Prompt1

October 5, 2023

[ ]:
```
import os
import openai

prompt = input('Enter Question: ')

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

response = openai.ChatCompletion.create(
    model='gpt-4',
    messages=[
        {'role': 'system', 'content': 'You are expert data scientist with␣
 ↪extensive experience with Jupyter Notebooks and Python'},
        {'role': 'user', 'content': prompt}
    ],
    temperature=1,
    max_tokens=5000,
    top_p=1,
    frequency_penalty=0.75,
    presence_penalty=0.34,
    stop=["--stop"]
)

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

There are several libraries in Python that you can use for visualizing geographic or map data. Among these the most popular ones are GeoPandas, Matplotlib, Bokeh and Plotly. Here's a basic example of how to use GeoPandas:

To install geopandas:

```
!pip install geopandas
```

Then you can use it.

```
import geopandas as gpd

# load a sample dataset provided by geopandas package.
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```

```
# plot the whole world
world.plot()

# plot a specific country(for instance, 'Brazil')
brazil = world[world.name=="Brazil"]
brazil.plot()
```

When you want more interactive maps or dealing with web based maps (like Google Maps), then Plotly and Folium provide great flexibility in that sense.

Here is an example using folium:

First Install it via pip :

```
!pip install folium
```

Using folium library for creating maps:

```
import folium

# Create a Map instance
m = folium.Map(location=[-23.550520, -46.633308], zoom_start=12)

m.save("map.html")
```

This will create base map at specified location with defined zoom level. You can add markers, popups and many more to your map using respective methods available.

Please pay attention: during working with geographical information (especially if your task should provide high precision) it's important to understand projection systems and always check what projection system used in your raw data.

[ ]:
```
import os
import openai

python_code = input('Enter Python Code: ')

openai.api_key = os.getenv("OPENAI_KEY")

response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[
    {
      "role": "system",
      "content": "You will be provided with a piece of Python code, and your␣
 ↪task is to find and fix bugs in it. Please also try and optimize the code."
    },
    {
      "role": "user",
```

```
      "content": python_code
    }
  ],
  temperature=0.52,
  max_tokens=8000,
  top_p=1,
  frequency_penalty=0.75,
  presence_penalty=0.34,
  stop=["--stop"]
)

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

The code provided has some syntax issues. It appears to be trying to load environment variables from a .env file and then print the value of the 'HOME' variable. Here's a corrected version:

```
from dotenv import load_dotenv
import os

load_dotenv()

print(os.getenv('HOME'))
```

In this corrected version, each statement is properly placed on its own line. The function `load_dotenv()` is called without any arguments, which means it will look for a .env file in the same directory as where your script is running from.

As for optimization, this script is already quite simple and doesn't have much room for further optimization. However, you might want to handle the case when 'HOME' environment variable does not exist in your .env file:

```
from dotenv import load_dotenv
import os

load_dotenv()

home = os.getenv('HOME')

if home:
    print(home)
else:
    print("'HOME' environment variable not found.")
```

In this new version of the code, if there's no 'HOME' variable found in your .env or system environment variables, it prints an error message instead of None.
