The Chat Format

In this notebook, you will explore how you can utilize the chat format to have extended conversations with chatbots personalized or specialized for specific tasks or behaviors.

Setup

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, methinks!
```

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

```
Hello Isa! 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'm not aware of your name as I'm an AI chatbot and I don't have access to any personal information
unless you have provided it to me before.
```

In [7]:
```
messages =  [
{'role':'system', 'content':'You are friendly chatbot.'},
{'role':'user', 'content':'Hi, my name is Isa'},
{'role':'assistant', 'content': "Hi Isa! 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 Isa! It's a pleasure to assist you, Isa. Do you need help with anything else?
```

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.

In [8]:
```
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: I am planning to order some stuff but I don't remember the menu. Can you list the menu for me?

Assistant: Sure, here's our menu:

- Pepperoni pizza: 12.95 (large), 10.00 (medium), 7.00 (small)
- Cheese pizza: 10.95 (large), 9.25 (medium), 6.50 (small)
- Eggplant pizza: 11.95 (large), 9.75 (medium), 6.75 (small)
- Fries: 4.50 (large), 3.50 (small)
- 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 (large), 2.00 (medium), 1.00 (small), sprite 3.00 (large), 2.00 (medium), 1.00 (small), bottled water 5.00

In [10]:
```
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 si
)
 #The fields should be 1) pizza, price 2) list of toppings 3) list of drinks, include size include price  4) lis

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

```
Sure, here's a JSON summary of your order:

```
{
  "pizza": {
    "size": "large",
    "type": "half-and-half",
    "toppings": ["cheese", "eggplant", "onions"],
    "price": 12.95
  },
  "drinks": [
    {
      "type": "coke",
      "size": "large",
      "price": 3.00
    }
  ],
  "sides": [],
  "total_price": 14.95
}
```

Please note that the summary only includes the pizza and the drink you ordered, as you did not order any sides.
```

Try experimenting on your own!

You can modify the menu or instructions to create your own orderbot!

In [ ]:
```
```