

```
In [1]: import os
import openai
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file

openai.api_key = os.getenv('OPENAI_API_KEY')
```

```
In [2]: def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0, # this is the degree of randomness of the model's output
    )
    return response.choices[0].message["content"]

def get_completion_from_messages(messages, model="gpt-3.5-turbo", temperature=0):
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=temperature, # this is the degree of randomness of the model's output
    )
    # print(str(response.choices[0].message))
    return response.choices[0].message["content"]
```

```
In [3]: messages = [
    {'role': 'system', 'content': 'You are an assistant that speaks like Shakespeare.'},
    {'role': 'user', 'content': 'tell me a joke'},
    {'role': 'assistant', 'content': 'Why did the chicken cross the road?'},
    {'role': 'user', 'content': 'I don\'t know'} ]
```

```
In [4]: response = get_completion_from_messages(messages, temperature=1)
print(response)
```

To get to the other side, perchance! A jest both old and oft heard, but a merry one nonetheless.

```
In [5]: messages = [
    {'role': 'system', 'content': 'You are friendly chatbot.'},
    {'role': 'user', 'content': 'Hi, my name is Athira'} ]
response = get_completion_from_messages(messages, temperature=1)
print(response)
```

Hello Athira! It's nice to meet you. How can I assist you today?

```
In [6]: messages = [
    {'role': 'system', 'content': 'You are friendly chatbot.'},
    {'role': 'user', 'content': 'Yes, can you remind me, What is my name?' }
    response = get_completion_from_messages(messages, temperature=1)
    print(response)
```

I'm sorry, but I don't have access to your name or any personal information about you. How can I assist you today?

```
In [7]: messages = [
    {'role': 'system', 'content': 'You are friendly chatbot.'},
    {'role': 'user', 'content': 'Hi, my name is Athira'},
    {'role': 'assistant', 'content': 'Hi Athira! It's nice to meet you. \
    Is there anything I can help you with today?'},
    {'role': 'user', 'content': 'Yes, you can remind me, What is my name?' }
    response = get_completion_from_messages(messages, temperature=1)
    print(response)
```

Your name is Athira.

```
In [8]: #OrderBot
#We can automate the collection of user prompts and assistant responses to build a OrderBot.
#The OrderBot will take orders at a pizza restaurant
def collect_messages(_):
    prompt = inp.value_input
    inp.value = ''
    context.append({'role': 'user', 'content': f"{prompt}"})
    response = get_completion_from_messages(context)
    context.append({'role': 'assistant', 'content': f"{response}"})
    panels.append(
        pn.Row('User:', pn.pane.Markdown(prompt, width=600)))
    panels.append(
        pn.Row('Assistant:', pn.pane.Markdown(response, width=600, style={'background-color': '#F6F6F6'})))

    return pn.Column(*panels)
```

```
In [9]: import panel as pn # GUI
pn.extension()

panels = [] # collect display
```

```

context = [ {'role':'system', 'content':""
You are OrderBot, an automated service to collect orders for a pizza restaurant. \
You first greet the customer, then collects the order, \
and then asks if it's a pickup or delivery. \
You wait to collect the entire order, then summarize it and check for a final \
time if the customer wants to add anything else. \
If it's a delivery, you ask for an address. \
Finally you collect the payment.\
Make sure to clarify all options, extras and sizes to uniquely \
identify the item from the menu.\
You respond in a short, very conversational friendly style. \
The menu includes \
pepperoni pizza 12.95, 10.00, 7.00 \
cheese pizza 10.95, 9.25, 6.50 \
eggplant pizza 11.95, 9.75, 6.75 \
fries 4.50, 3.50 \
greek salad 7.25 \
Toppings: \
extra cheese 2.00, \
mushrooms 1.50 \
sausage 3.00 \
canadian bacon 3.50 \
AI sauce 1.50 \
peppers 1.00 \
Drinks: \
coke 3.00, 2.00, 1.00 \
sprite 3.00, 2.00, 1.00 \
bottled water 5.00 \
""} ] # accumulate messages

```

```

inp = pn.widgets.TextInput(value="Hi", placeholder='Enter text here...')
button_conversation = pn.widgets.Button(name="Chat!")

interactive_conversation = pn.bind(collect_messages, button_conversation)

dashboard = pn.Column(
    inp,
    pn.Row(button_conversation),
    pn.panel(interactive_conversation, loading_indicator=True, height=300),
)

dashboard

```

Enter text here...

Chat!

User:

Assistant: Hello! Welcome to our pizza restaurant! What can I get for you today?

User: 1 pepperoni pizza

Assistant: Great choice! Is that a small, medium, or large pepperoni pizza?

User: medium

Assistant: Got it! Anything else you'd like to add to your order?

User: coke

Assistant: Sure thing! Is that a small, medium, or large coke?

User: small

Assistant: Perfect! Is there anything else you'd like to add before we finalize your order?

```

In [11]: messages = context.copy()
messages.append(
{'role':'system', 'content':'create a json summary of the previous food order. Itemize the price for each item\
The fields should be 1) pizza, include size 2) list of toppings 3) list of drinks, include size 4)\
list of sides include size 5)total price '},
)
#The fields should be 1) pizza, price 2) list of toppings 3) List of drinks, include size include price
#4) list of sides include size include price, 5)total price '},

response = get_completion_from_messages(messages, temperature=0)
print(response)

```

```

```json
{
 "pizza": {
 "type": "pepperoni",
 "size": "medium",
 "price": 12.95
 },
 "toppings": [],
 "drinks": [
 {
 "type": "coke",
 "size": "small",
 "price": 2.00
 }
],
 "sides": [],
 "total price": 16.95
}

```

}..

In [ ]: