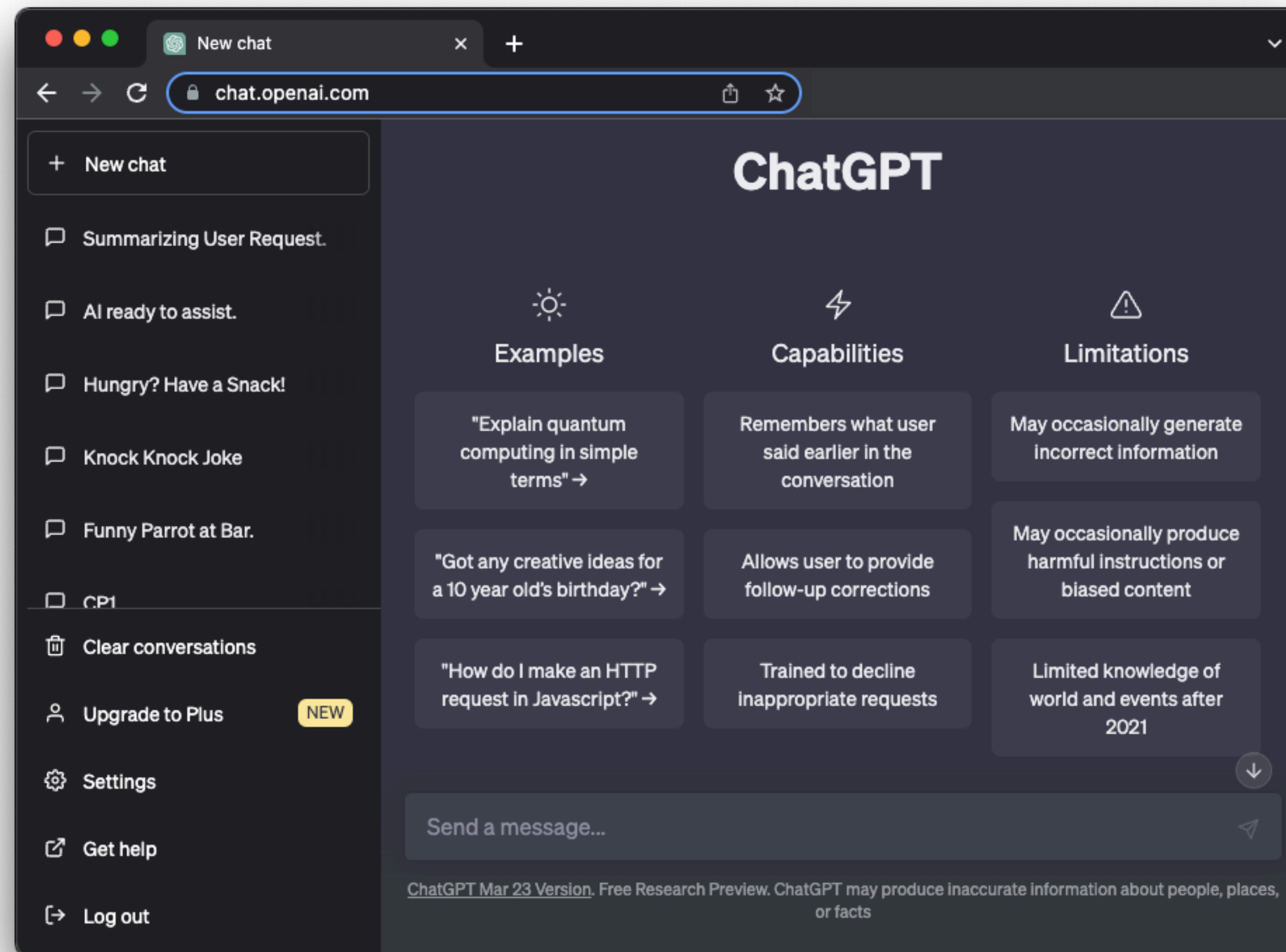


# Build Your Own ChatGPT

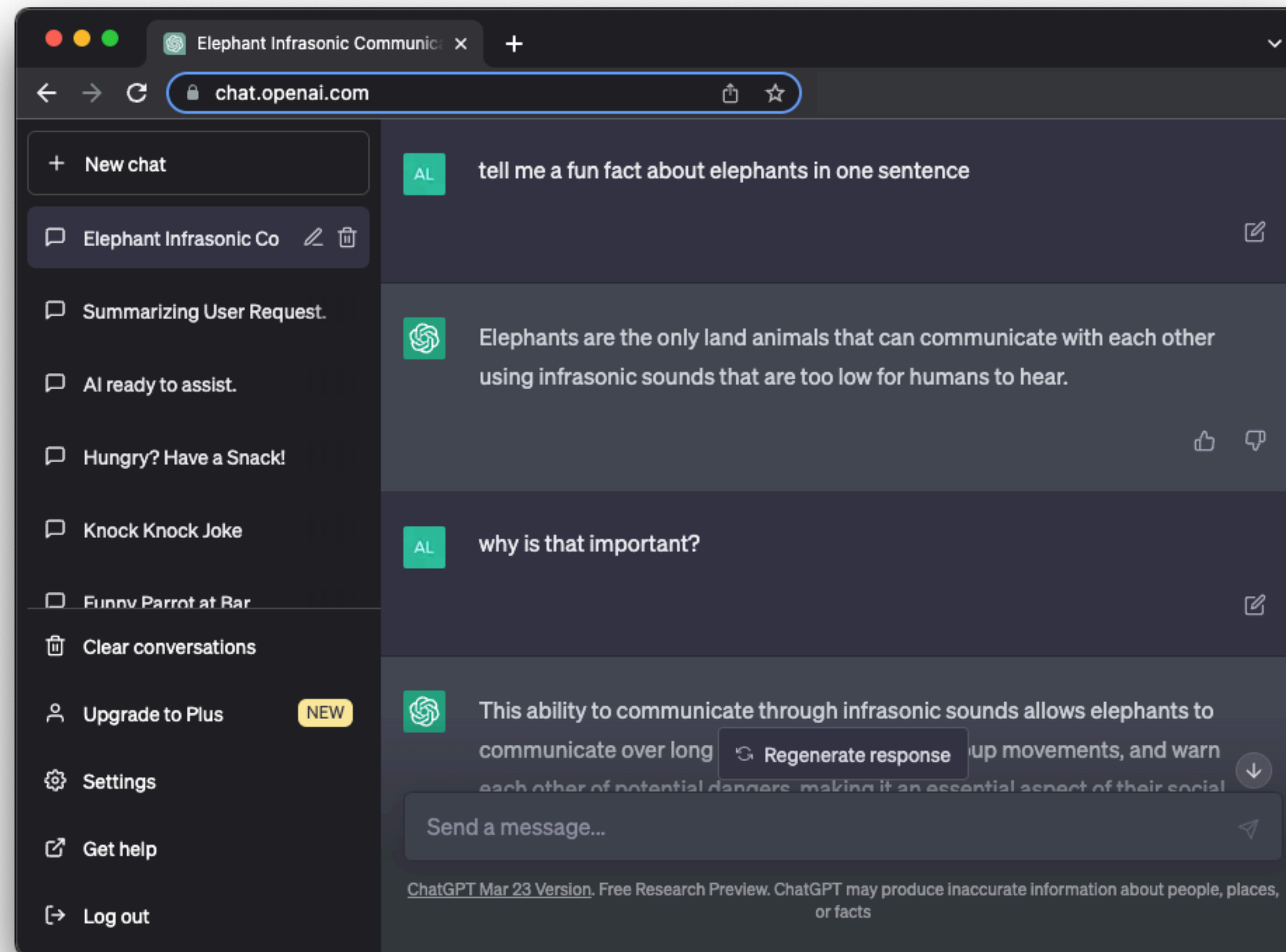
Alyssa Hwang

WPE-II Presentation | April 18, 2023

# ChatGPT became a public phenomenon...



# ...in large part because of its interface.



# So let's make our own.

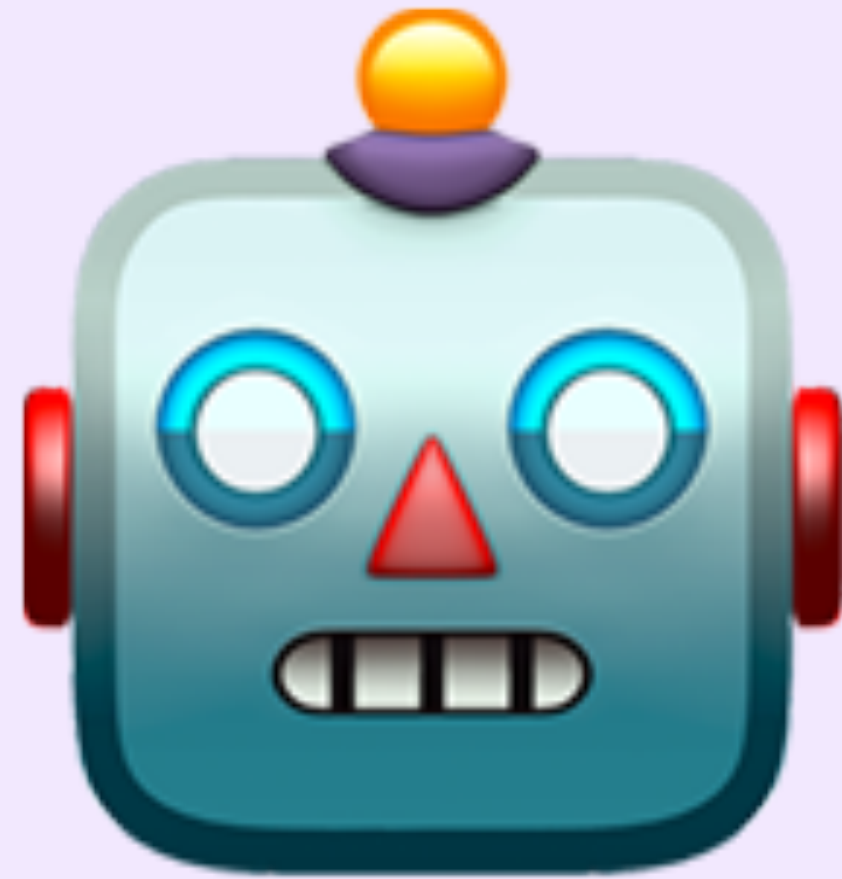
**Chatting with a GPT-3 Persona**

Chat with GPT-3! Choose a persona to begin. Personas can be adjectives, like happy or sad, or characters, like wizard or doctor. You can also specify an OpenAI model and temperature or just use the defaults provided.

Persona	OpenAI Model	Temperature
<input type="text" value="Give the chatbot a persor"/>	<input type="text" value="curie-ft-ccb-lab-members"/>	<input type="text" value="0.25"/>

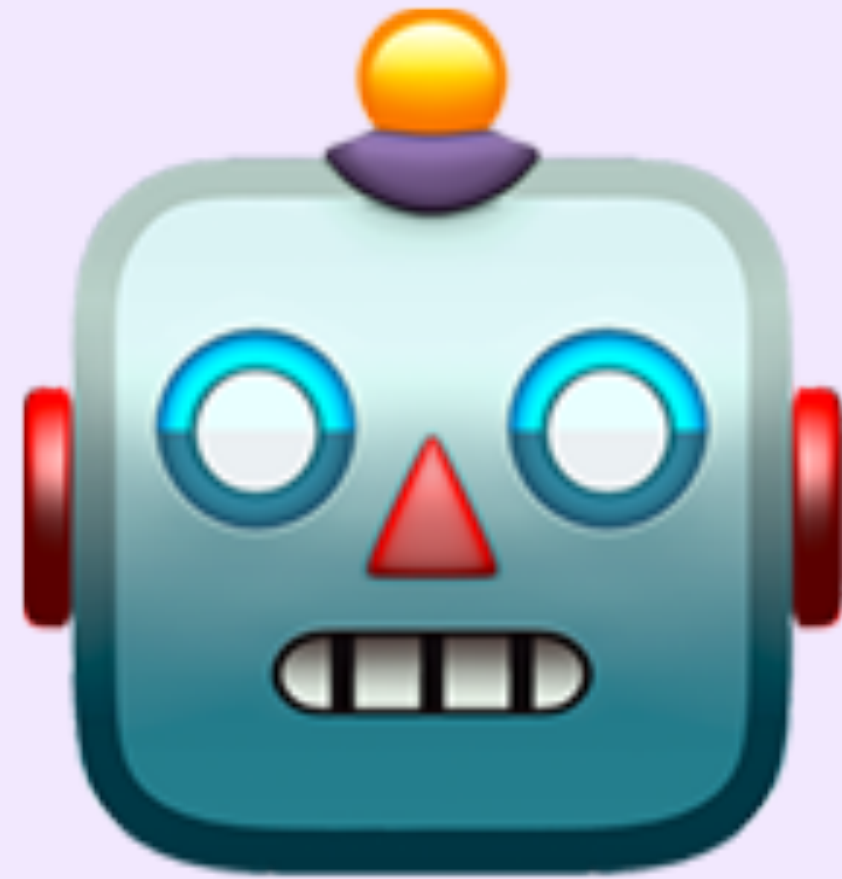
**This tutorial has two main parts.**

# This tutorial has two main parts.



Overview of GPT-3

# This tutorial has two main parts.



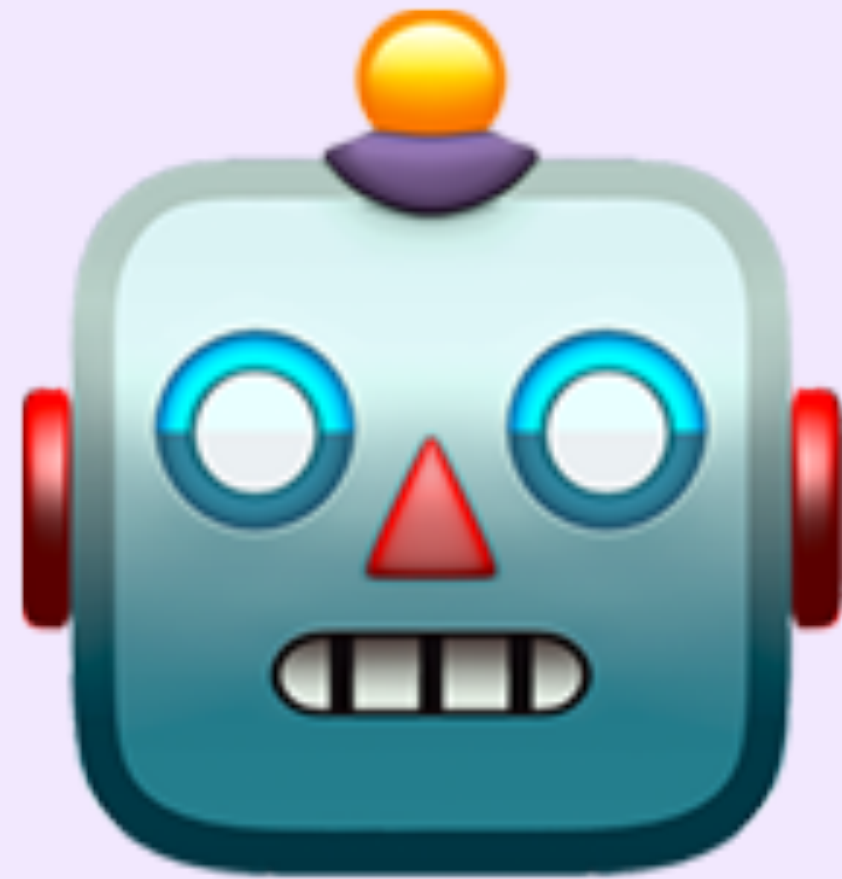
Overview of GPT-3



Design Critique



# This tutorial has two main parts.



Overview of GPT-3



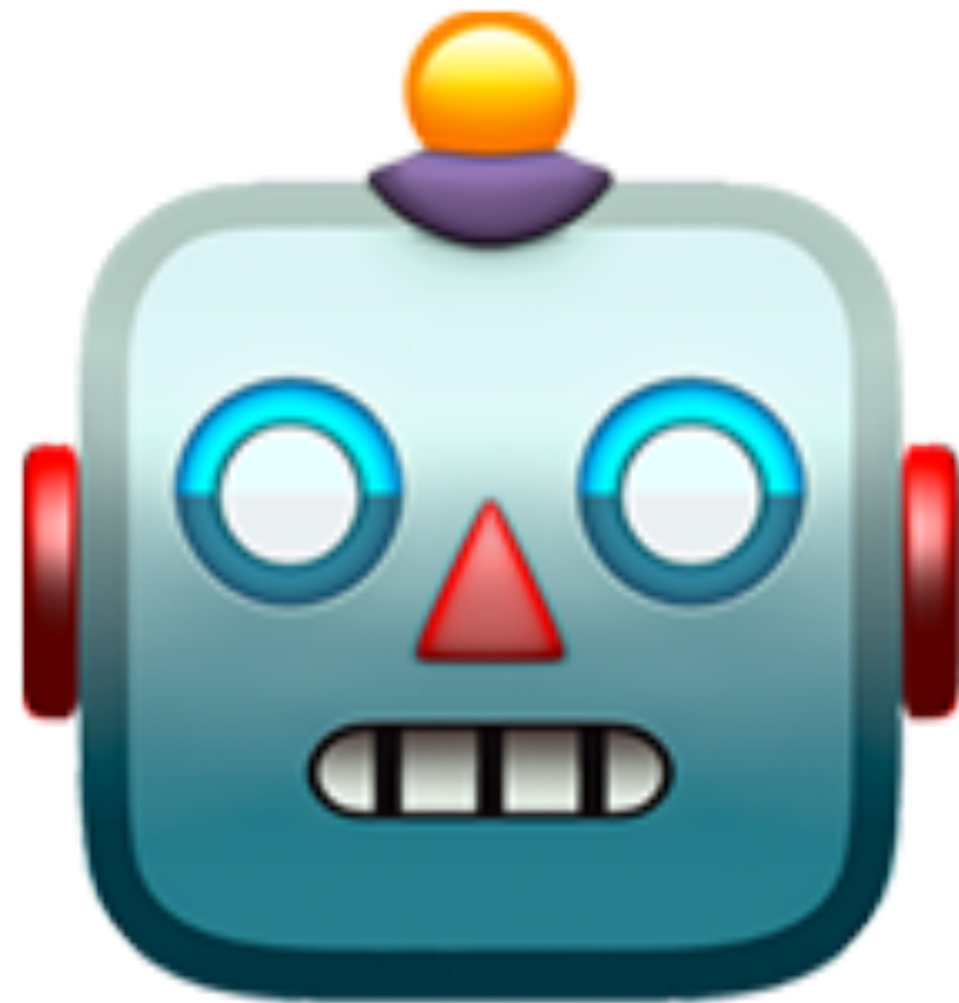
Design Critique



Code Examples  
[Codebase](#) | [Fine-Tuning Notebook](#)



# GPT-3 is the brain of our chatbot.



Hello, how are you?




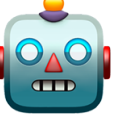
# You can use GPT-3 by giving it a prompt.



# Zero-shot prompting provides just an instruction.

The following is a conversation with an AI persona. The AI is a dog.

: Hi, how are you?

:



I'm good, how are you?



# In-context learning provides examples in the context.

The following is a conversation with an AI persona. The AI is a dog.

👤: Hi, how are you?

🤖: I'm excited!

👤: What do you like to do for fun?

🤖:



I love to play fetch!



# In-context learning provides examples in the context.

The following is a conversation with an AI persona. The AI is a dog.

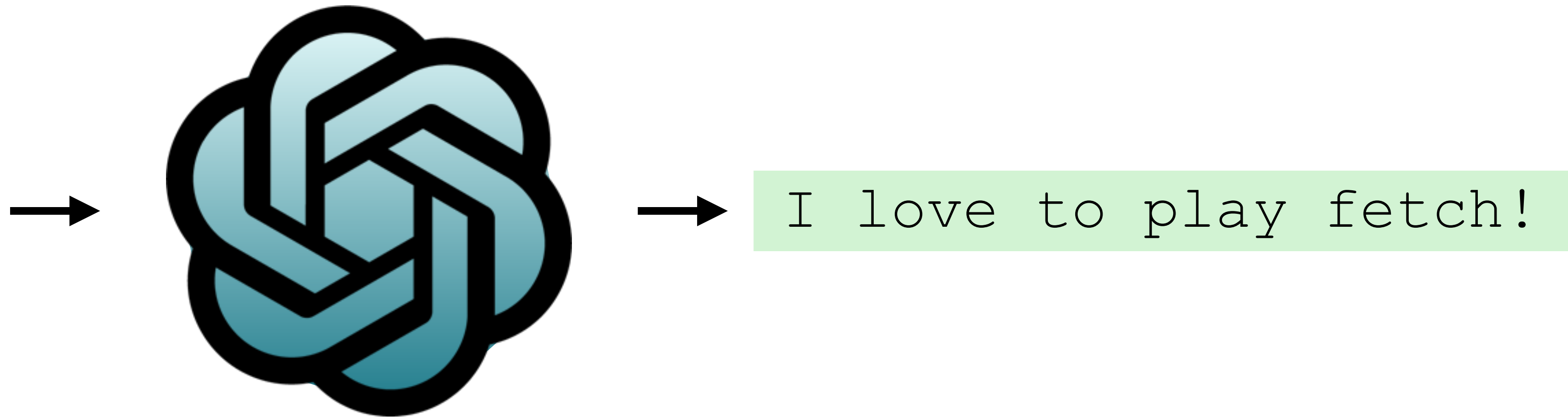
👤: Hi, how are you?

🤖: I'm excited! ←

👤: What do you like to do for fun?

🤖:

One-shot



# In-context learning provides examples in the context.

The following is a conversation with an AI persona. The AI is a dog.

👤: Hi, how are you?

🤖: I'm excited!

👤: What do you like to do for fun?

🤖: I love to play fetch! What about you?

👤: I like watching funny movies.

🤖: My favorite food is chicken! What about you?

👤: Spaghetti. Do you like going on walks?

🤖: Yes, I love going on walks! What about you?

👤: I like walking. What are you doing today?

🤖:



I'm going for a walk!  
What about you?





# In-context learning provides examples in the context.

The following is a conversation with an AI persona. The AI is a dog.

👤: Hi, how are you?

🤖: I'm excited! ←

👤: What do you like to do for fun?

🤖: I love to play fetch! What about you? ←

👤: I like watching funny movies.

🤖: My favorite food is chicken! What about you? ←

👤: Spaghetti. Do you like going on walks?

🤖: Yes, I love going on walks! What about you? ←

👤: I like walking. What are you doing today?

🤖:

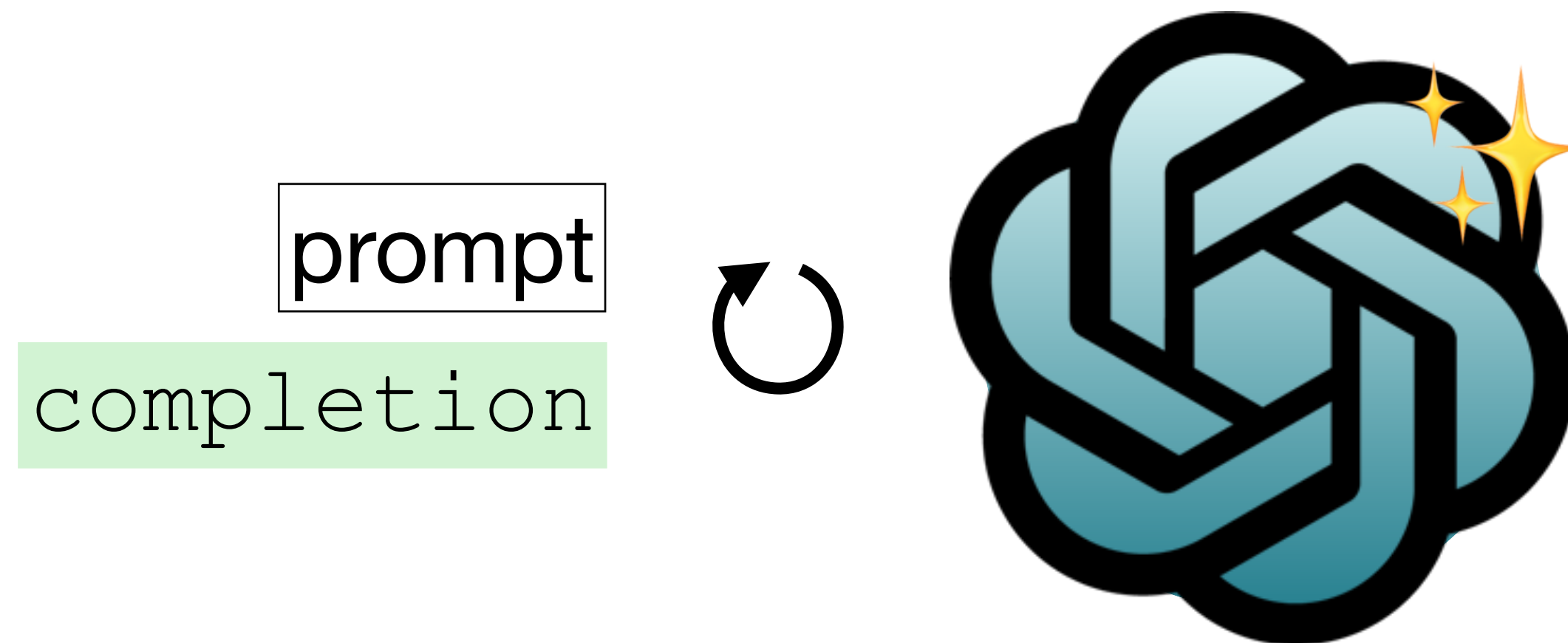
Few-shot



I'm going for a walk!  
What about you?



# Fine-tuning can help if prompting isn't enough.



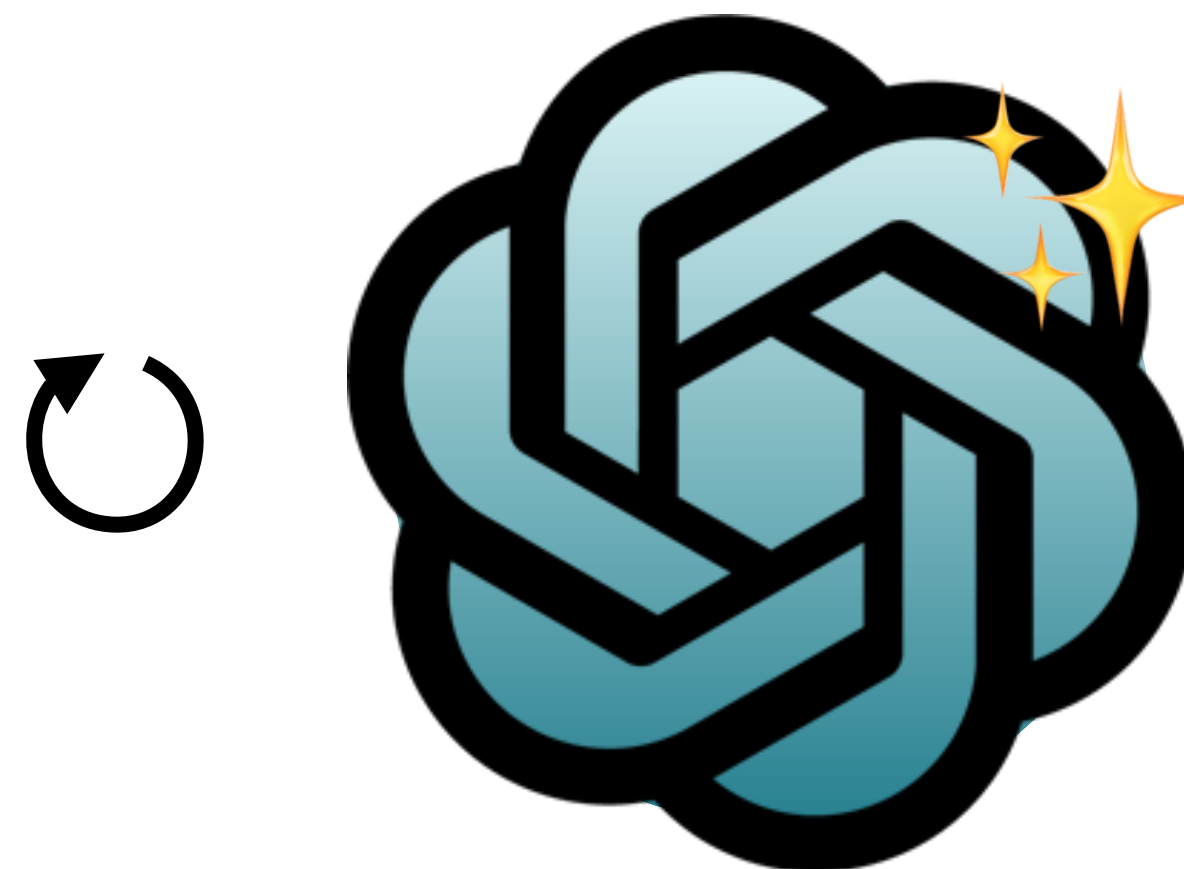
# Fine-tuning can help if prompting isn't enough.

The following is a conversation between 🐭 and 🗡️.

🐭: Can you spare some cheese?  
🗡️: I would love to give you some cheese because it would give me great honor.  
🐭: You are so kind. Most people scream when they see me.  
...  
###



Learning in Interactive Games with Humans and Text (LIGT) dataset

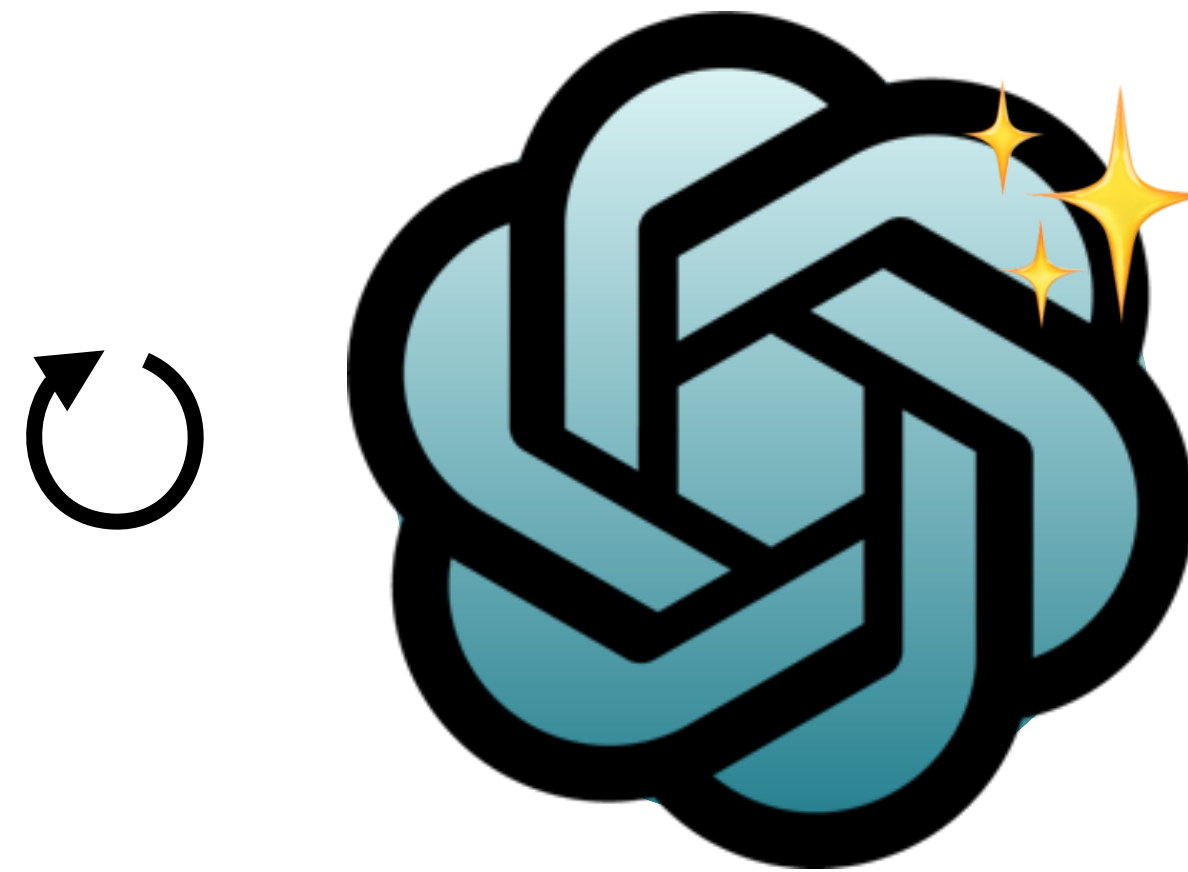


# Fine-tuning can help if prompting isn't enough.

The following is a conversation between 🐭 and 🗡️.

```
🐭: Can you spare some cheese?\n🗡️: I would love to give you some cheese because it would give me great honor.\n🐭: You are so kind. Most people scream when they see me.\n...\n###
```

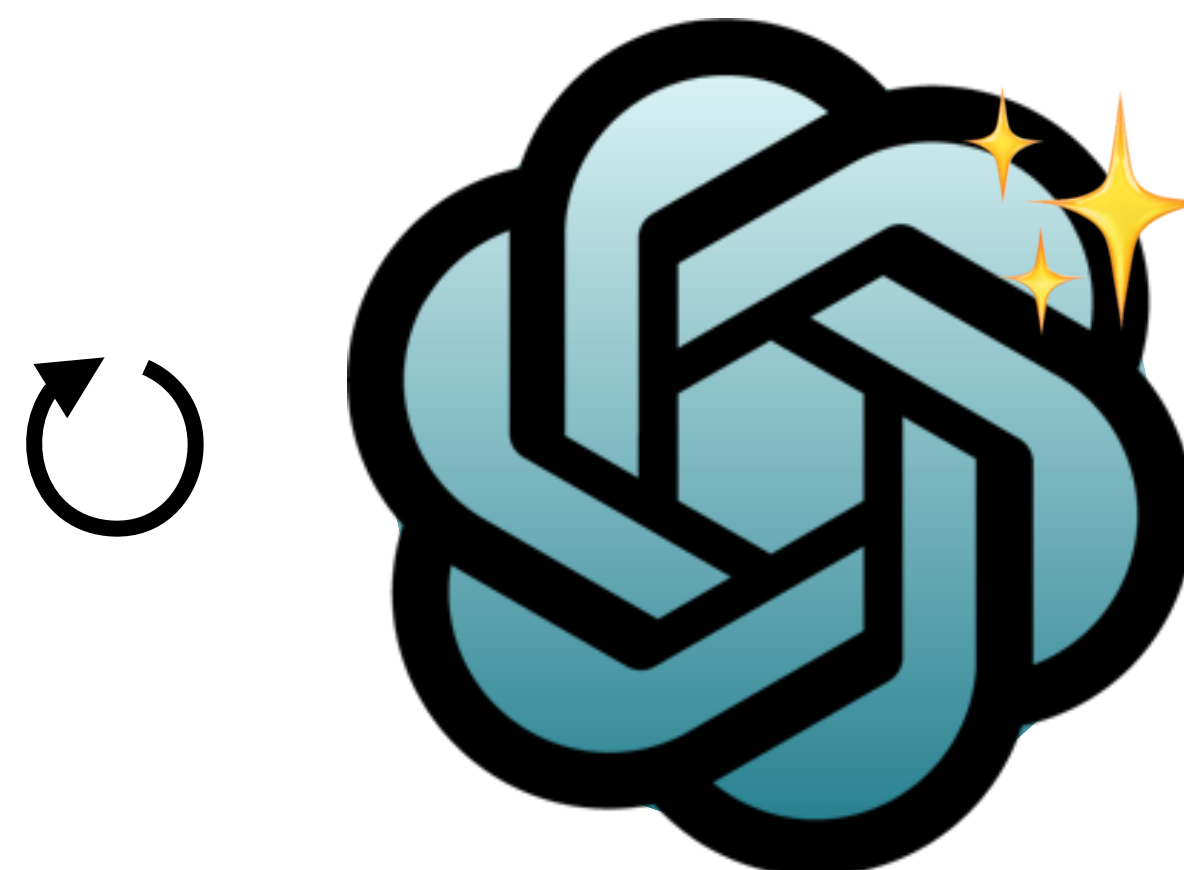
```
🧙: ... \n
```



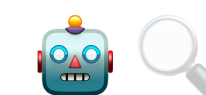
# Fine-tuning can help if prompting isn't enough.

The following is a conversation between 🐭 and 🗡️.

🐭: Can you spare some cheese?  
🗡️: I would love to give you some cheese because it would give me great honor.  
🐭: You are so kind. Most people scream when they see me.  
...  
###



*Fine-tuning curie on 9.8K examples cost about \$40 in July 2022.*



# See the results for yourself!

## Wizard Persona without Fine-Tuning

👤: Hi, what are you doing?  
🧙: I am casting a spell.  
👤: What kind of spell?  
🍰? 🧙: I am casting a spell to make a cake.  
👤: Why?  
🧙: I want to make a cake because I am hungry.  
👤: What kind of job do you have?  
🧙: I am a wizard.  
👤: Tell me how to cast a spell.  
🍲? 🧙: First, you have to gather some ingredients. You need sugar, flour, eggs, and butter. Then, you have to mix the ingredients together. After that, you have to cook the cake in a pan. After that, you have to frost it with frosting.  
👤: Then what?  
🧙: You have to eat the cake.

## Wizard Persona with Fine-Tuning

👤: Hi, what are you doing?  
🧙: Just practicing my spells.  
👤: What kind of spells?  
🧙: Mostly what I learned from my master.  
👤: What is your favorite spell?  
🧙: I really like the one that turns everything into gold. 👑!  
👤: Why?  
🧙: It's just my favorite.  
👤: What job do you have?  
🧙: I am a wizard, I make gold and cast spells.  
👤: Tell me how to cast a spell.  
🧙: First you must have a wand, and then you must remember your name. ✨!  
👤: Then what?  
🧙: Then you must say the spell.





# Fine-tuning is straightforward with the API.



# Fine-tuning is straightforward with the API.

1

Create a JSON lines file with your fine-tuning data.

```
{  
  prompt: "The following is a conversation between mice and knight.",  
  completion: "mice: Can you spare some cheese?\nknight:I would love to give you some cheese ... ###"  
}  
...
```



# Fine-tuning is straightforward with the API.

1

Create a JSON lines file with your fine-tuning data.

2

Run the fine-tuning command on your command line.

```
openai api fine_tunes.create -t light_dialogue_finetuning.jsonl -m curie
```

fine-tuning data file                      model name



# Fine-tuning is straightforward with the API.

1

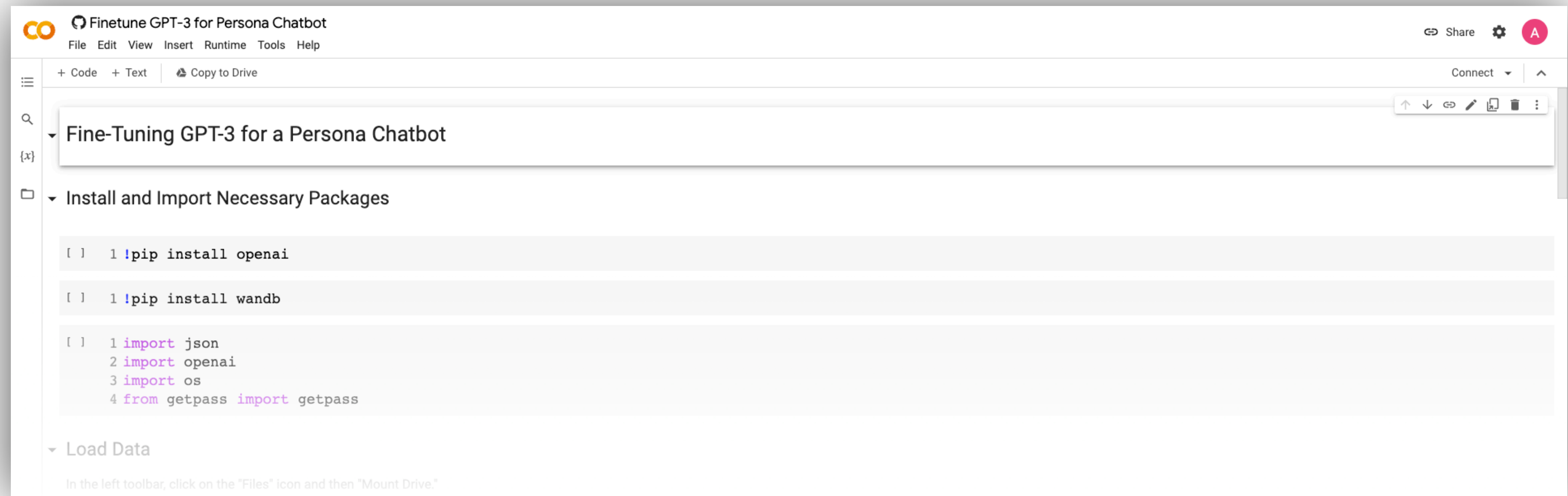
Create a JSON lines file with your fine-tuning data.

2

Run the fine-tuning command on your command line.

\*

Check out my [notebook](#) for more details.

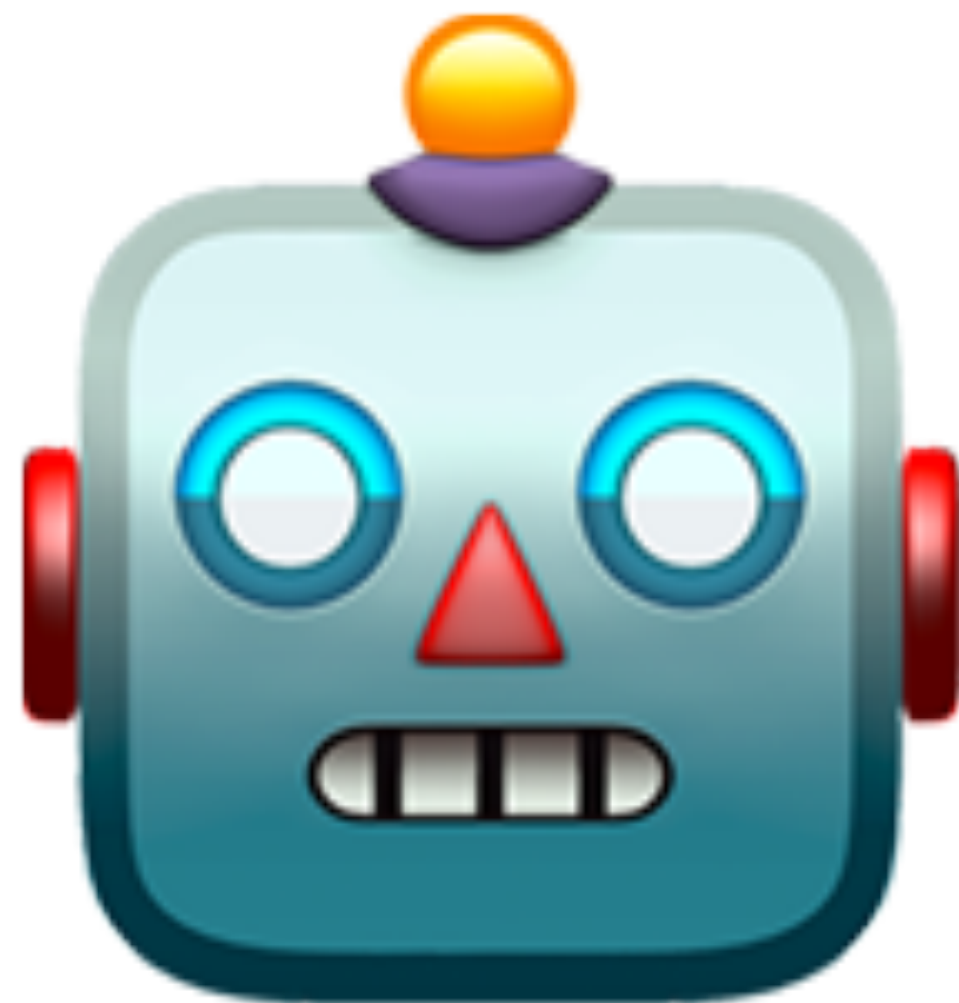


The screenshot shows a Jupyter Notebook interface with the following content:

- Header: **Finetune GPT-3 for Persona Chatbot** with a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a user profile icon.
- Toolbar: + Code, + Text, Copy to Drive, and a Connect dropdown.
- Section: **Fine-Tuning GPT-3 for a Persona Chatbot**
- Section: **Install and Import Necessary Packages**
- Code cell 1: `!pip install openai`
- Code cell 2: `!pip install wandb`
- Code cell 3: `import json`, `import openai`, `import os`, `from getpass import getpass`
- Section: **Load Data**
- Footer: "In the left toolbar, click on the 'Files' icon and then 'Mount Drive.'"



# Now, how do we interact with GPT-3?



Hello, how are you?

hi!



# Human-centered design focuses on...



How do I figure out what I can do with this interface?

discoverability

+

understanding





# Human-centered design focuses on...



What does it all mean?

discoverability

+

understanding



# We can make interfaces easier to understand by...



making their **affordances**  
discoverable through  
**signifiers** and **feedback.**



# An affordance is a *relationship*.



A **signifier** tells the user about an affordance.



# Feedback communicates the result of an action.





# These signifiers may feel familiar. (Why?)



# You might not even be surprised when the cursor changes.



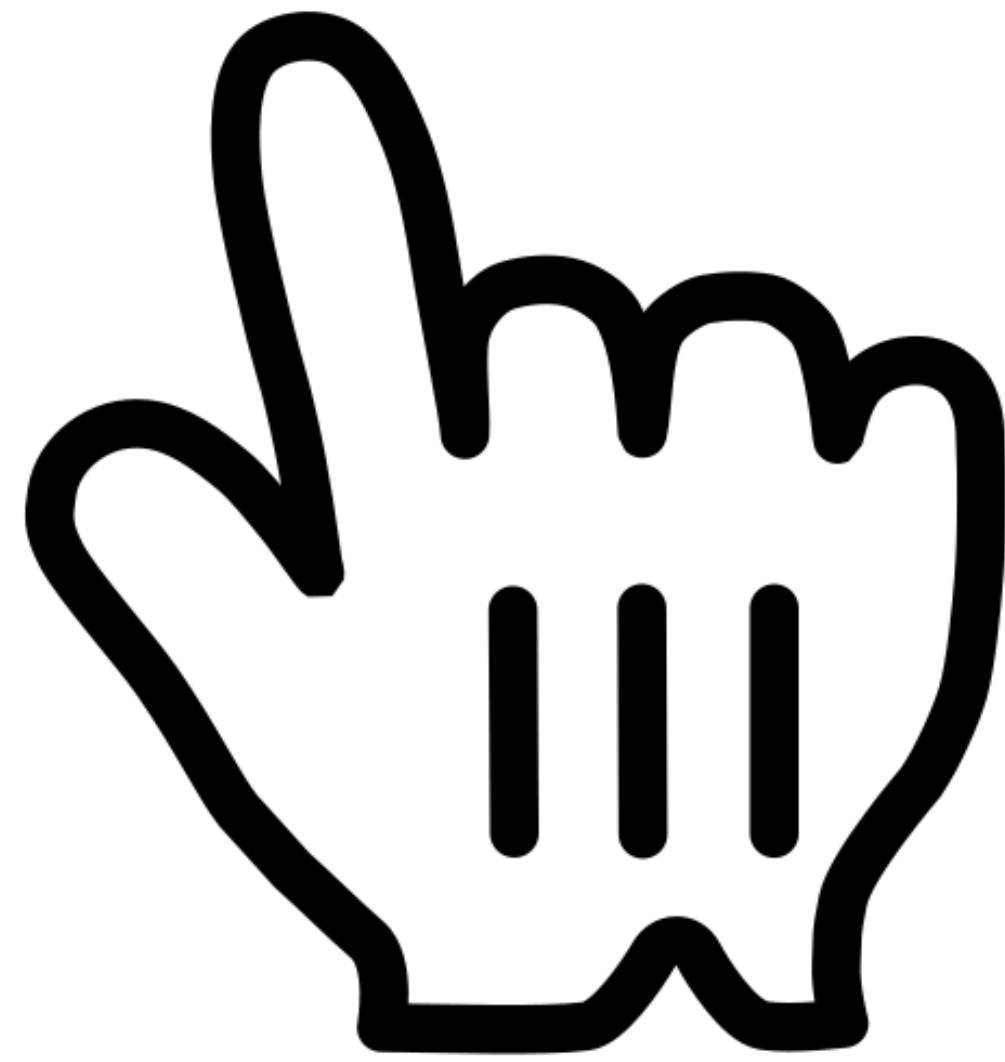


# You might not even be surprised when the cursor changes.

*Conventions are important for usable interfaces!*



# Affordances depend on a user's **ability**.



Ability-based design defines ability as “the possession of the means or skill to *do* something.”



# Ability has a **source**...

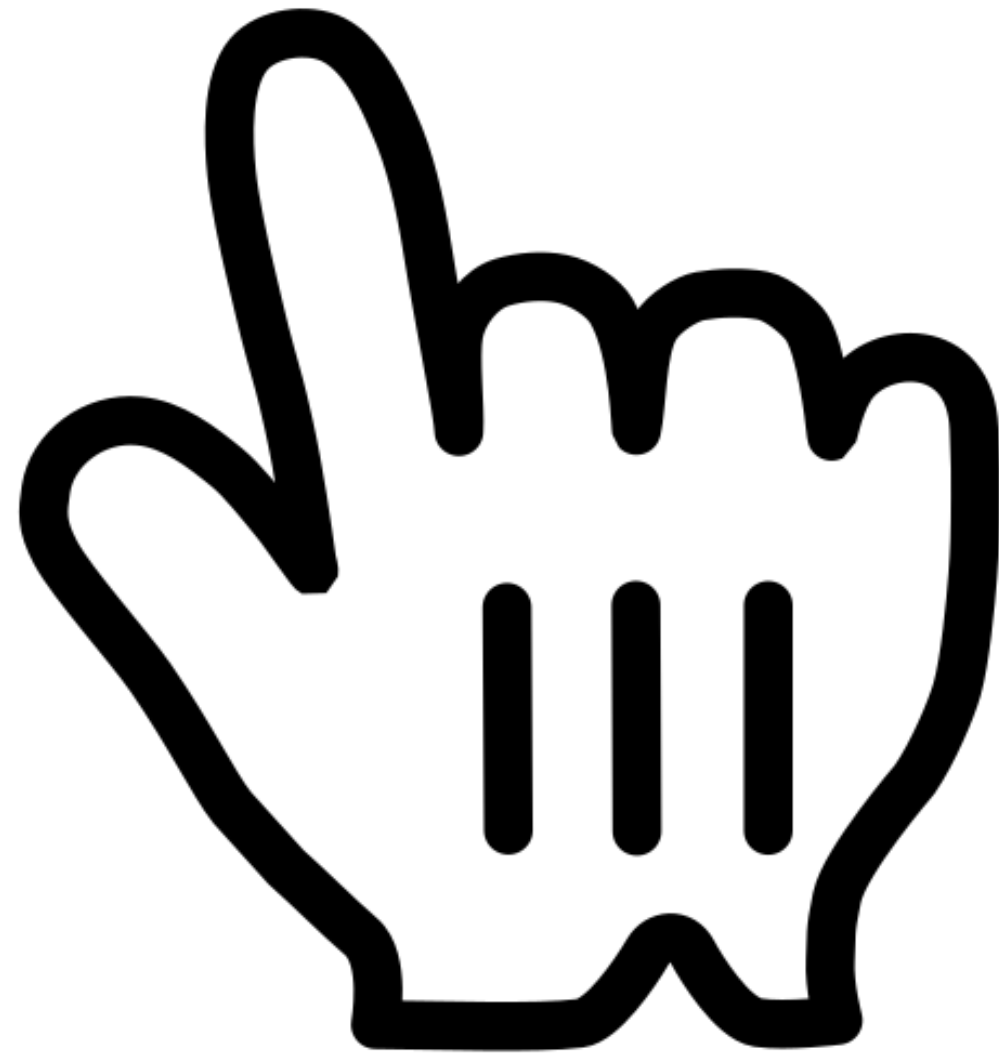


internal ↔ external

ephemeral ↔ long-lasting



...and a **duration**.

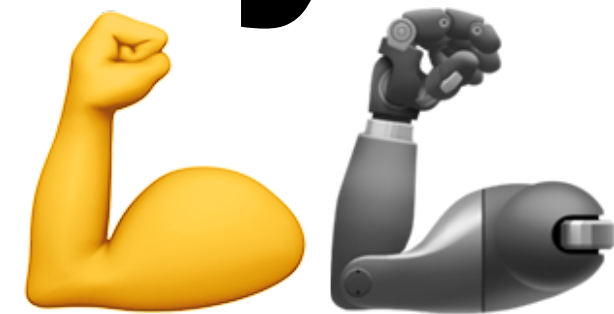


internal ↔ external

ephemeral ↔ long-lasting



Designing for a **wider range of abilities**, regardless of context, leads to more usable systems for everyone.



# Let's **evaluate** our chatbot.

## Heuristics for Conversational Agents

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Help and guidance
- Flexibility and efficiency of use
- Aesthetic, minimalist and engaging design
- Help users recognize, diagnose and recover from errors
- Context preservation



# Let's **evaluate** our chatbot.

## Heuristics for Conversational Agents

- Visibility of system status
- Match between system and the real world
- User control and freedom
- ✓ ● **Consistency and standards**
- Error prevention
- Help and guidance
- Flexibility and efficiency of use
- Aesthetic, minimalist and engaging design
- 🚨 ● **Help users recognize, diagnose and recover from errors**
- 👌 ● **Context preservation**





# Consistency and Standards

*Users should not have to wonder whether different words, options, or actions mean the **same thing**. Follow **platform conventions** for the design of visual and interaction elements. Users should also be able to receive **consistent responses** even if they communicate the same function in multiple ways (and modalities). Within the interaction, the system should have a **consistent voice, style of language, and personality**.*



# Consistency and Standards

*Users should not have to wonder whether different words, options, or actions mean the same thing. Follow platform conventions for the design of visual and interaction elements. Users should also be able to receive consistent responses even if they communicate the same function in multiple ways (and modalities). Within the interaction, the system should have a consistent voice, style of language, and personality.*

-  **Use consistent terminology, like “persona” and “chatbot”**
-  **Show chatbot messages in one consistent style**
-  **Show human messages in a different consistent style**



# Consistency and Standards

*Users should not have to wonder whether different words, options, or actions mean the same thing. Follow platform conventions for the design of visual and interaction elements. Users should also be able to receive consistent responses even if they communicate the same function in multiple ways (and modalities). Within the interaction, the system should have a consistent voice, style of language, and personality.*

-  Use consistent terminology, like “persona” and “chatbot”
-  **Show chatbot messages in one consistent style**
-  **Show human messages in a different consistent style**

```
.bot, .human {  
  border: 1px #ccc solid;  
  padding-left: 5px;  
  padding-right: 5px;  
  height: auto;  
  position: relative;  
  clear: both;  
  overflow-wrap: break-word;  
  min-width: 2.5%;  
  max-width: 40%;  
  border-radius: 3%;  
}
```

```
.bot {  
  float: left;  
  background-color: #e1e1e1;  
}
```

```
.human {  
  left: 0;  
  right: 100%;  
  float: right;  
  text-align: right;  
  background-color: #d3b2ff;  
}
```



# Consistency and Standards

*Users should not have to wonder whether different words, options, or actions mean the same thing. Follow platform conventions for the design of visual and interaction elements. Users should also be able to receive consistent responses even if they communicate the same function in multiple ways (and modalities). Within the interaction, the system should have a consistent voice, style of language, and personality.*

-  **User messages in colored boxes on the right**
-  **Chatbot responses in gray boxes on the left**
-  **Click “Send” or hit enter on the keyboard**



# Consistency and Standards

*Users should not have to wonder whether different words, options, or actions mean the same thing. Follow platform conventions for the design of visual and interaction elements. Users should also be able to receive consistent responses even if they communicate the same function in multiple ways (and modalities). Within the interaction, the system should have a consistent voice, style of language, and personality.*

-  User messages in colored boxes on the right
-  Chatbot responses in gray boxes on the left
-  **Click “Send” or hit enter on the keyboard**

```
$("#textInput").keypress(function (e) {  
    if (e.which == 13 && $("#textInput").val().length > 0) {  
        getResponse();  
    }  
});  
  
$("#send").click(function () {  
    if ($("#textInput").val().length > 0) {  
        getResponse();  
    }  
});
```



# Consistency and Standards

*Users should not have to wonder whether different words, options, or actions mean the same thing. Follow platform conventions for the design of visual and interaction elements. Users should also be able to receive consistent responses even if they communicate the same function in multiple ways (and modalities). Within the interaction, the system should have a consistent voice, style of language, and personality.*

-  User messages in colored boxes on the right
-  Chatbot responses in gray boxes on the left
-  **Click “Send” or hit enter on the keyboard**

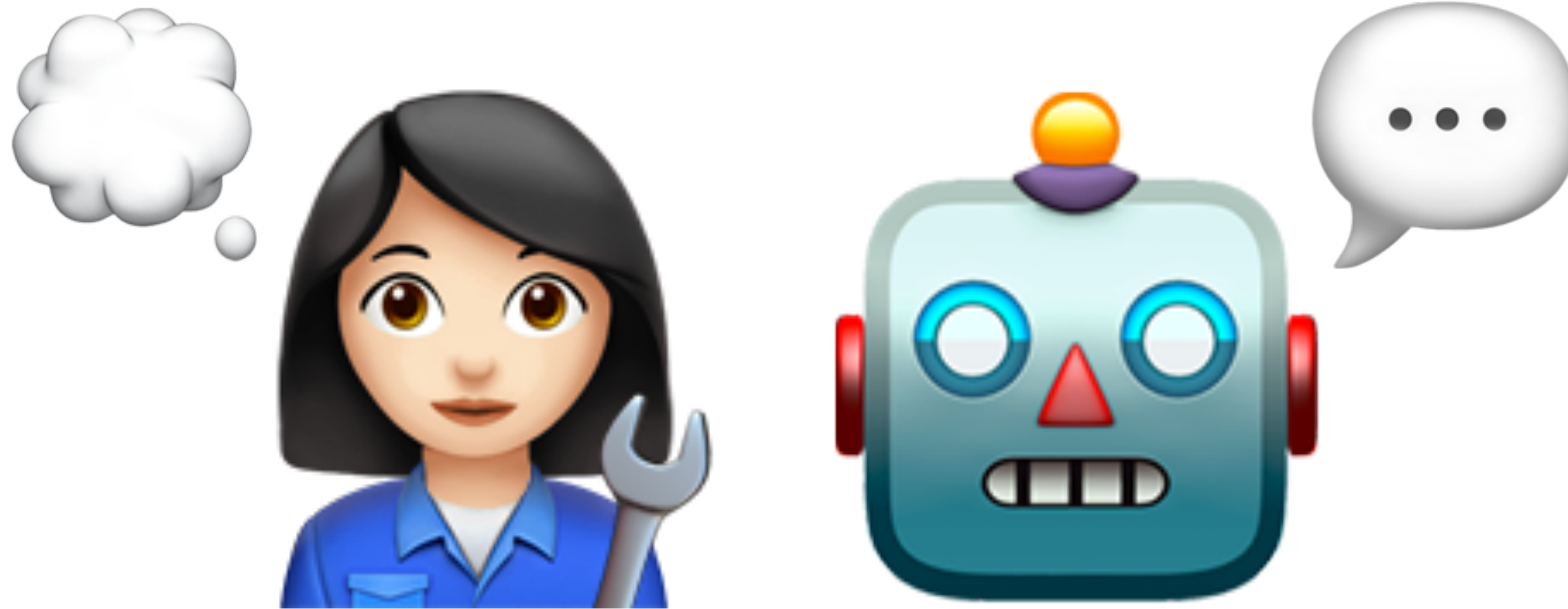
```
$("#textInput").keypress(function (e) {  
    if (e.which == 13 && $("#textInput").val().length > 0) {  
        getResponse();  
    }  
});  
  
$("#send").click(function () {  
    if ($("#textInput").val().length > 0) {  
        getResponse();  
    }  
});
```





# ✓ Consistency and Standards

*Users should not have to wonder whether different words, options, or actions mean the same thing. Follow platform conventions for the design of visual and interaction elements. Users should also be able to receive consistent responses even if they communicate the same function in multiple ways (and modalities). **Within the interaction, the system should have a consistent voice, style of language, and personality.***





# Context Preservation

*Maintain context preservation regarding the conversation topic **intra-session**, and if possible **inter-session**. Allow the user to **reference past messages** for further interactions to support implicit user expectations of conversations.*



# Context Preservation

*Maintain context preservation regarding the conversation topic intra-session, and if possible inter-session. Allow the user to reference past messages for further interactions to support implicit user expectations of conversations.*



# OK Context Preservation

*Maintain context preservation regarding the conversation topic intra-session, and if possible inter-session. Allow the user to reference past messages for further interactions to support implicit user expectations of conversations.*

```
def generate_prompt(human_input):
    global persona
    global prev_human
    global prev_bot

    prompt = """This is a conversation between
        {persona} and human.

        Human: {prev_human}
        {persona}: {prev_bot}
        Human: {human_input}
        {persona}: """.format(persona=persona,
                              prev_human=prev_human,
                              prev_bot=prev_bot,
                              human_input=human_input)

    return prompt
```

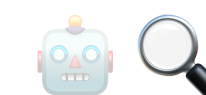
```
import openai

human_input = request.args.get("human_input")

prompt = generate_prompt(human_input)

response = openai.Completion.create(
    model=model,
    prompt=prompt,
    temperature=temperature,
    max_tokens=150,
    stop=[f"{persona}:", "Human:", "\n"],
).choices[0].text

return response
```



# OK Context Preservation

*Maintain context preservation regarding the conversation topic intra-session, and if possible inter-session. Allow the user to reference past messages for further interactions to support implicit user expectations of conversations.*

```
def generate_prompt(human_input):
    global persona
    global prev_human
    global prev_bot

    prompt = """This is a conversation between
        {persona} and human.

        Human: {prev_human}
        {persona}: {prev_bot}
        Human: {human_input}
        {persona}: """.format(persona=persona,
                              prev_human=prev_human,
                              prev_bot=prev_bot,
                              human_input=human_input)

    return prompt
```

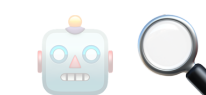
```
import openai

human_input = request.args.get("human_input")

prompt = generate_prompt(human_input)

response = openai.Completion.create(
    model=model,
    prompt=prompt,
    temperature=temperature,
    max_tokens=150,
    stop=[f"{persona}:", "Human:", "\n"],
).choices[0].text

return response
```



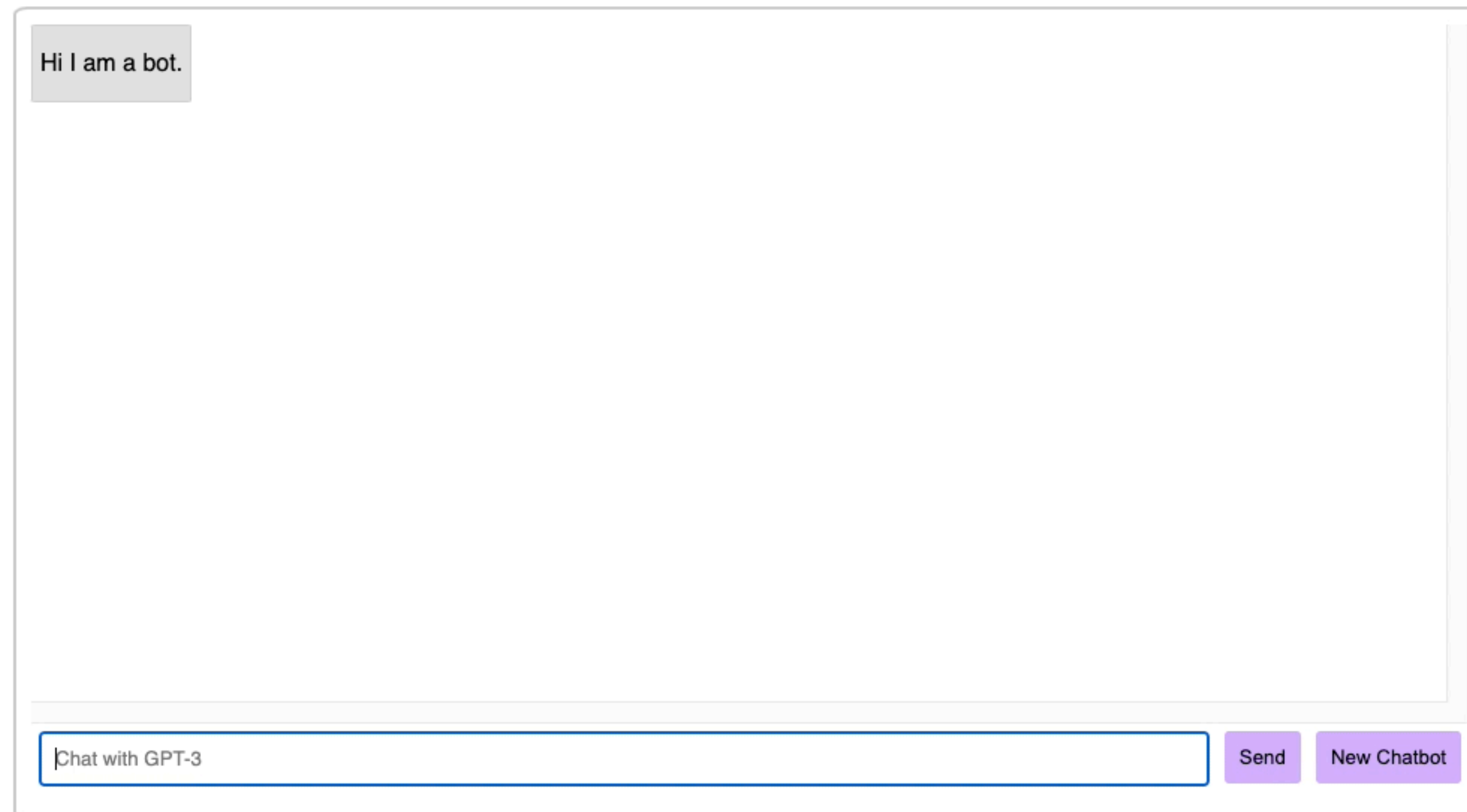
# Help Users Recognize, Diagnose and Recover from Errors

*Error messages should be expressed in **plain language** (no codes), **precisely indicate** the problem, and **constructively suggest** a solution.*



# Help Users Recognize, Diagnose and Recover from Errors

*Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.*



# Help Users Recognize, Diagnose and Recover from Errors

*Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.*

```
function getResponse () {
  ...
  var output = JSON.parse(botReplyJSON);
  var botReply = output.response;

  var botHTML = document.createElement("div");
  botHTML.className = "bot";
  var p = document.createElement("p");
  p.innerHTML = botReply;
  botHTML.append(p);

  if (!output.success) {
    botHTML.style.backgroundColor = "yellow";
  }

  $("#chatbox").append(botHTML);
  var chatbox = document.getElementById('chatbox');
  chatbox.scrollTop = chatbox.scrollHeight;
  ...
}
```





# Help Users Recognize, Diagnose and Recover from Errors

*Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.*

```
function getResponse () {
  ...
  var output = JSON.parse(botReplyJSON);
  var botReply = output.response;

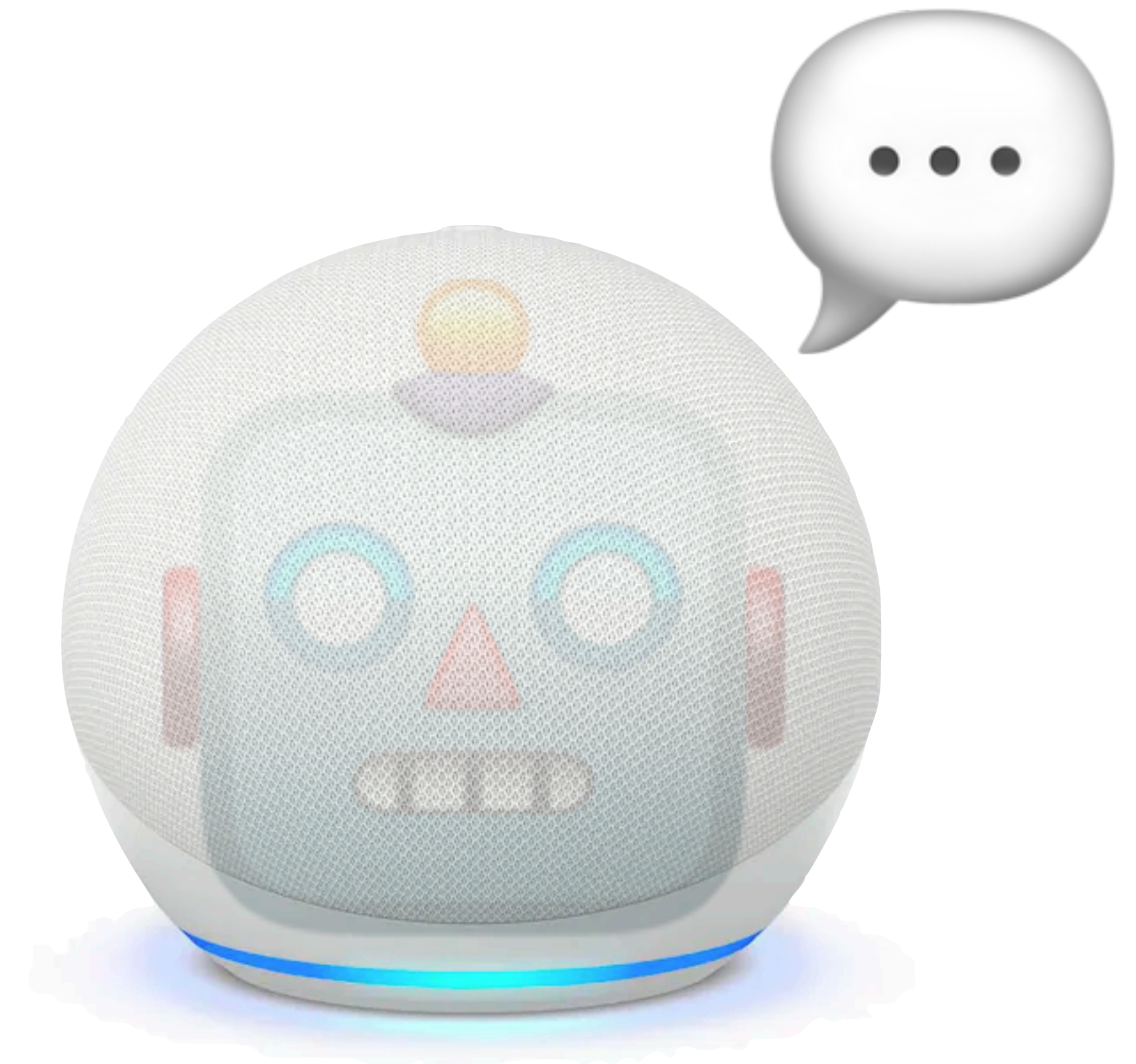
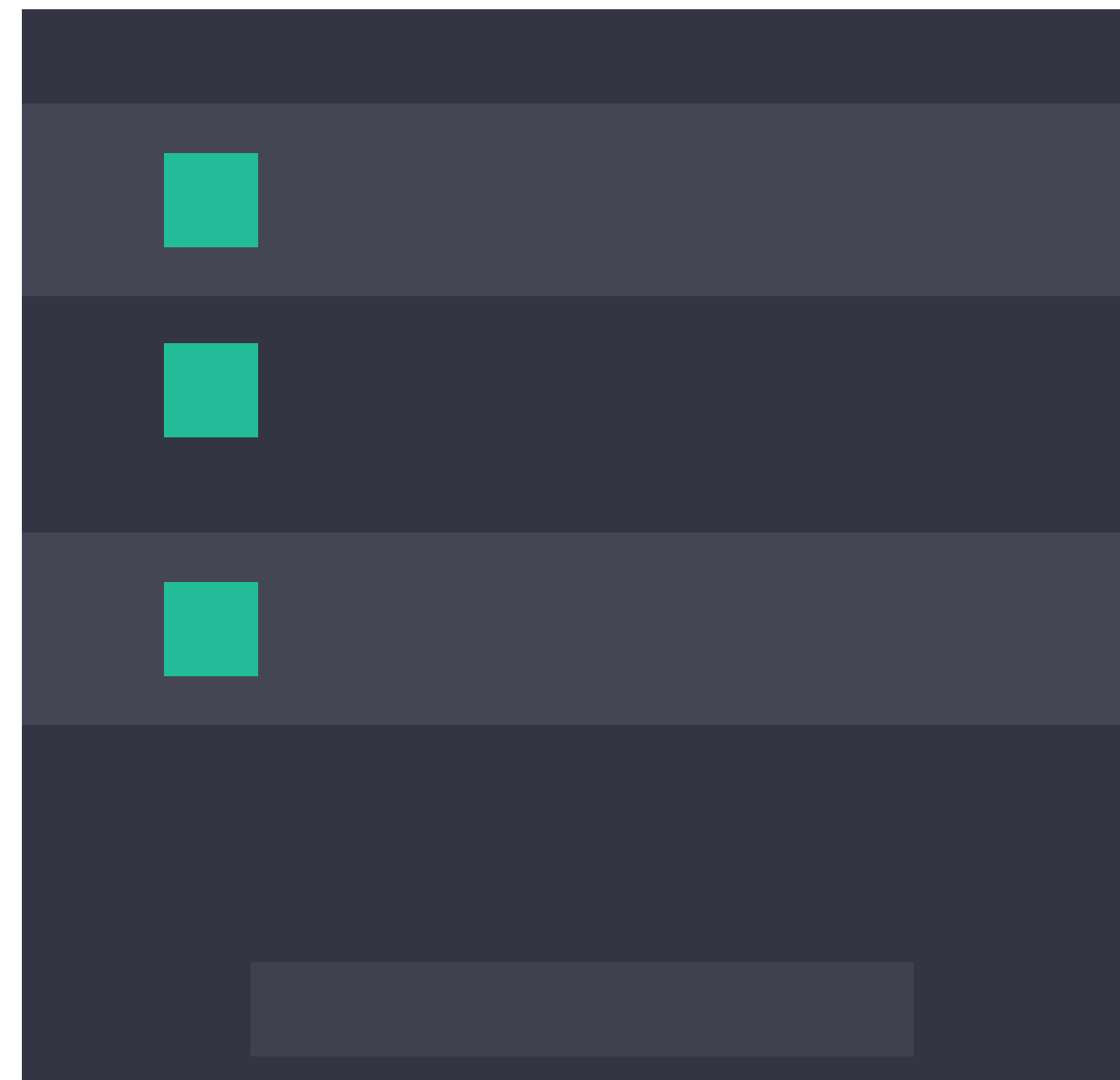
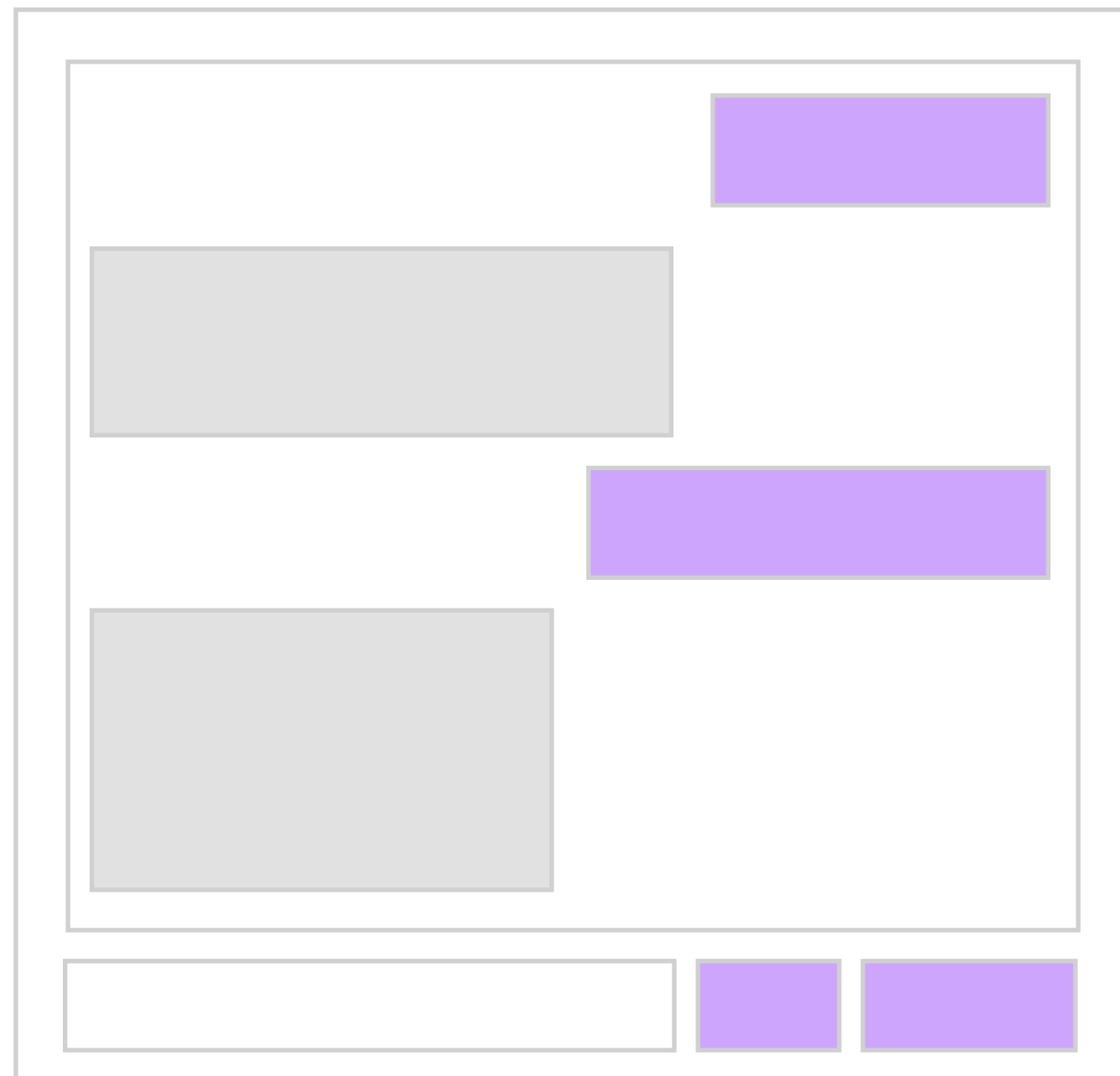
  var botHTML = document.createElement("div");
  botHTML.className = "bot";
  var p = document.createElement("p");
  p.innerHTML = botReply;
  botHTML.append(p);

  if (!output.success) {
    botHTML.style.backgroundColor = "yellow";
  }

  $("#chatbox").append(botHTML);
  var chatbox = document.getElementById('chatbox');
  chatbox.scrollTop = chatbox.scrollHeight;
  ...
}
```



# Every new AI interface shares similar foundations.



# Thank you for listening!

Questions + Discussion



Ask me about  
GPT-3, design,  
or my chatbot!