ChatGPT API with Python and tKinter

Setup

```
sudo apt install python3-tk

sudo apt install pip

pip3 install openai
```

Rate Limits

https://platform.openai.com/docs/guides/rate-limits/overview
3 Requests per Minute for Free Tier

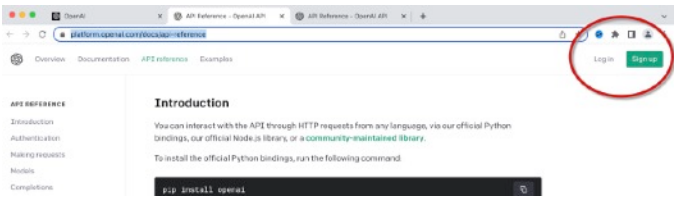
Tokens

750 Words = 1000 Tokens (maybe…?)
$0.002 = 1,000 Tokens (One Fifth of a Penny)
Query and Response count towards Token Cost

Create an Account

Make sure to create an account for the API and not for the Platform. When you login to the API Dashboard you are simply given documentation pages and such, there is no ability to interact with ChatGPT from the API Dashboard.

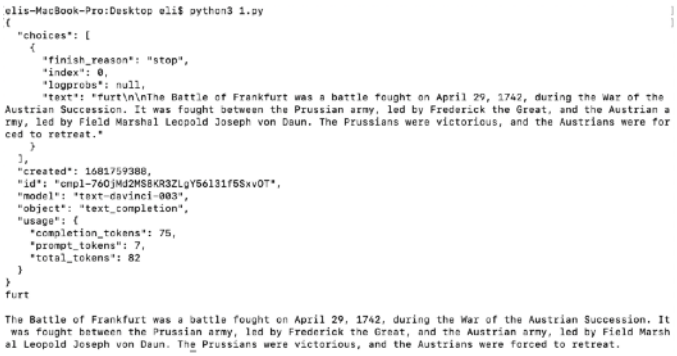
$18 in Credit for first 3 Months
https://platform.openai.com/docs/api-reference



1 - Basic 3.0 Davinci Model

This lab shows you how make an API call and the results from the 3.0 Davinci model. This model seems to give more concise answers then 3.5 Turbo which may be useful if you have to parse results to provide information to your users.

max_tokens = Maximum number of tokens to use
stop = Character to stop on (generally remove)

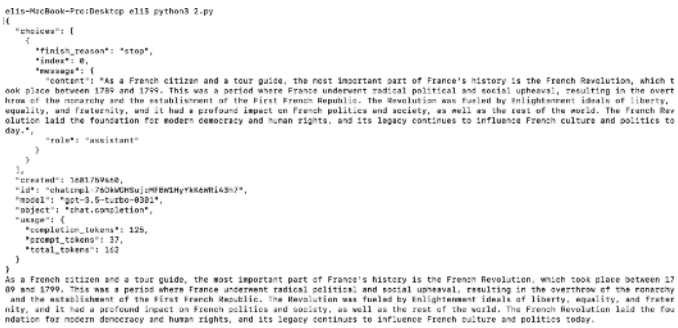



```
import openai

openai.api_key = ‘API KEY'

query = "what was the battle of the frank"

response = openai.Completion.create(
 model="text-davinci-003",
 prompt=query,
 temperature=0,
 max_tokens=1000,
 top_p=1,
 frequency_penalty=0.0,
 presence_penalty=0.0,
)

print(response)

print(response["choices"][0]["text"])
```

2 - Basic 3.5 Turbo Model

This lab shows you how to use the 3.5 Turbo model. With this model you can give GPT a role to and build in guiding language to skew results into a particular direction. This can be useful for dynamically creating answers based on information about a user. Such as asking “Who was president in 2000”, and the answer being based on the nationality of the asker.

There are 3 roles for GPT in this model. The System role states the “character” GPT should answer as (Advisor, Travel Agent, Professor, Boss), and there can be only 1. The Assistant role provides context for the question and there can be a number of them. The User role states the query the user is making, and there can be only 1.




```
import openai

openai.api_key = ‘API KEY'
nationality = "france"
query = "what was the most important part of your history"

response = openai.ChatCompletion.create(
 model="gpt-3.5-turbo",
 messages=[
   {"role": "system", "content": "you are a tour guide"},
   {"role": "assistant", "content": "answer as a " + (nationality) + " citizen"},
   {"role": "user", "content": query}
 ]
)

print(response)
print(response["choices"][0]["message"]["content"])
```

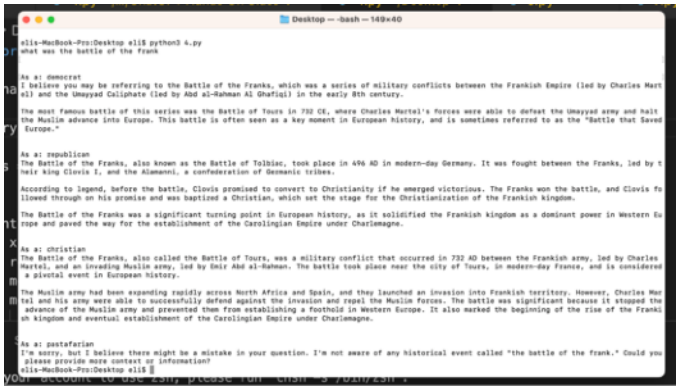
3 - Compare 3.0 to 3.5 Results

This lab shows the difference in results between 3.0 and 3.5. There are practical reasons that the older model might provide better results for your particular project.

```
import openai

openai.api_key = 'API KEY'

query = "what was the battle of the frank"

response3 = openai.Completion.create(
 model="text-davinci-003",
 prompt=query,
 temperature=0,
 max_tokens=1000,
 top_p=1,
 frequency_penalty=0.0,
 presence_penalty=0.0,
)

response35 = openai.ChatCompletion.create(
 model="gpt-3.5-turbo",
 messages=[
  {"role": "system", "content": ""},
  {"role": "assistant", "content": ""},
  {"role": "user", "content": query}
 ]
)

print("GPT 3.0 Says:")
print(response3["choices"][0]["text"])

print("\nGPT 3.5 Turbo Says:")
print(response35["choices"][0]["message"]["content"])
```

4 - Building in Bias

This lab looks at how bias coded into the prompt you provide for ChatGPT can change the results in major and unexpected ways. Bias is not simply an issue ingrained into ChatGPT and AI, but also into the code that makes the API call.

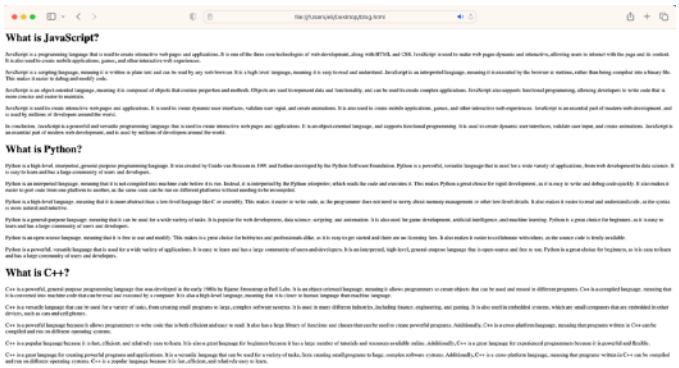



```
import openai

openai.api_key = ‘API KEY'

query = "what was the battle of the frank"

bias = ["democrat", "republican", "christian", "pastafarian"]

print(query)
for x in bias:
   response = openai.ChatCompletion.create(
   model="gpt-3.5-turbo",
   messages=[
      {"role": "system", "content": ""},
      {"role": "assistant", "content": "as a" + x},
      {"role": "user", "content": query}
   ]
   )

   print("\n\nAs a: " + x)
   print(response["choices"][0]["message"]["content"])
```

5 - Auto Blog

This lab shows you how to automatically create and fill a blog with unique blog posts. In a real world setting you would dump the results into a database for a CMS (Content Management System) to use, but here we are writing to a simple HTML file.

We use the 3.0 model because we are able to ask for the results to be formatted automatically into HTML so that we do not have to do any additional parsing of the returns before writing to the file.



What is JavaScript?

What is Python?

What is C++?

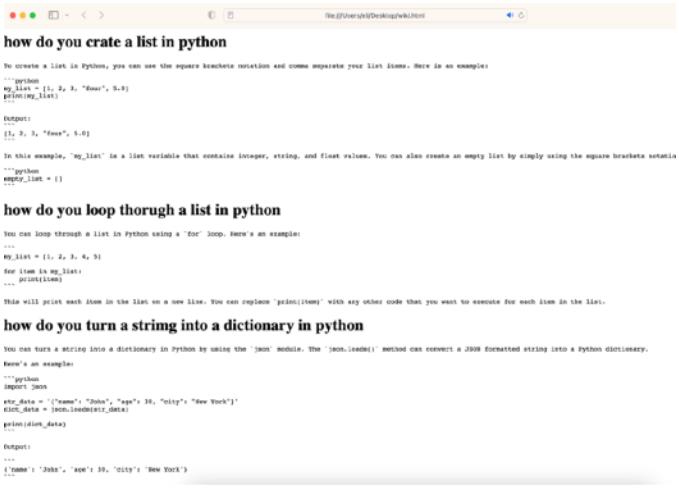
```
import openai

openai.api_key = 'API KEY'

posts = ["typescript", "go", "objective c", "dart", "golang"]

for query in posts:
   query = "what is " + query + " answer in 500 word blog post formatted in html"
   response = openai.Completion.create(
   model="text-davinci-003",
   prompt=query,
   temperature=0,
   max_tokens=1000,
   top_p=1,
   frequency_penalty=0.0,
   presence_penalty=0.0,
   )

   file = open("blog.html", "a")
   answer = (response["choices"][0]["text"])
   input = (answer)
   file.write(input)
```

```
file.close()
print("\n\n" + query)
print(answer)
```

6 - Build Wiki

This lab allows users to ask questions and have the results be limited to a specific topic. The answers are then written to an HTML page as a demonstration of dumping to some type of permanent data store. This could be useful if you want to create a local wiki platform that is augmented by AI.

Learning to prompt AI can be more frustrating then you may first think. If you don't spell out exactly what you want returned the results can be odd, and strangely passive aggressive.



how do you crate a list in python

To create a list in Python, you can use the square brackets notation and comma separate your list items. Here is an example:

```python
my_list = [1, 2, 3, "four", 5.0]
print(my_list)
```

Output:

```
[1, 2, 3, "four", 5.0]
```

In this example, `my_list` is a list variable that contains integer, string, and float values. You can also create an empty list by simply using the square brackets notatio

```python
empty_list = []
```

how do you loop thorugh a list in python

You can loop through a list in Python using a `for` loop. Here's an example:

```
my_list = [1, 2, 3, 4, 5]

for item in my_list:
    print(item)
```

This will print each item in the list on a new line. You can replace `print(item)` with any other code that you want to execute for each item in the list.

how do you turn a strimg into a dictionary in python

You can turn a string into a dictionary in Python by using the `json` module. The `json.loads()` method can convert a JSON formatted string into a Python dictionary.

Here's an example:

```python
import json

str_data = '{"name": "John", "age": 30, "city": "New York"}'
dict_data = json.loads(str_data)

print(dict_data)
```

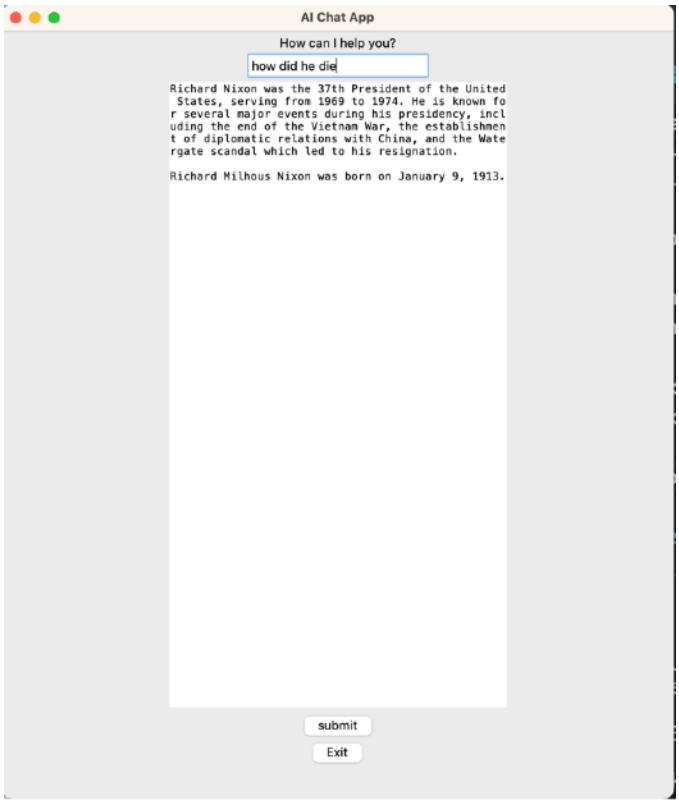
Output:

```
{'name': 'John', 'age': 30, 'city': 'New York'}
```

```
import openai

openai.api_key = 'API KEY'

query = "how do you sort a dictionary by key in python"

bias = "Only answer questions related to programming.  If the question is not programming specific reply with 'this is not a coding question' and nothing else."

response = openai.ChatCompletion.create(
model="gpt-3.5-turbo",
messages=[
   {"role": "system", "content": ""},
   {"role": "assistant", "content":  bias},
   {"role": "user", "content": query}
]
)
file = open("wiki.html", "a")
answer = (response["choices"][0]["message"]["content"])
```

```
input = ("<h1>") + (query) + ("</h1>") + ("<pre>") + (answer) + ("</pre>")
file.write(input)
file.close()
print(query)
print(answer)
```

7 - Conversation App




```
from tkinter import *
import openai

openai.api_key = 'API KEY'

window = Tk()
window.geometry("600x600")
window.title("AI Chat App")

lbl = Label(window, text="How can I help you?").pack()

ent = Entry(window)
ent.pack()

txt = Text(window, height = 25, width = 50)
txt.pack()

conversation = ""
```

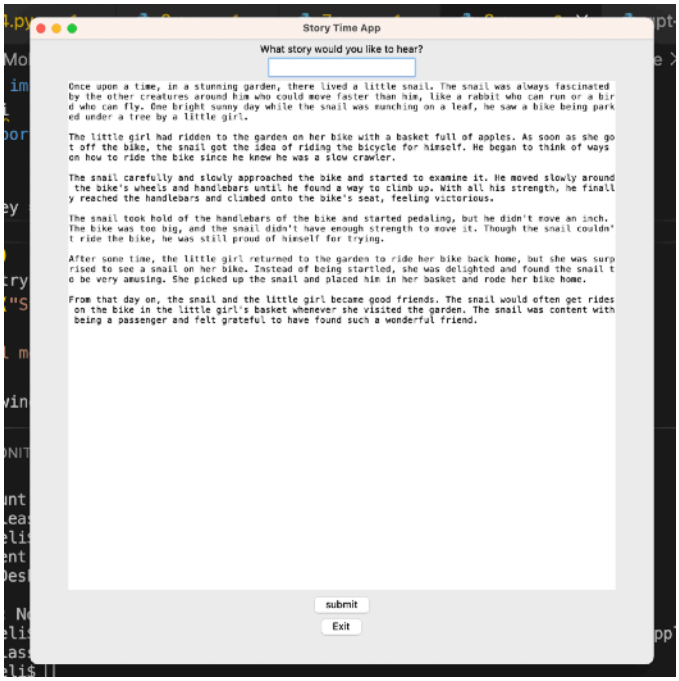
```
def ask(event):
    global conversation
    query = ent.get()
    ent.delete(0,END)
    conversation = conversation + "\n\n" + query
    response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "system", "content": "you are an advisor"},
        {"role": "assistant", "content": conversation},
        {"role": "user", "content": query}
    ]
    )
    answer = response["choices"][0]["message"]["content"]
    txt.insert(END, answer + "\n\n")

btn = Button(window, text="submit", command=ask)
btn.pack()

btn_exit = Button(window, text ="Exit", command=window.destroy).pack()

window.bind("<Return>", ask)

window.mainloop()
```

8 - Bedtime Story

In this lab we ask ChatGPT to write a story and then that story is read out loud using Google Text to Speech. We also resubmit the previoius story so that ChatGPT can continue from where it stopped. This lab is for having students start thinking of inputs and outputs beyond simple text formatting.

gTTS makes an API call to Google's services. You don't need to add a Google API key for this, but please note I'm not sure of rate limits or restrictions. The standard gTTS has pricing and API keys, but this module doesn't seem to require them.

gTTS = Google Text to Speech Module




```
pip3 install gTTS

sudo apt install mpg123
```

```
from tkinter import *
import openai
from gtts import gTTS
import os

openai.api_key = 'API KEY'

window = Tk()
window.geometry("600x600")
window.title("Story Time App")
```

```
story = "Tell me a story about"

lbl = Label(window, text="What story would you like to hear?").pack()

ent = Entry(window)
ent.pack()

txt = Text(window, height=25, width=50)
txt.pack()

def ask(event):
    global story
    query = ent.get()
    ent.delete(0, END)
    story = story + "\n\n" + query
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "you are a story teller"},
            {"role": "assistant", "content": story},
            {"role": "user", "content": query}
        ]
    )
    answer = response["choices"][0]["message"]["content"]
    txt.insert(END, answer + "\n\n")
    speech = gTTS(answer)
    speech.save('speech.mp3')
    os.system("mpg123 speech.mp3") #MacOS = afplay, Ubuntu = mpg123
    os.system("rm speech.mp3")

btn = Button(window, text="submit", command=ask)
btn.pack()

btn_exit = Button(window, text="Exit", command=window.destroy).pack()

window.bind("<Return>", ask)

window.mainloop()
```