



Using Python with (Chat)GPT to help automate Content Creation...

Andrew Timmins

Online Marketing Surgery Ltd



<https://yourseo.agency/brightonseo>



<https://github.com/andrewtimmins>

Disclaimer...

AI is advancing at such a rate
that by the time you view this,
some of this content may
already be out of date!



What is GPT?

GPT is a language model trained on a large amount of text taken from books, articles and websites and other sources.

**Why
automate
content
creation?**

Saves time...

Saves money...

Provides inspiration...

Respond quickly...

The first step
is to define
your content
type...

Blog Posts...

Product Descriptions...

Social Media Posts...

Email Newsletters...

FAQs...

What
automation
cannot
replace....

(...yet)

Humans!

Human intervention is
Still required to vet
and rework created
content.



What? I need
to vet and
modify
content?

AI isn't perfect.

What you provide to it in terms
of a subject, context and
noteworthy points,
really matter!



**An example
of non-vetted
piece of
content...**

“BrightonSEO started from a meet-up of a few digital marketers at a pub in Brighton in 2010, exchanging ideas on their working domains.

Today BrightonSEO conference with an info generating, stay-code boasting, elective hot result driven wham-information at #BrightonSEO presentations.

Bark Conversion speaking on Instant Fancy Cards shortly.”

Do I need to
be one with
code to get
started?

Not at all.

**OpenAI provides
a handy Web Portal
to use Chat GPT.**

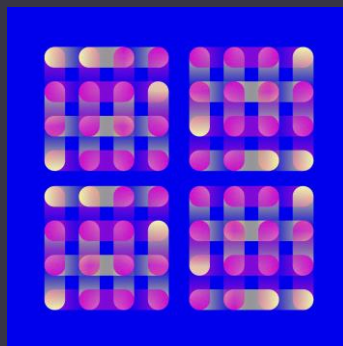
<https://chat.openai.com/>

I love to code
and want
to get my
developer
on...



So, what do
you use at
OMS?

GPT-3



Chat GPT

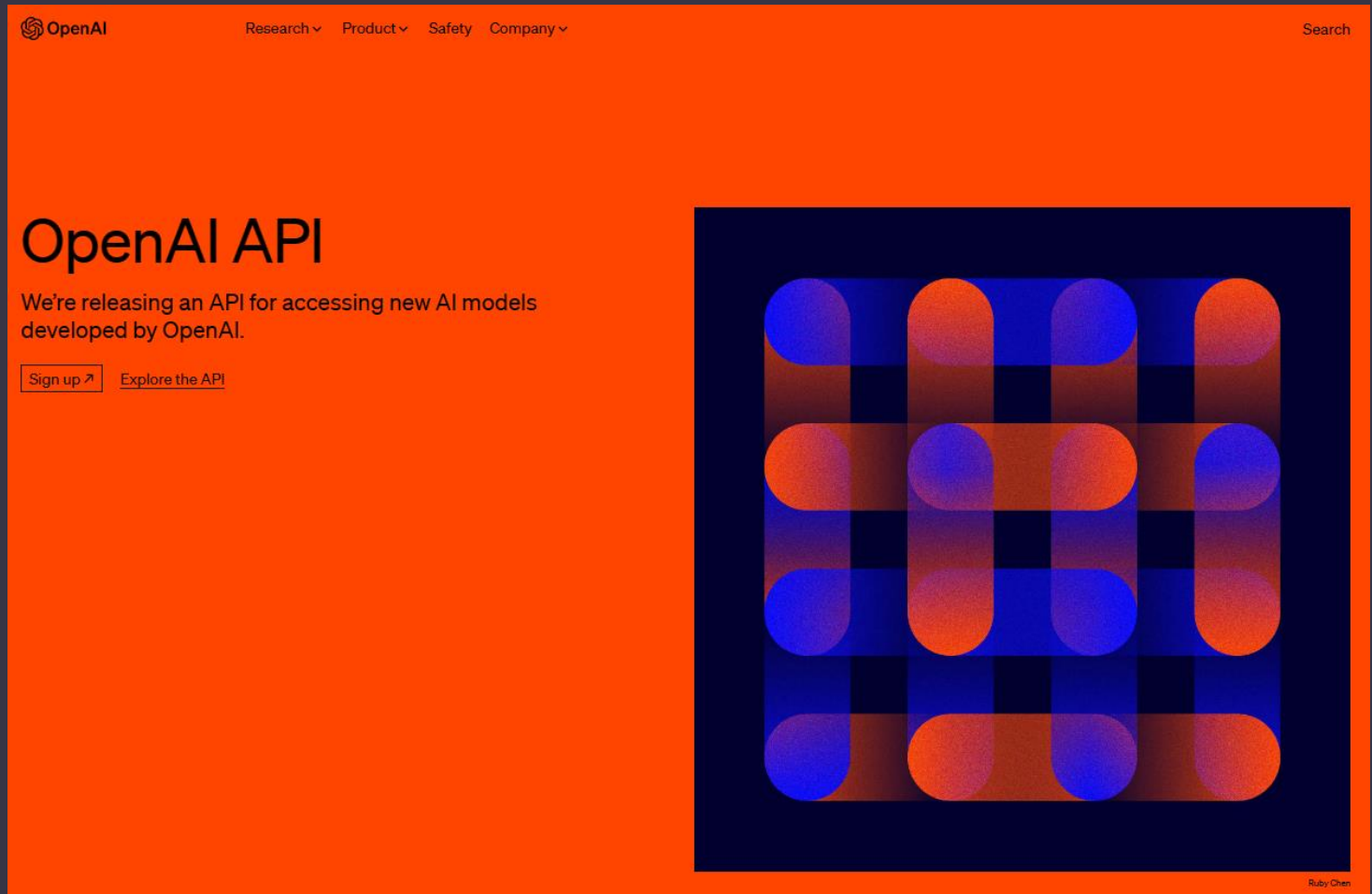


Python



Create
yourself an
OpenAI
Account...

<https://platform.openai.com/signup>



Create an API Key...

Remember to keep your key safe!

The screenshot shows the OpenAI API Keys management interface. The top navigation bar includes links for Overview, Documentation, Examples, and Playground, along with an Upgrade button, a Help icon, and the user profile 'Online Marketing Surgery'. The left sidebar lists various settings under 'ORGANIZATION' and 'USER', with 'API Keys' highlighted under the 'USER' section. The main content area is titled 'API keys' and contains a warning about the security of API keys. Below the warning is a table with one API key listed. A button to 'Create new secret key' is visible. At the bottom, there is a section for 'Default organization' with a dropdown menu set to 'Online Marketing Surg...' and a note about specifying the organization for each API request.

SECRET KEY	CREATED	LAST USED
sk-...DGG0	13 Mar 2023	Never

[+ Create new secret key](#)

Default organization

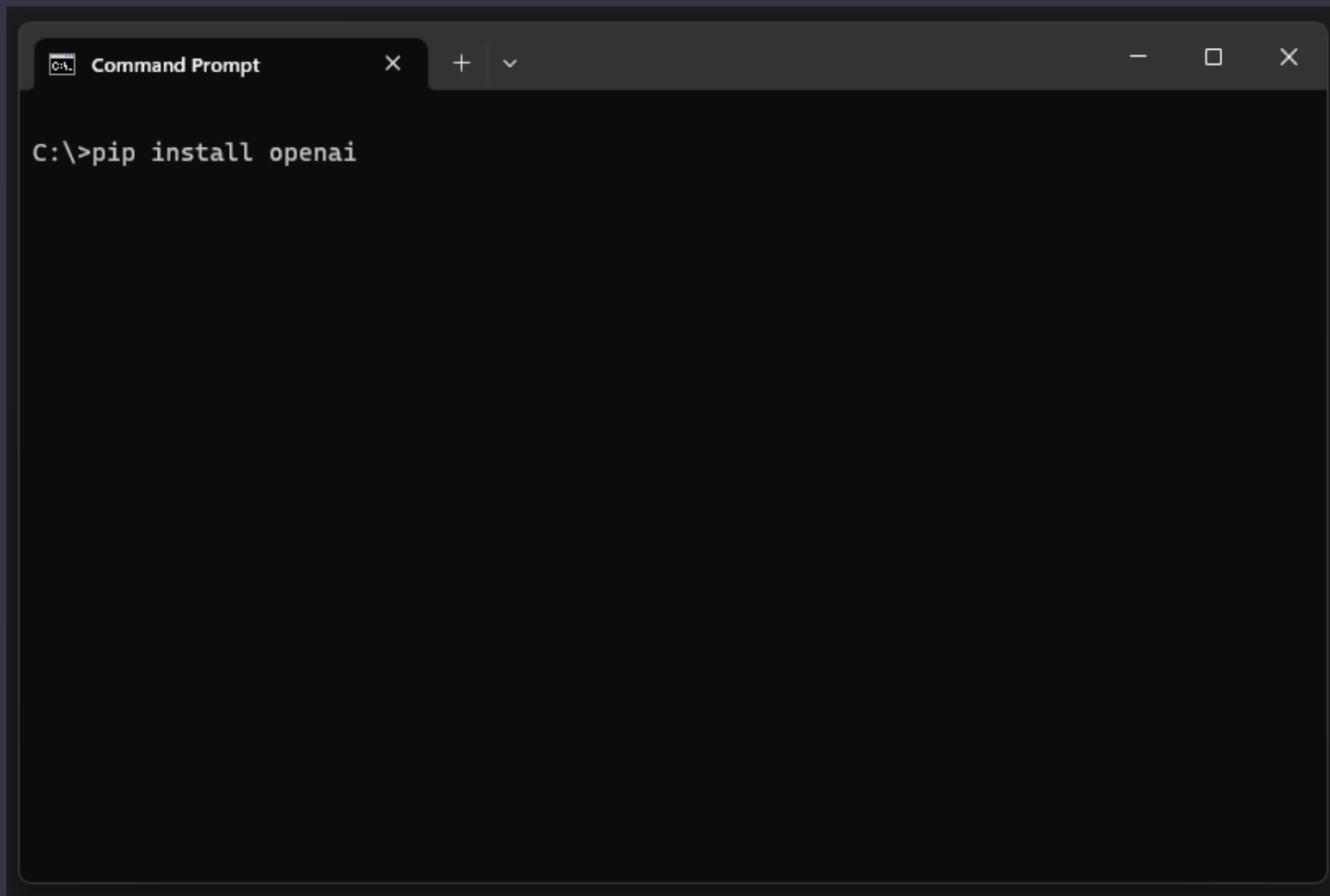
If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.

Online Marketing Surg... ▾

Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.

Install The OpenAI API client...

> pip install openai

A screenshot of a Windows Command Prompt window. The title bar at the top reads "Command Prompt" and includes standard window controls (close, maximize, minimize). The command prompt shows the command "C:\>pip install openai" entered at the prompt. The background of the window is black, and the text is white.

How to use
the API...

Constructing a completion...

Setting parameters...

Sending the request...

Handling the response...

Constructing a completion...

```
completion = openai.ChatCompletion.create(  
    model = 'gpt-3.5-turbo',  
    messages = [  
        {"role": "user", "content": "Write a summary of the history of the BrightonSEO event."}  
    ],  
    temperature = 0  
)
```

Model = The model we're going to use (e.g. GPT or Davinci)

Role = System, User or Assistant

Temperature = 0: more accurate, 2: more random

Sending the request...

```
aiResponse = completion['choices'][0]['message']['content'])
```

Handling the response...

```
print(aiResponse)
```

Putting it all together...

```
import openai
openai.api_key = "***key here***"

completion = openai.ChatCompletion.create(
    model = 'gpt-3.5-turbo',
    messages = [
        {"role": "user", "content": "Write a summary of the history of the BrightonSEO event."}
    ],
    temperature = 0
)

print(completion['choices'][0]['message']['content'])
```

Example output from our script...

“BrightonSEO is a digital marketing conference that takes place twice a year in Brighton, UK.

The event was founded in 2010 by Kelvin Newman, who was working as a freelance SEO consultant at the time.

The first BrightonSEO conference was held in a pub with just 35 attendees, but it quickly grew in popularity and moved to larger venues.”

So, this is cool,
but how can
we automate
creation?



**Automating
WordPress
Posts...**

Scrape news site...

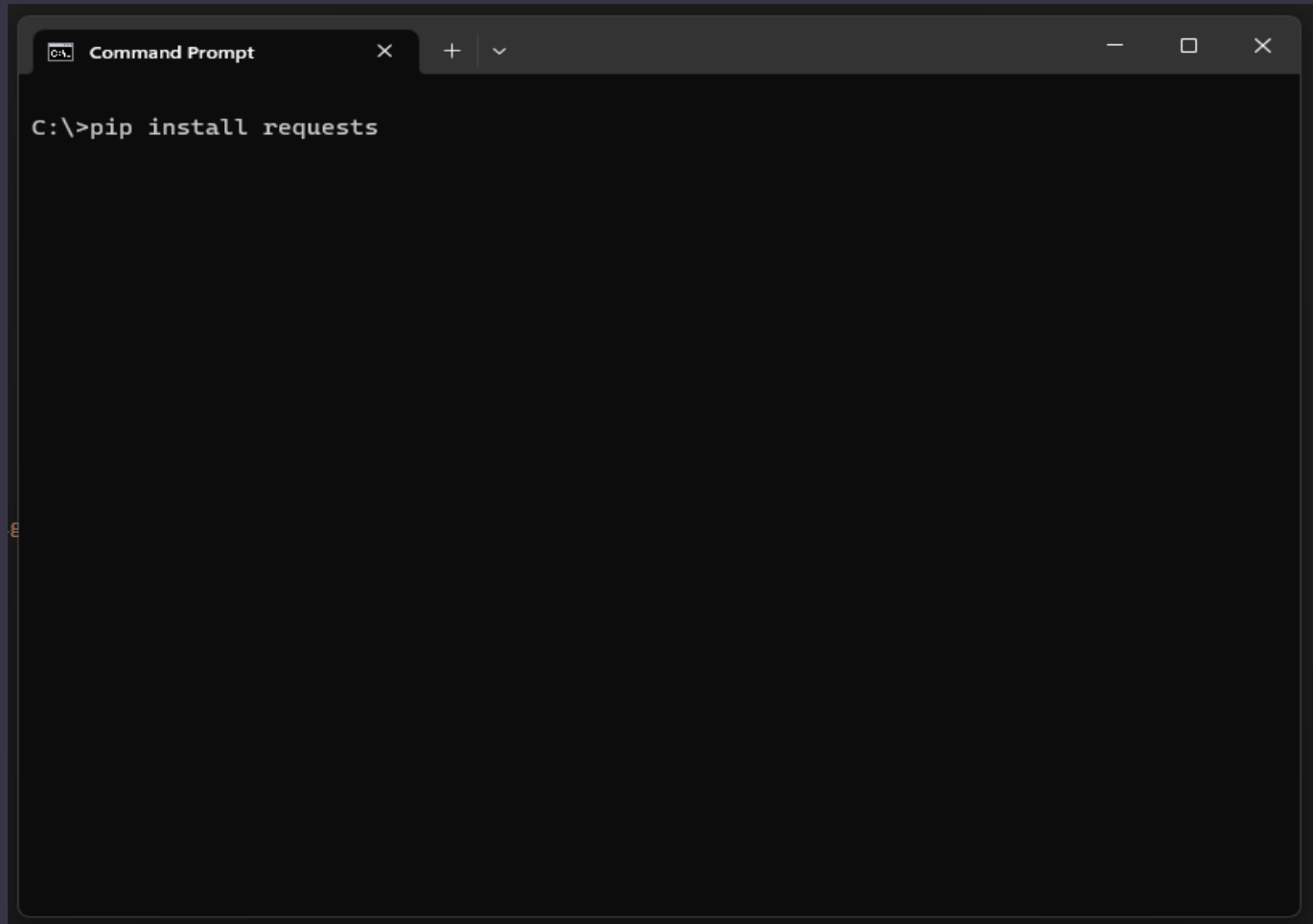
Send scraped content to GPT...

Create draft post...

Check & publish post...

Install The Requests library...

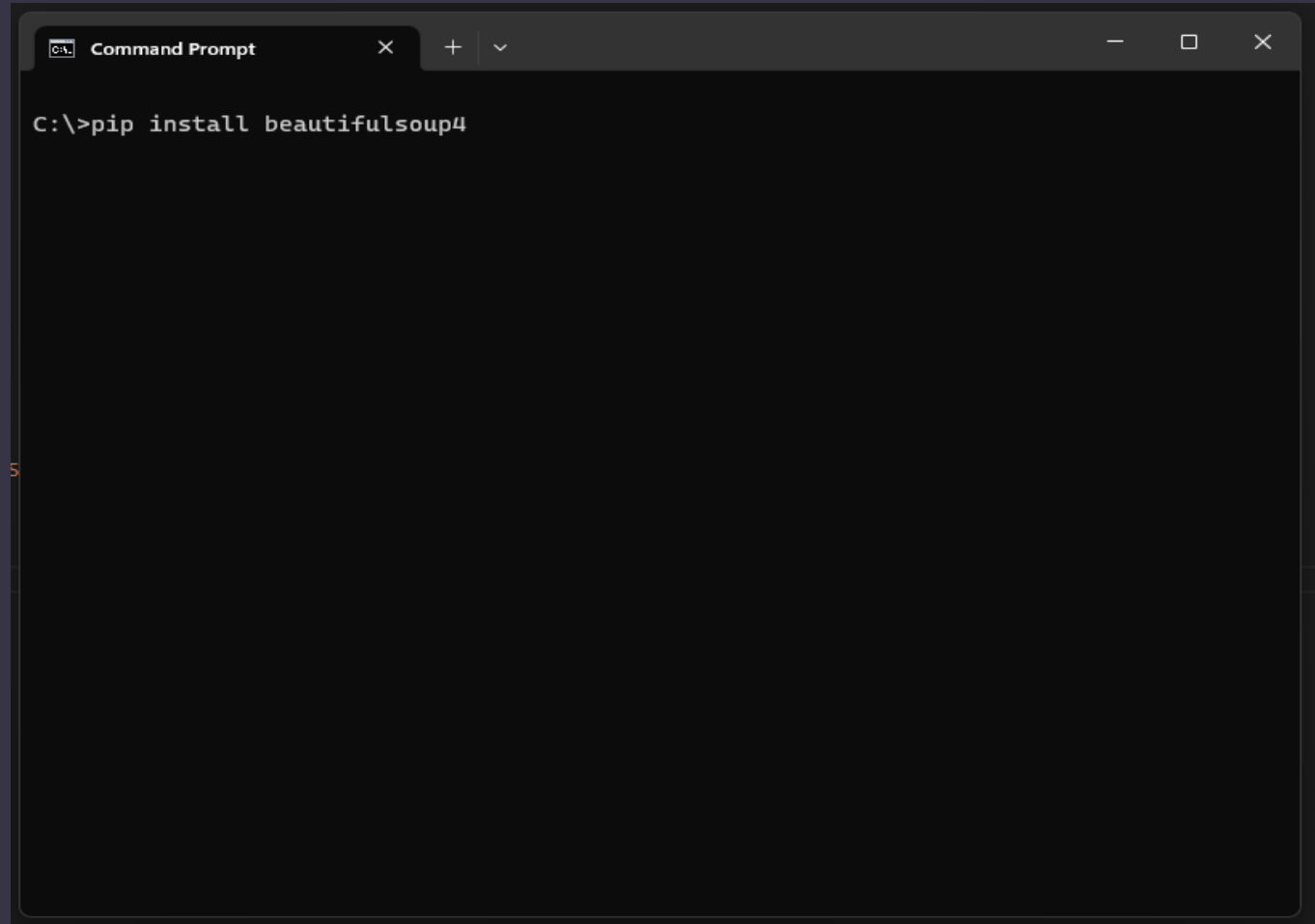
> pip install requests



```
Command Prompt
C:\>pip install requests
```


Install The Beautiful Soup library...

> pip install beautifulsoup4



```
Command Prompt
C:\>pip install beautifulsoup4
```

Scraping the news site...

```
import requests
from bs4 import BeautifulSoup

# Configurable options
newsURL = "http://feeds.bbc.co.uk/news/rss.xml"

# Fetch the 10 latest news article
xmlResponse = requests.get(newsURL)
xmlSoup = BeautifulSoup(xmlResponse.text,
    "html.parser")
newsArticles = xmlSoup.find_all('item')
current_item=0
for item in range(10):
    articleURL = newsArticles[current_item].guid.text
    articleHeadLine =
newsArticles[current_item].title.text
    current_item+=1

print(newsArticles[0].title.text)
print(newsArticles[0].guid.text)
```

```
# Get the first news article in the list, we're not
worrying about the others in this example
articleResponse = requests.get(newsArticles[0].guid.text)
articleSoup = BeautifulSoup(articleResponse.text,
    "html.parser")
# Find all instances of <P></P> tags
articleScrapedContent = articleSoup.find_all('p')
articleText = ""
for p_tag in range(len(articleScrapedContent)):
    if p_tag < 0:
        pass
    else:
        articleText = articleText +
articleScrapedContent[p_tag].text

print(articleText)
```

Send content to GPT...

```
# Build the completion based on the extracted content
for the content
contentCompletion = openai.ChatCompletion.create(
    model = 'gpt-3.5-turbo',
    messages = [
        {"role": "user", "content": "Rewrite the
following text to be more concise: " + articleText}
    ],
    temperature = 0
)

# Place our new content into the variable
wpPostContent
wpPostContent =
contentCompletion['choices'][0]['message']['content']
```

```
# Build the completion based on the extracted content for
the content
titleCompletion = openai.ChatCompletion.create(
    model = 'gpt-3.5-turbo',
    messages = [
        {"role": "user", "content": "Rewrite the following
text to be more concise: " + newsArticles[0].title.text}
    ],
    temperature = 0
)

# Place our new slug into the variable wpSlug,
transforming it to be URL safe
wpSlug =
titleCompletion['choices'][0]['message']['content']
wpSlug = wpSlug.lower()
wpSlug = wpSlug.replace(" ", "-")
```

Create the draft post...

```
wpUser = "***username here***"
wpPaassword = "***password here***"
wpURL = 'https://***url here***/wp-json/wp/v2/posts'

# Create the authentication header for Wordpress
wpCredentials = wpUser + ":" + wpPaassword
wpToken = base64.b64encode(wpCredentials.encode())
wpHeader = {'Authorization': 'Basic ' +
wpToken.decode('utf-8'), 'Content-Type':
'application/json'}

# Place our new slug into the variable wpSlug,
transforming it to be URL safe
wpSlug =
titleCompletion['choices'][0]['message']['content']
wpSlug = wpSlug.lower()
wpSlug = wpSlug.replace(" ", "-")
```

```
#print(titleCompletion['choices'][0]['message']['content'])
#print(wpPostContent)
#print(wpSlug)

# Send the new post to Wordpress in a draft state
wpData = {
'title' :
titleCompletion['choices'][0]['message']['content'],
'status': 'draft',
'slug' : wpSlug,
'content': wpPostContent
}

wpResponse = requests.post(wpURL, headers=wpHeader,
json=wpData)
print(wpResponse)
```

Putting it all together...

```
import requests
import base64
import openai
from bs4 import BeautifulSoup
import warnings

# Configurable options
openai.api_key = "****key here****"
wpUser = "****username here****"
wpPaassword = "****password here****"
wpURL = 'https://****url here****/wp-
json/wp/v2/posts'
newsURL = "http://feeds.bbc.co.uk/news/rss.xml"

# Create the authentication header for Wordpress
wpCredentials = wpUser + ":" + wpPaassword
wpToken = base64.b64encode(wpCredentials.encode())
wpHeader = {'Authorization': 'Basic ' +
wpToken.decode('utf-8'),
'Content-Type': 'application/json'}

# Fetch the 10 latest news article
xmlResponse = requests.get(newsURL)
xmlSoup = BeautifulSoup(xmlResponse.text,
"html.parser")
newsArticles = xmlSoup.find_all('item')
current_item=0
for item in range(10):
    articleURL = newsArticles[current_item].guid.text
    articleHeadLine =
newsArticles[current_item].title.text
    current_item+=1
```

```
#print(newsArticles[0].title.text)
#print(newsArticles[0].guid.text)

# Get the first news article in the list, we're not worrying
about the others in this example
articleResponse = requests.get(newsArticles[0].guid.text)
articleSoup = BeautifulSoup(articleResponse.text,
"html.parser")
articleScrapedContent = articleSoup.find_all('p')
articleText = ""
for p_tag in range(len(articleScrapedContent)):
    if p_tag < 0:
        pass
    else:
        articleText = articleText +
articleScrapedContent[p_tag].text

#print(articleText)

# Build the completion based on the extracted content for
the content
contentCompletion = openai.ChatCompletion.create(
    model = 'gpt-3.5-turbo',
    messages = [
        {"role": "user", "content": "Rewrite the following text
to be more concise: " + articleText}
    ],
    temperature = 0
)

# Place our new content into the variable wpPostContent
wpPostContent =
contentCompletion['choices'][0]['message']['content']
```

```
# Build the completion based on the extracted content for the
content
titleCompletion = openai.ChatCompletion.create(
    model = 'gpt-3.5-turbo',
    messages = [
        {"role": "user", "content": "Rewrite the following text
to be more concise: " + newsArticles[0].title.text}
    ],00
    temperature = 0
)

# Place our new slug into the variable wpSlug, transforming
it to be URL safe
wpSlug = titleCompletion['choices'][0]['message']['content']
wpSlug = wpSlug.lower()
wpSlug = wpSlug.replace(" ", "-")

#print(titleCompletion['choices'][0]['message']['content'])
#print(wpPostContent)
#print(wpSlug)

# Send the new post to Wordpress in a draft state
wpData = {
    'title' :
titleCompletion['choices'][0]['message']['content'],
    'status': 'draft',
    'slug' : wpSlug,
    'content': wpPostContent
}

wpResponse = requests.post(wpURL,headers=wpHeader,
json=wpData)
print(wpResponse)
```

The content scraped from a news article...

Controversial social media influencer Andrew Tate and his brother Tristan are to be moved from custody to house arrest with immediate effect, a Romanian judge has ruled.

The ruling by the Court of Appeal in Bucharest replaces the latest period of custody, which was to end on 29 April.

Two associates, Georgiana Naghel and Luana Radu, are also being released."

**That content,
output from
GPT, ready for
our post...**

Social media influencer Andrew Tate and his brother Tristan, along with two associates, have been moved from custody to house arrest by a Romanian judge.

The ruling replaces their latest period of custody, which was due to end on 29

The four have been ordered to remain in their homes unless they have judicial permission to leave.



Making use of
the example
code...

A few ideas.

Change the input

Change the output

Add in additional services

Share your code

Useful URLs...

OpenAI:
<https://openai.com/>

GPT Demo:
<https://beta.openai.com/demo/>

EleutherAI:
<https://www.eleuther.ai/>



Thank You!

Andrew Timmins

Online Marketing Surgery Ltd



<https://yourseo.agency/brightonseo>



<https://github.com/andrewtimmins>