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 [21]:

```
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 [24]:

```
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 outp
    )
    print(str(response.choices[0].message))
    return response.choices[0].message["content"]
```

In [25]:

```
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 [26]:

```
response = get_completion_from_messages(messages, temperature=1)
print(response)
```

```
{
  "content": "To get to the other side! 'Tis an oldie but a goodie.",
  "role": "assistant"
}
To get to the other side! 'Tis an oldie but a goodie.
```

In [27]:

```
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)
```

```
{
  "content": "Hello Isa! How can I assist you today?",
  "role": "assistant"
}
Hello Isa! How can I assist you today?
```

In [28]:

```
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)
```

```
{
  "content": "I'm sorry, but I'm not able to know your name as I am just a
n AI language model. However, you can tell me your name, and I will make s
ure to refer to you by your name from now on.",
  "role": "assistant"
}
I'm sorry, but I'm not able to know your name as I am just an AI language
model. However, you can tell me your name, and I will make sure to refer t
o you by your name from now on.
```

In [29]:

```
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)
```

```
{
  "content": "Of course! Your name is Isa.",
  "role": "assistant"
}
Of course! Your name is Isa.
```

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 [30]:

```
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-c

    return pn.Column(*panels)
```

In [31]:

```
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
```

```
{
  "content": "Hello! Welcome to our pizza restaurant. What can I get for y
ou today?",
  "role": "assistant"
}
```

yes

Chat!

User:

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

User: I want a eggplant pizza, sausage topping and no cheese, and a medium sprite. I will p
thank you.

Assistant: Great! So that's an eggplant pizza with sausage topping and no cheese, and a me
pickup. Is there anything else you'd like to add to your order?

User: no ,thx

Assistant: Alright, just to confirm, your order is an eggplant pizza with sausage topping and r
a medium sprite for pickup. Is that correct?

User: yes

Assistant: Perfect! Your order for an eggplant pizza with sausage topping and no cheese, an
sprite for pickup has been placed. Your total comes out to $15.45. Please come to

In [32]:

```
messages =  context.copy()
messages.append(
{'role':'system', 'content':'create a json summary of the previous food order. Itemize
 The fields should be 1) pizza, include size 2) list of toppings 3) list of drinks, inc
)
 #The fields should be 1) pizza, price 2) list of toppings 3) list of drinks, include s

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

```
{
  "content": "Here's a JSON summary of your order:\n\n```\n{\n  \"pizza\":
{\n    \"type\": \"eggplant\",\n    \"size\": \"medium\",\n    \"toppings
\": [\n      \"sausage\"\n    ],\n    \"cheese\": false,\n    \"price\": 1
1.95\n  },\n  \"drinks\": [\n    {\n      \"type\": \"sprite\",\n      \"s
ize\": \"medium\",\n      \"price\": 2.00\n    }\n  ],\n  \"sides\": [],\n
\"total_price\": 13.95\n}\n```",
  "role": "assistant"
}
```

Here's a JSON summary of your order:

```
{
  "pizza": {
    "type": "eggplant",
    "size": "medium",
    "toppings": [
      "sausage"
    ],
    "cheese": false,
    "price": 11.95
  },
  "drinks": [
    {
      "type": "sprite",
      "size": "medium",
      "price": 2.00
    }
  ],
  "sides": [],
  "total_price": 13.95
}
```

Try experimenting on your own!

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

In [ ]: