Hello

October 5, 2023

1 Hello Jupyter Notebooks

Let's make sure python environment is properly activated and installed. Python Version 3.8 should be selected from the conda environment.

[ ]:
```
import sys
print(sys.version)
```

```
3.8.18 (default, Sep 11 2023, 13:40:15)
[GCC 11.2.0]
```

1.0.1 This completion component can be used to ask Data Science or Jupyter Notebook related questions

[ ]:
```
import os
import openai

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': 'What are the best graphing or␣
 ↪visualization tools for Jupyter Notebooks?'}
    ]
)

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

```
1. Matplotlib: This is the most used Python library for 2D graphics. It allows
easy creation of static, animated, and interactive visualizations in Python and
offers flexibility in controlling every aspect of the graph.

2. Seaborn: Seaborn library is based on Matplotlib and provides a higher-level
interface for creating well-designed graphics. It includes some high level,
dataset-oriented plotting routines.
```

3. Plotly: Plotly supports many types of charts and graphs that are not available in other libraries. It provides tools for interactive and real-time graphs. Plotly integrates with Jupyter notebooks, allowing interactive plots within the notebook.

4. Bokeh: Bokeh is a Python library capable of producing interactive and scalable visualizations in a browser using JavaScript widgets. It's suitable for creating interactive plots, dashboards, and data applications.

5. ggplot: ggplot is a plotting system for Python based on R's ggplot2 and the Grammar of Graphics. Though not as flexible as Matplotlib, it makes creating more complex graphics easier.

6. Altair: This library provides a simple API for generating complex visualizations using a declarative syntax. It's built on top of Vega and Vega-Lite, JavaScript visualization libraries.

7. Pandas Visualization: Pandas itself has straightforward, high-level functions for creating a variety of plots like line, bar, scatter, histogram, etc. It's built on Matplotlib and aims to integrate well within a scientific computing environment with many other 3rd party libraries.

Remember, the choice of visualization tool often depends heavily on the specific needs of the data science project, as well as on personal preference. Each of the libraries listed above have their unique strongest aspects and capabilities, making each more suitable for certain scenarios than others.

1.0.2 This component can be used to debug any Python script/file you have.

[ ]:
```
import os
import openai

# python_code = input()

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."
    },
    {
      "role": "user",
```

```
      "content": "import Random\na = random.randint(1,12)\nb = random.
 ↪randint(1,12)\nfor i in range(10):\n    question = \"What is \"+a+\" x␣
 ↪\"+b+\"? \"\n    answer = input(question)\n    if answer = a*b\n       ␣
 ↪print (Well done!)\n    else:\n        print(\"No.\")"
    }
  ],
  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 has several bugs:

1. Python is case-sensitive and the module `random` is all lower-case.
2. The variables `a` and `b` should be inside the loop if you want to generate different questions each time.
3. In Python, string concatenation of numbers with strings requires explicit conversion of number to string using str() function.
4. In if statement, we use '==' for comparison not '=' which is used for assignment.
5. Strings in print statements must be enclosed in quotes.

Here's the corrected code:

```python
import random
for i in range(10):
    a = random.randint(1,12)
    b = random.randint(1,12)
    question = "What is "+ str(a) +" x "+ str(b) +"? "
    answer = int(input(question))
    if answer == a*b:
        print ("Well done!")
    else:
        print("No.")
```

This will now correctly ask 10 different multiplication questions and check the user's answers.