# Hello1

October 5, 2023

## 1 Hello Jupyter Notebooks

**Let's make sure python environment is properly activated and installed**

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

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

**Python Version 3.8 should be selected from the conda environment.**

**Next, lets test the openAI package.** Make sure VPN is off.

```
[ ]: import os
     import openai

     print(os.getenv('OPENAI_KEY'))

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

     print(openai.Model.list())
```

```
[ ]: 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)
```

```
There are several libraries in Python that help with making graphs or
visualizations in Jupyter Notebooks. Here are a some of the most popular:

1. Matplotlib: This is most widely used data visualization library in Python.
Matplotlib is highly customizable and allows you to create a wide variety of
plots like line plots, scatter plots, barplots, histograms, 3D plots and much
more.

2. Seaborn: Built on top of Matplotlib, Seaborn simplifies many visualization
tasks and adds extra functionality. Seaborn's functions can create beautiful and
informative statistical graphics.

3. Plotly: Plotly is an open-source library that allows creation of interactive
plots within the notebook itself. Plot types include 3D charts, statistical
charts, geographical plots, and more.

4. Bokeh: Bokeh translates Python data and function calls into JavaScript and
HTML for stunning and interactive visualizations that can be embedded in Jupyter
notebooks or served as standalone webpage.

5. Altair: Altair provides a simple API for generating complex visualizations in
a declarative manner. It has a straightforward design and it's constructed on
top of the powerful Vega-Lite JavaScript library.

6. ggplot: ggplot is based on the grammar of graphics and allows you to create
plots by combining different components. It's versatile and more user-friendly
than Matplotlib for certain tasks.

The best tool depends on the specific requirements of your project. For
customizability and control, you might want to go with Matplotlib. For
statistical visualizations, Seaborn might be a better option. If you require
interactive visualizations, consider using Plotly or Bokeh. The other tools have
their own strengths as well.
```

[ ]:
```
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 Python code has the following bugs:

1. The `Random` module is imported with a capital 'R', which should be in
lowercase.
2. In the `if` statement, there's an assignment operator (=) instead of a
comparison operator (==).
3. The print statements are not enclosed within quotes.
4. Variables `a` and `b` need to be inside the loop if you want different
questions each time.

Here is 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.")
```
~~~

In this corrected code:

- We import the module as `random`, not `Random`.
- We use double equals (==) for comparison in the `if` statement.
- The strings for printing are enclosed in quotes.
- We convert user input into an integer since input() function returns string by default.
- Variables 'a' and 'b' are moved inside loop to generate new numbers every iteration.