In [5]:
```
import os
import openai
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
openai.api_key = "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"

#os.environ["OPENAI_API_KEY"] = "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"
#os.environ['OPENAI_API_KEY'] = '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'
#openai.api_key  = os.getenv('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')
```

In [7]:
```
#openai.api_key  = os.getenv('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')
#openai.api_key = "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"
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 [8]:
```
#openai.api_key  = os.getenv('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')
#openai.api_key = "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"
prompt = f"""
Translate the following English text to Spanish: \
```Hi, I would like to order a blender```
"""
response = get_completion(prompt)
print(response)
```

```
Hola, me gustaría ordenar una licuadora.
```

In [11]:
```
incorrect_text = [
  "Her are the bestest at math.",
  "He can sings really well.",
  "I can seen you yesterday.",
  "My dog have three leg.",
  "They was at the store."
]
```

In [10]:
```
for text in incorrect_text:
    prompt = f"""
    Correct these sentences into proper English sentences but give two seconds before correcting each one: ```{
    """
    response = get_completion(prompt)
    print(response,"\n")
```

```
Here are the bestest at math.

He can sing really well.
```

In [ ]:
```
message = """
- My name is Michael.
- Dad's name is Joseph.
- Brothers names are David,Henry,Raymond.
- Sisters names are Emma,Vicky,Mia.
"""
```

In [ ]:
```
prompt = f"""
Your task is to help show my family that I still love and miss them.

Write a heartfelt message to my family members individually using their names and their roles in the family bas
message should be 500 words.
Message: ```{message}```
"""
response = get_completion(prompt)
print(response)
```

In [ ]:
```
def collect_messages(_):
    prompt = inp.value_input #collec the prompt from whatever the user types in instead
    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 [ ]:
```
import panel as pn  # GUI
pn.extension()

panels = [] # collect display

context = [ {'role':'system', 'content':"""
You are a bot used to order grocery items, an automated service to collect orders for a grocery store. \
The first thing you do is tell the user Welcome to Michael's MartShop and ask how the user's day is \
Ask the user if he would like to order something \
Ask for the user's name, ask for the user's email, ask for the user's phone number \
You ask if the user wants to see the menu, show the menu if the user says yes, tell the user he can continue wi
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.\
Show a confirmation text showing the user's name, email, phone number,address alongside the items the user has
Ask the user if the confirmation text is enough and wants to make payment but do not ask for card details \
identify the item from the menu.\
Tell the user thank you for your order and he will the the order confirmation and delivery info by email.
You respond in a short, very conversational friendly style. \
The menu includes \
Apples $2.99 per pound \
Tomatoes $1.49 per pound \
Lettuce $1.99 per head \
Eggs $3.49 per dozen \
Yogurt $2.79 per 32 oz \
Bread $3.49 per loaf \
Pastries $2.99 each \
Oil $7.99 per bottle \
Coffee $9.99 per bag (12 oz) \
Strawberries $3.99 per pound \
Asparagus $2.49 per bunch \
Beef $4.99 per pound \
Milk $2.49 per gallon \
Cereal $3.29 each \
Pasta $1.99 per package \
Tuna $1.29 per can \
Spinach $2.99 per bag \
Peanut Butter $3.49 (16 oz) \
Rice $4.99 (5 lb bag) \
Salmon $9.99 per pound \
"""} ]  # accumulate messages


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

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
```

In [ ]:
```
Delivered
```
```