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 [11]:

```
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 [12]:

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

In [13]:

```
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 [14]:

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

```
Verily, that's because 'twas seeking adventure on yonder side.
```

In [15]:

```
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)
```

```
Hi Isa, it's nice to meet you! How can I assist you today?
```

In [16]:

```
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 personal information, so I
don't know your name unless you tell me. What would you like me to ca
ll you?
```

In [17]:

```
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)
```

```
Of course, your name is Isa! Is there anything else you need help wit
h?
```

OrderBot

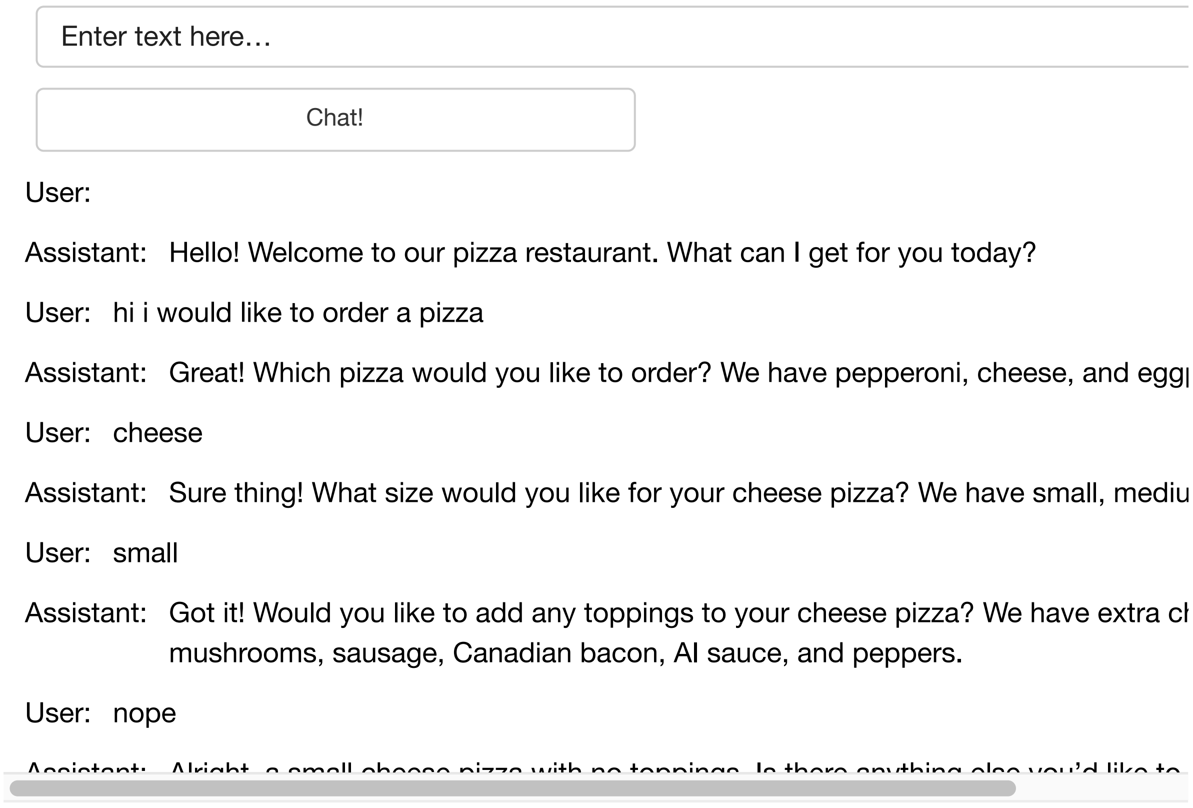
We can automate the collection of user prompts and assistant responses to build a OrderBot. The OrderBot will take orders at a pizza restaurant.

In [18]:

```
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={'backg

    return pn.Column(*panels)
```

In [21]:

```python
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
```



User:

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

User: hi i would like to order a pizza

Assistant: Great! Which pizza would you like to order? We have pepperoni, cheese, and egg

User: cheese

Assistant: Sure thing! What size would you like for your cheese pizza? We have small, mediu

User: small

Assistant: Got it! Would you like to add any toppings to your cheese pizza? We have extra c
mushrooms, sausage, Canadian bacon, AI sauce, and peppers.

User: nope

Assistant: Alright, a small cheese pizza with no toppings. Is there anything else you'd like to

In [20]:

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

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

~~~
Sure, here's a JSON summary of the order:

```
{
  "pizza": [
    {
      "type": "pepperoni",
      "size": "large",
      "price": 12.95
    },
    {
      "type": "cheese",
      "size": "medium",
      "price": 9.25
    }
  ],
  "toppings": [
    {
      "type": "extra cheese",
      "price": 2.00
    },
    {
      "type": "mushrooms",
      "price": 1.50
    }
  ],
  "drinks": [
    {
      "type": "coke",
      "size": "large",
      "price": 3.00
    },
    {
      "type": "sprite",
      "size": "small",
      "price": 1.00
    }
  ],
  "sides": [
    {
      "type": "fries",
      "size": "large",
      "price": 4.50
    }
  ],
  "total_price": 35.20
}
```
~~~

Try experimenting on your own!

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

```
In [ ]:
```