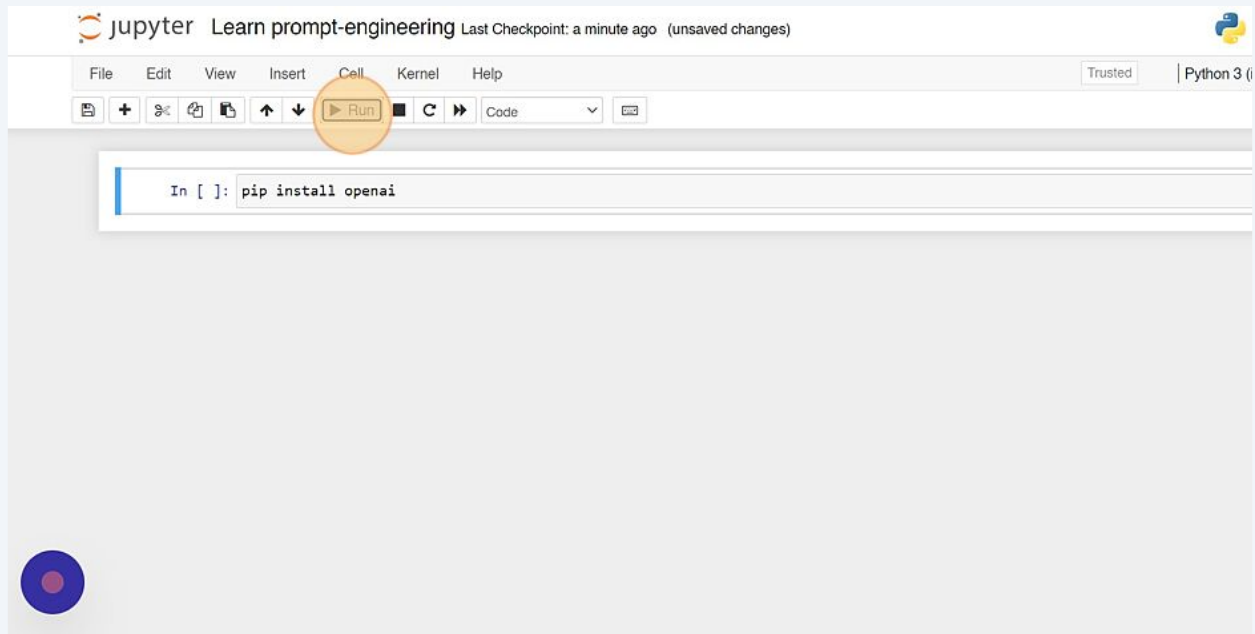
1 [s172-31-9-165p49722.lab-aws-production.deeplear...](https://s172-31-9-165p49722.lab-aws-production.deeplear...)

2 Click ""

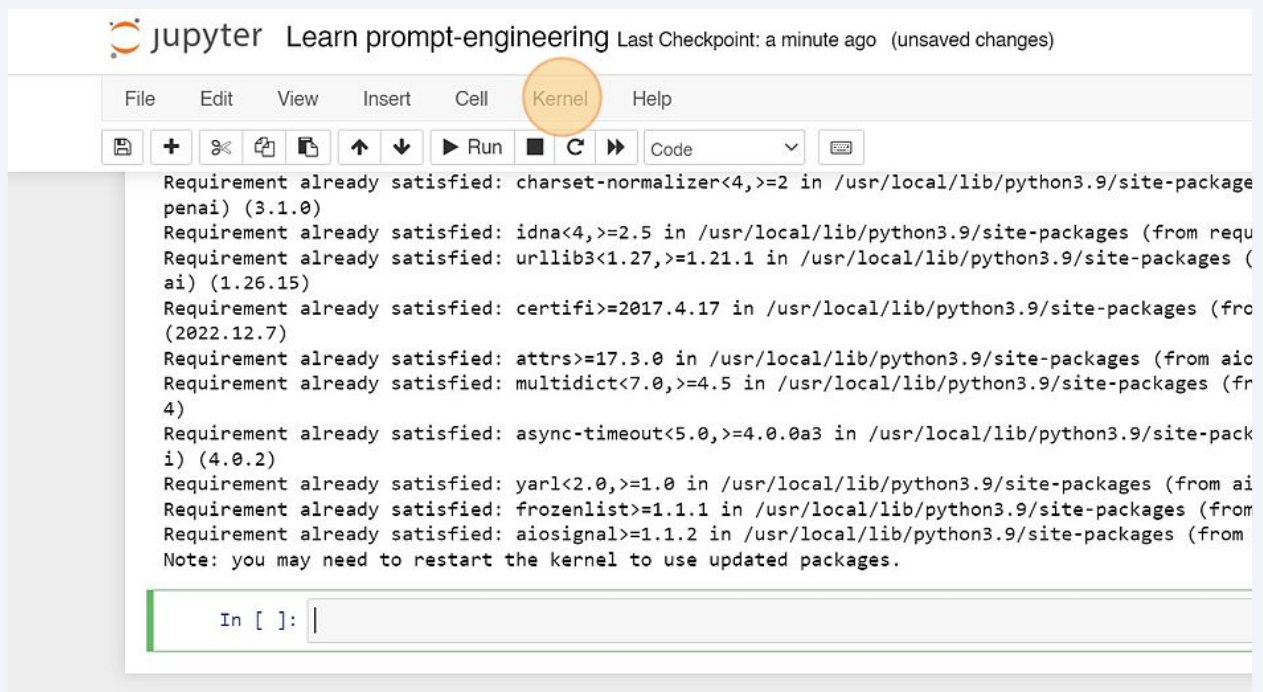


3 Type "pip install openai"

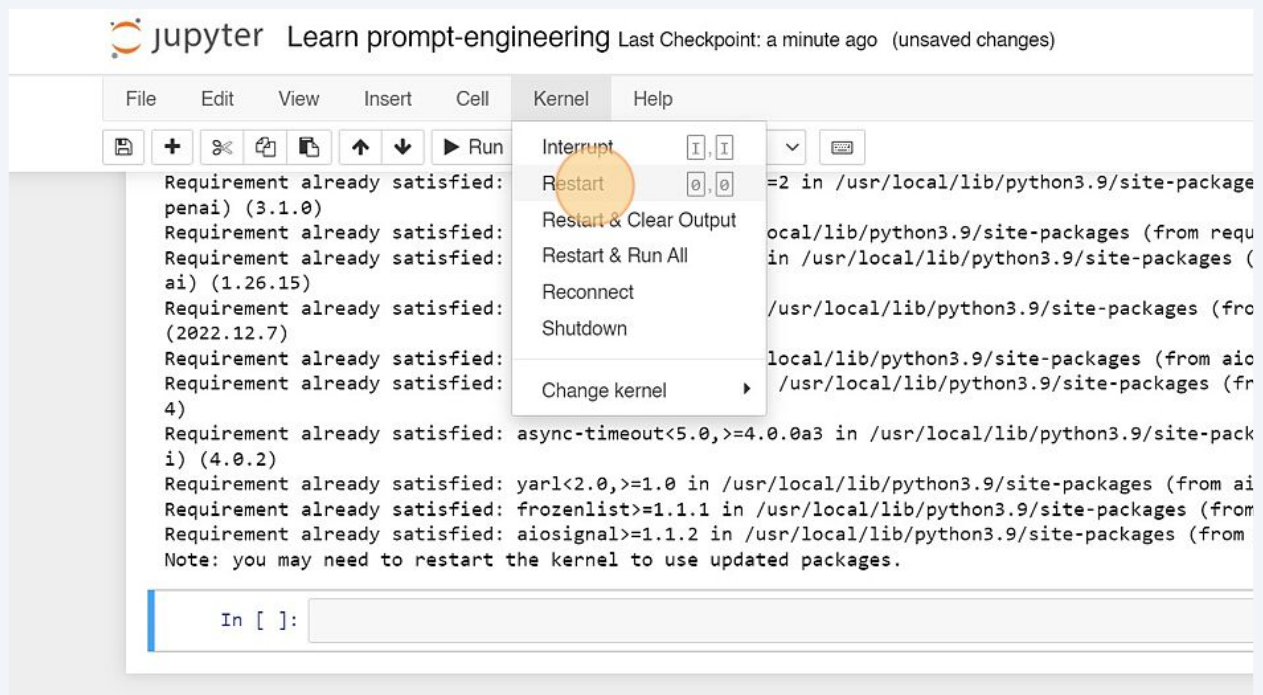
#### 4 Click "Run"



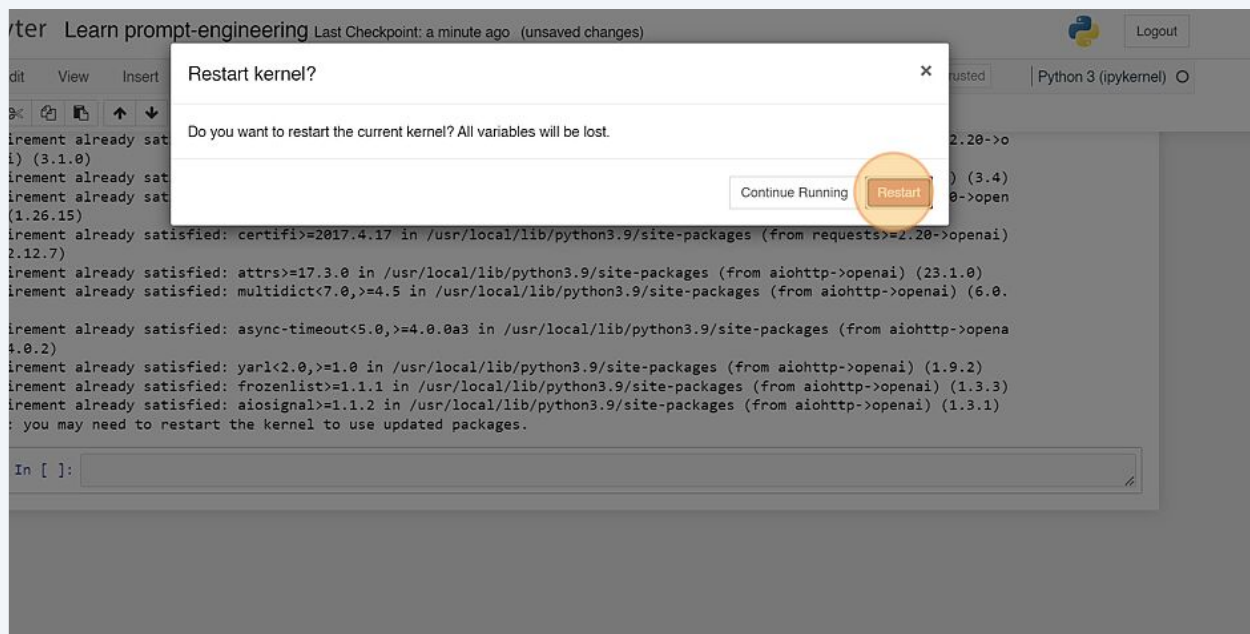
#### 5 Click "Kernel"



## 6 Click "Restart"

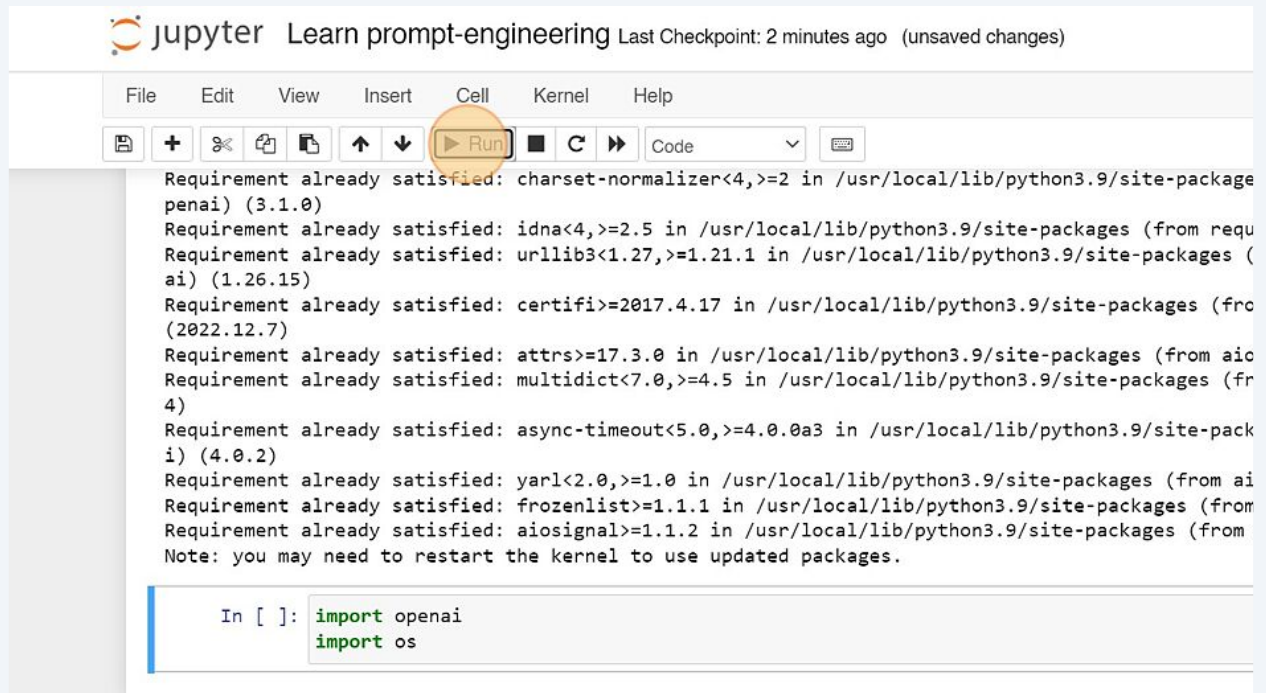


## 7 Click "Restart"



8 Type "import openai  
import os "

9 Click "Run"



Jupyter Learn prompt-engineering Last Checkpoint: 2 minutes ago (unsaved changes)

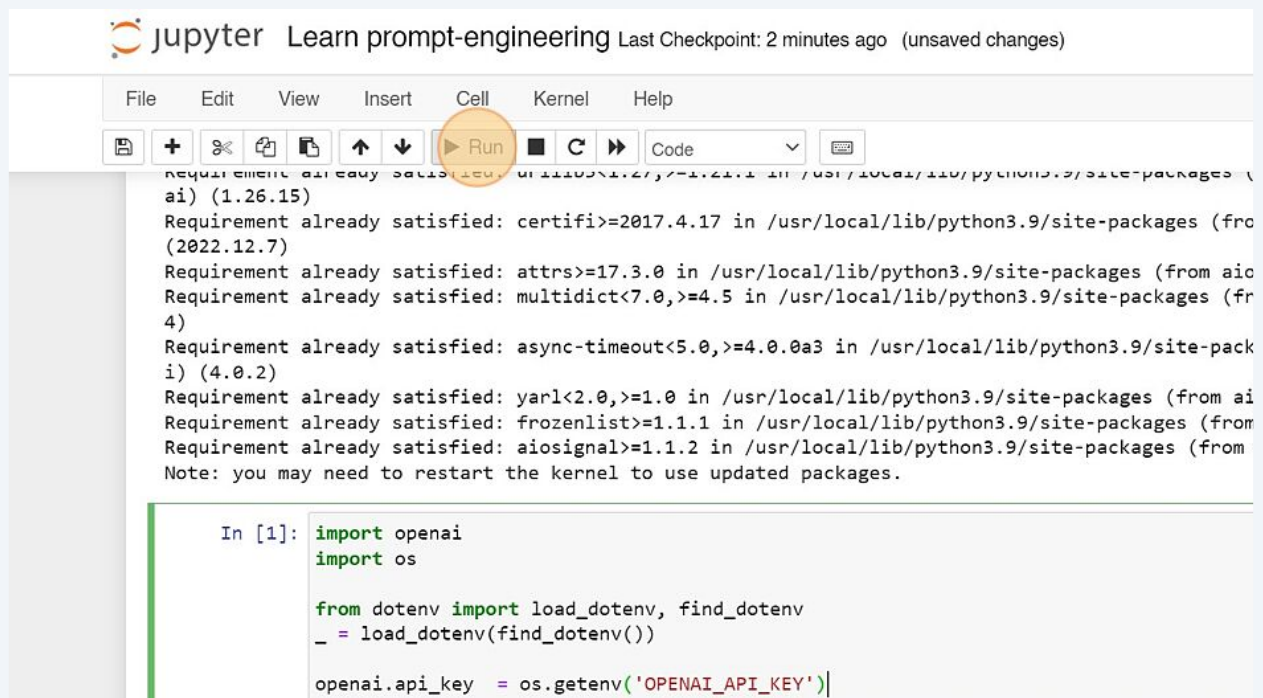
File Edit View Insert Cell Kernel Help

Run

```
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.9/site-packages (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/site-packages (from requ
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/site-packages (
ai) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/site-packages (fro
(2022.12.7)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.9/site-packages (from aio
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.9/site-packages (fr
4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.9/site-pack
i) (4.0.2)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.9/site-packages (from ai
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.9/site-packages (from
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.9/site-packages (from
Note: you may need to restart the kernel to use updated packages.
```

In [ ]: `import openai`  
`import os`

## 10 Now "write this code"



Jupyter Learn prompt-engineering Last Checkpoint: 2 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

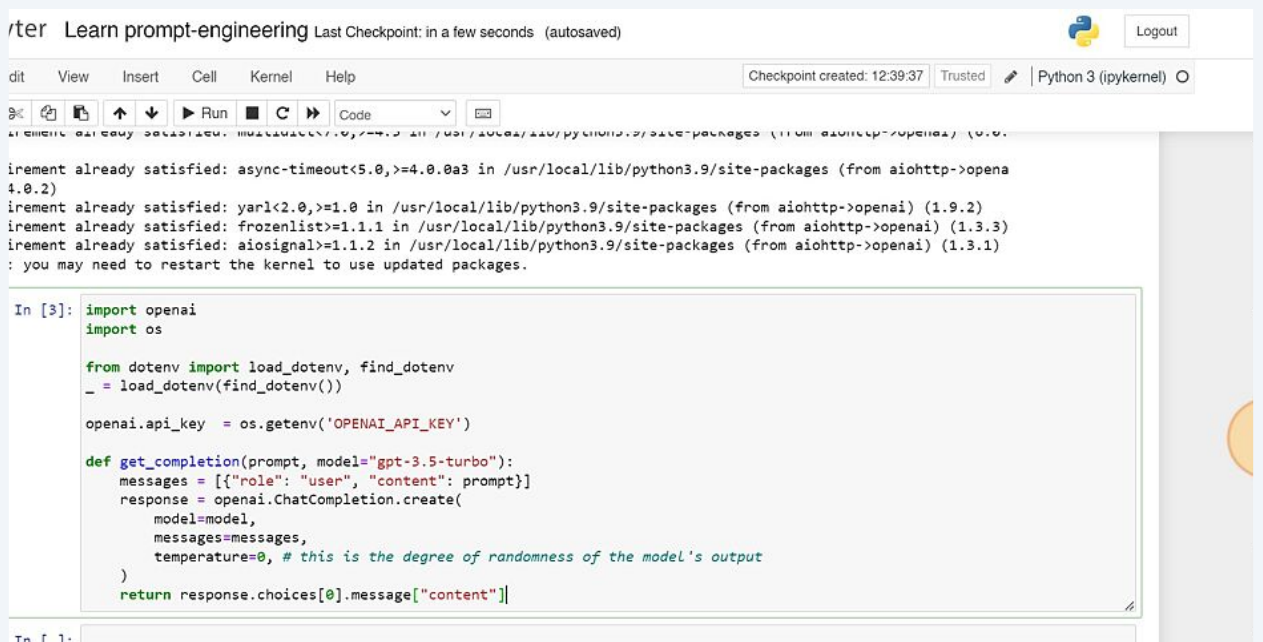
Requirement already satisfied: certifi in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (1.26.15)  
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (2022.12.7)  
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (21.2.0)  
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (4.7.6)  
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (4.0.2)  
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (1.8.2)  
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (1.3.3)  
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (1.3.1)  
Note: you may need to restart the kernel to use updated packages.

```
In [1]: import openai
import os

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv())

openai.api_key = os.getenv('OPENAI_API_KEY')
```

## 11 Now "define the function"



Learn prompt-engineering Last Checkpoint: in a few seconds (autosaved)

File Edit View Insert Cell Kernel Help

Checkpoint created: 12:39:37 Trusted Python 3 (ipykernel)

Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (4.7.6)  
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (4.0.2)  
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (1.8.2)  
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (1.3.3)  
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.9/site-packages (from aiohttp>openai) (1.3.1)  
Note: you may need to restart the kernel to use updated packages.

```
In [3]: import openai
import os

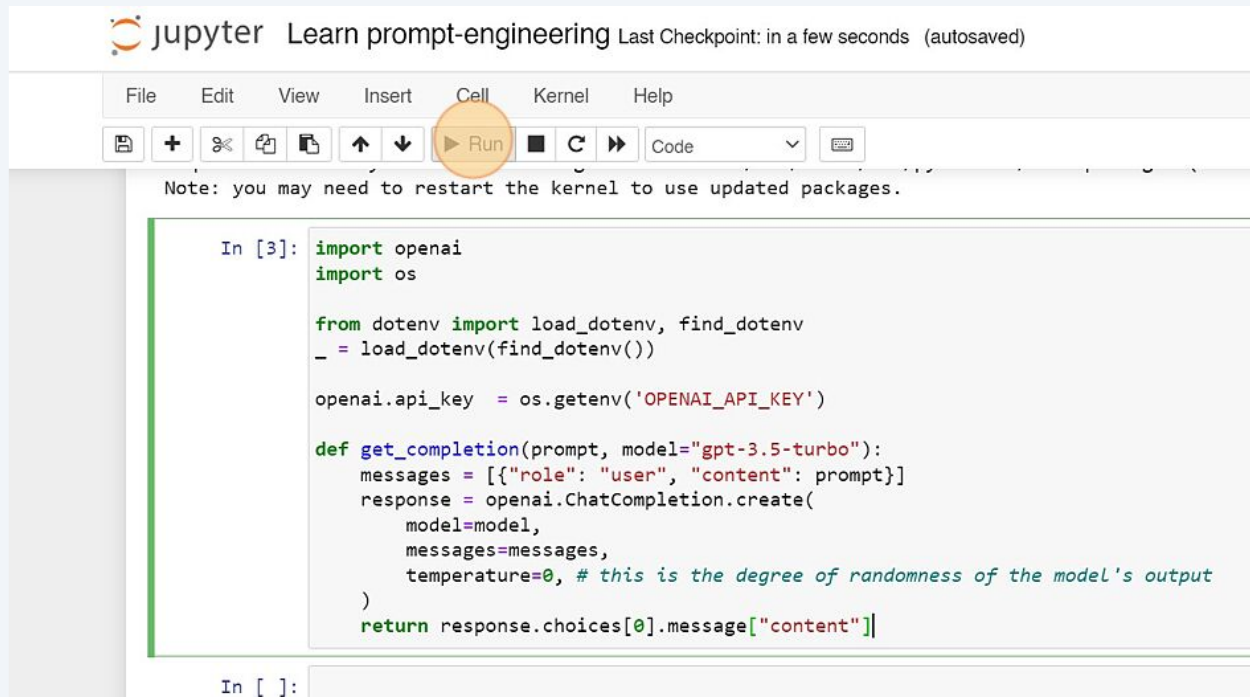
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv())

openai.api_key = os.getenv('OPENAI_API_KEY')

def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0, # this is the degree of randomness of the model's output
    )
    return response.choices[0].message["content"]
```



## 12 Click "Run"



Jupyter Learn prompt-engineering Last Checkpoint: in a few seconds (autosaved)

File Edit View Insert Cell Kernel Help

Note: you may need to restart the kernel to use updated packages.

```
In [3]: import openai
import os

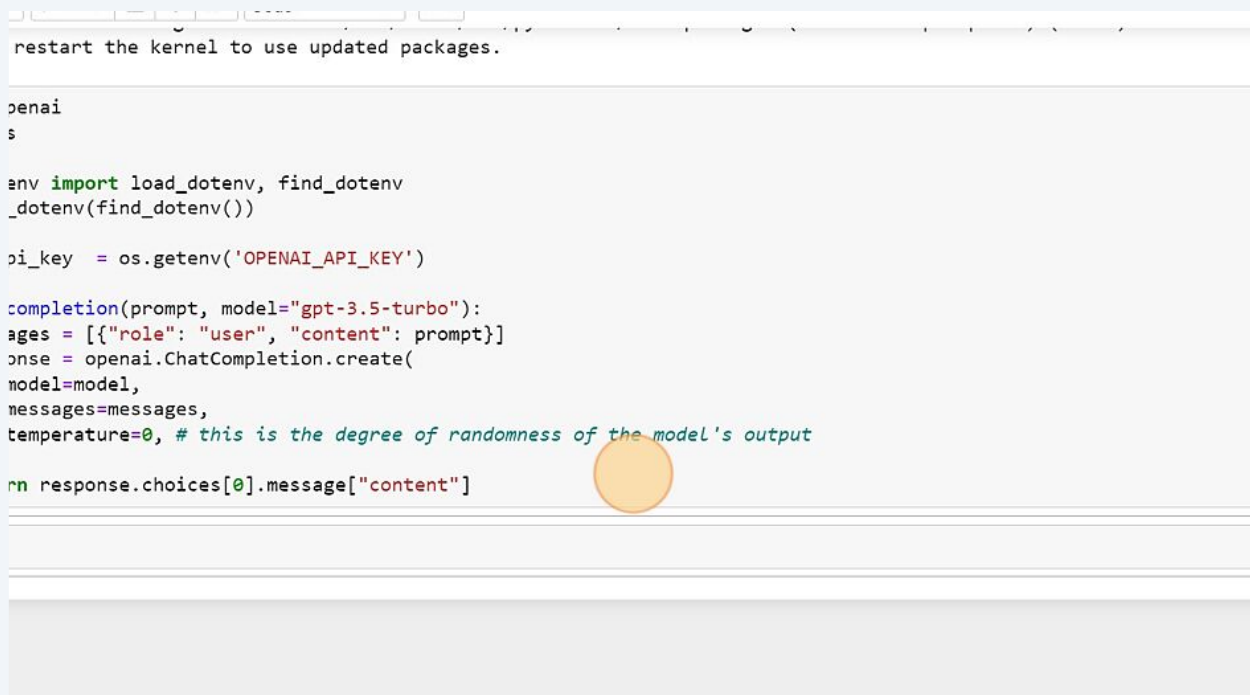
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv())

openai.api_key = os.getenv('OPENAI_API_KEY')

def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0, # this is the degree of randomness of the model's output
    )
    return response.choices[0].message["content"]
```

In [ ]:

## 13 Click "return response.choices[0].message[\"content\"]"



```
restart the kernel to use updated packages.

openai
s

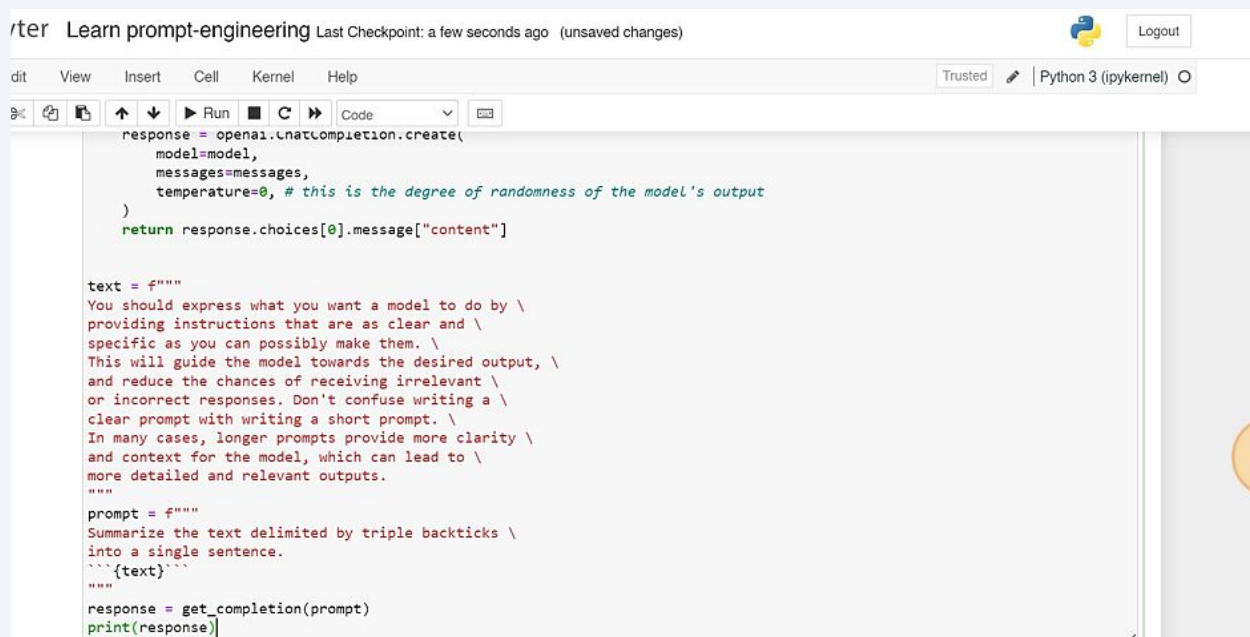
env import load_dotenv, find_dotenv
_dotenv(find_dotenv())

pi_key = os.getenv('OPENAI_API_KEY')

completion(prompt, model="gpt-3.5-turbo"):
ages = [{"role": "user", "content": prompt}]
onse = openai.ChatCompletion.create(
model=model,
messages=messages,
temperature=0, # this is the degree of randomness of the model's output

rn response.choices[0].message["content"]
```

## 14 Now "give the text (e.g) paragraph and set the prompt"



The screenshot shows a Jupyter Notebook window titled "Learn prompt-engineering". The code defines a function to create an OpenAI completion and a text prompt. The prompt is a detailed instruction on how to write prompts for a model.

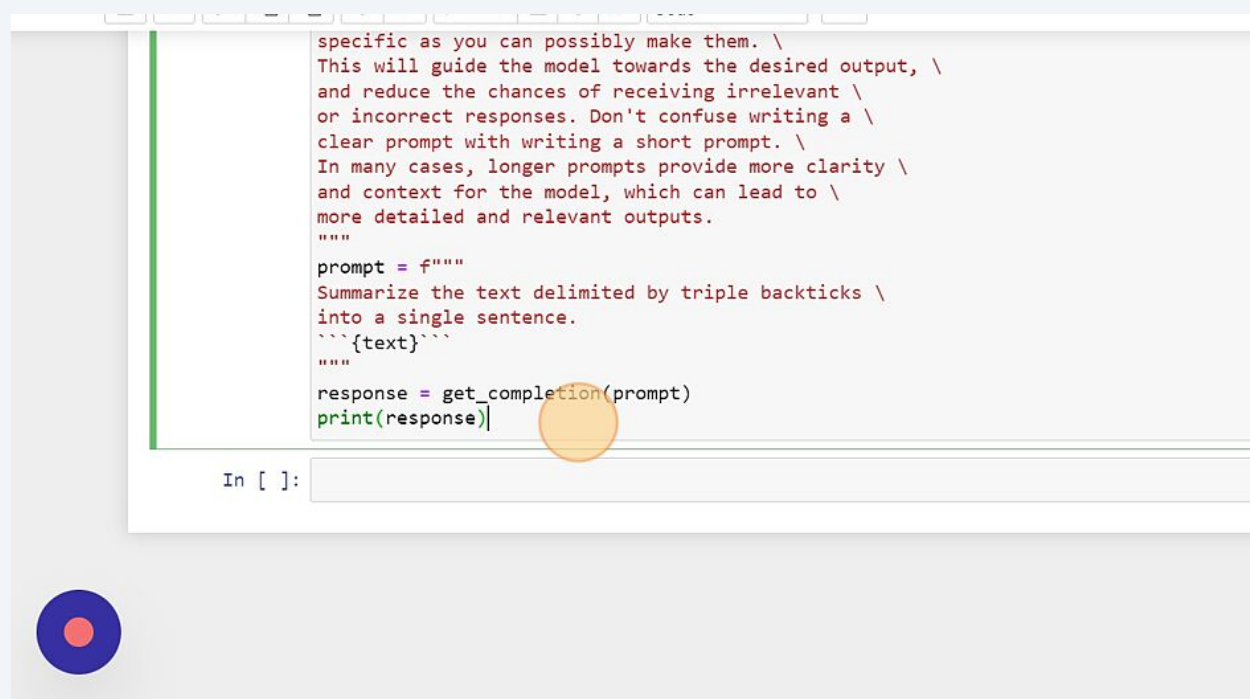
```
response = openai.ChatCompletion.create(
    model=model,
    messages=messages,
    temperature=0, # this is the degree of randomness of the model's output
)
return response.choices[0].message["content"]

text = f"""
You should express what you want a model to do by \
providing instructions that are as clear and \
specific as you can possibly make them. \
This will guide the model towards the desired output, \
and reduce the chances of receiving irrelevant \
or incorrect responses. Don't confuse writing a \
clear prompt with writing a short prompt. \
In many cases, longer prompts provide more clarity \
and context for the model, which can lead to \
more detailed and relevant outputs.
"""

prompt = f"""
Summarize the text delimited by triple backticks \
into a single sentence.
```{text}```
"""

response = get_completion(prompt)
print(response)
```

## 15 Now "print the response"



The screenshot shows the same Jupyter Notebook code as in the previous step, but with a focus on the execution. The code is highlighted, and the output area is visible, showing the prompt text and the function call. The prompt text is repeated from the previous step.

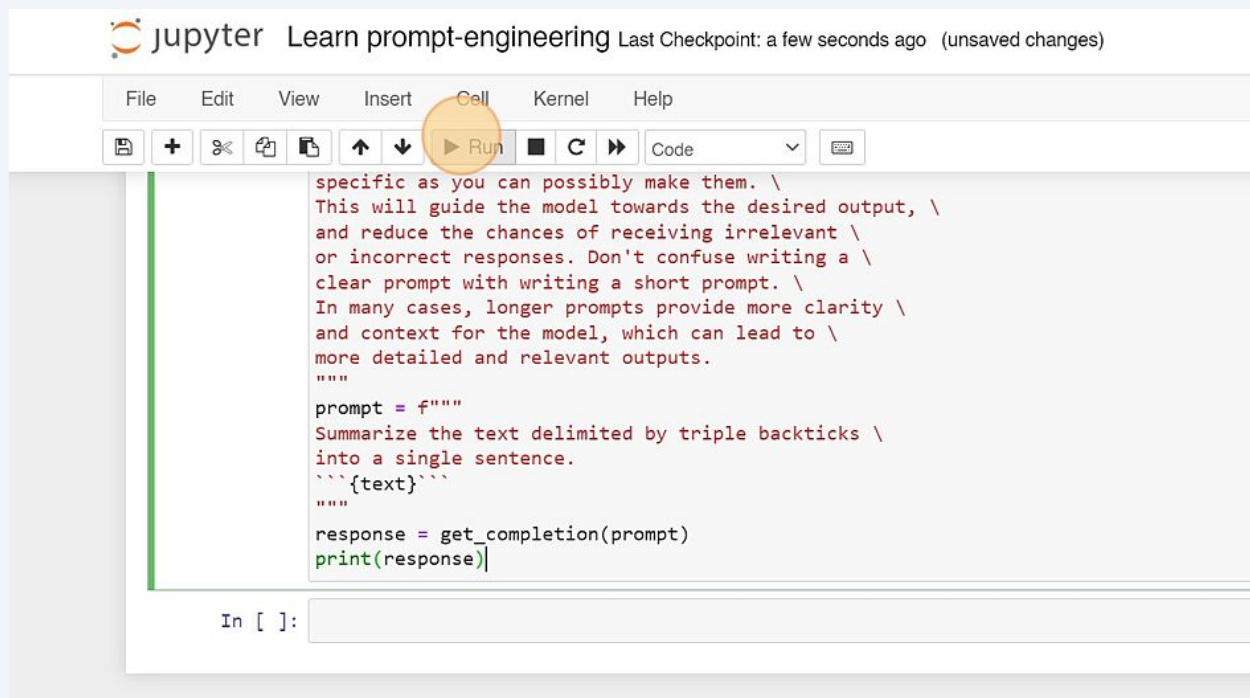
```
specific as you can possibly make them. \
This will guide the model towards the desired output, \
and reduce the chances of receiving irrelevant \
or incorrect responses. Don't confuse writing a \
clear prompt with writing a short prompt. \
In many cases, longer prompts provide more clarity \
and context for the model, which can lead to \
more detailed and relevant outputs.
"""

prompt = f"""
Summarize the text delimited by triple backticks \
into a single sentence.
```{text}```
"""

response = get_completion(prompt)
print(response)
```

In [ ]:

## 16 Now "finally run the code"



The image shows a JupyterLab interface. At the top, the title bar reads "Jupyter Learn prompt-engineering Last Checkpoint: a few seconds ago (unsaved changes)". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". Under the "Cell" menu, a "Run" button (a yellow circle with a right-pointing triangle) is highlighted with an orange circle. Below the menu bar is a toolbar with various icons, including a "Run" button (a right-pointing triangle) which is also highlighted with an orange circle. The main area contains a code cell with the following text:

```
specific as you can possibly make them. \
This will guide the model towards the desired output, \
and reduce the chances of receiving irrelevant \
or incorrect responses. Don't confuse writing a \
clear prompt with writing a short prompt. \
In many cases, longer prompts provide more clarity \
and context for the model, which can lead to \
more detailed and relevant outputs.
"""
prompt = f"""
Summarize the text delimited by triple backticks \
into a single sentence.
```{text}```
"""
response = get_completion(prompt)
print(response)
```

Below the code cell, the prompt "In [ ]:" is visible.

## 17 Now "you will see the answer in the form of summarized text"