Learning Objectives

Learners will be able to...

- **Describe how a chatbot works**
- **Create a simple chatbot in Python with a tkinter GUI**
- **Create a simple chatbot using NLTK with a tkinter GUI**
- **Create a chatbot powered by OpenAI**

info

Make Sure You Know

You are familiar with Python.

Limitations

This is a gentle introduction. So there is a little bit of Python programming. The information might not be the most up to date as OpenAI releases new features.

Introduction to Chatbots

One of the most common tasks and a subset of Natural Language Processing is **Natural Language Generation** (NLG).

- It is also sometimes described as the opposite of speech recognition.
- It's the task of generating human language text responses by putting together structured information.
- This information follows **syntactical** and **semantic** rules from a program's knowledge base.

The best examples of real-world applications of this concept are **virtual agents** and **chatbots**.
- Apple's Siri and Amazon's Alexa are examples of virtual agents.
- They use:
- **Speech recognition** to understand user commands
- **NLG** to respond with the required information or conduct the requested action.

A **Chatbot**, on the other hand, operates on **typed text**.
- It processes and simulates **human dialogs** and **conversations**, allowing users to interact with various kinds of digital services.

- They can be as simple as a program comprised of a set of **hard-coded rules**, to very sophisticated **AI-based** chatbots that deliver **personalized responses** and learn as they go.

Connecting to GitHub

Connecting to GitHub

In this lab, you are going to build a simple Chatbot. In order to showcase this project for your portfolio, you are going to use GitHub. If you do not yet have an account, please create one now. We are going to clone a repo that will contain the code for your Chatbot.

Connecting GitHub and Codio

You need to connect GitHub to your Codio account. This only needs to be done one time.

- In your Codio account, click on your username
- Click on **Applications**

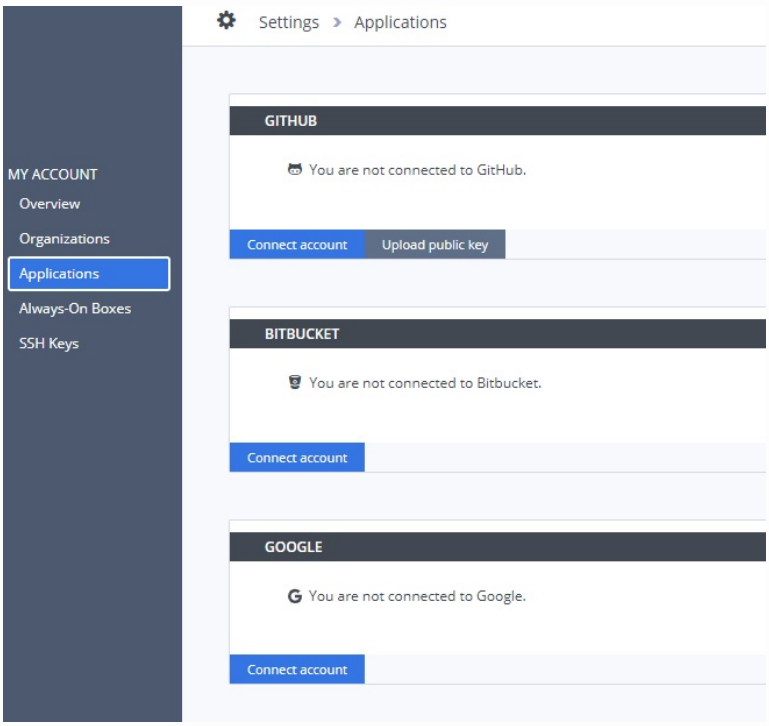
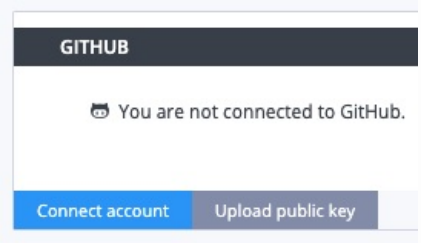


Image showing codio account bar. The word Application is highlighted

- Under GitHub, click on **Connect account**



the image displaying the words connect account in blue and upload public key in gray

- You will be using an SSH connection, so you need to click on **Upload public key**

Fork the Repo

- Go to the nlp-simplebot repo. This repo is the starting point for your project.
- Click on the “Fork” button in the top-right corner.
- Click the green “Create fork” button.
- Click the green “Code” button.
- Copy the **SSH** information. It should look something like this:

```
git@github.com:<your_github_username>/nlp-simplebot.git
```

info

Important

If you do not use the SSH information, you will have to provide your username and Personal Access Token (PAT) to GitHub each time you push or pull from the repository. See this documentation for setting up a PAT for your GitHub account.

In the Terminal

- Clone the repo. Your command should look something like this:

```
git clone git@github.com:<your_github_username>/nlp-simplebot.git
```

- You should see a `nlp-simplebot` directory appear in the file tree.
- When asked, you should type yes

On the last page, we’ll show you how to push to GitHub.

You are now ready for the lab!

Initializing the GUI

A simple, rule-based chatbot.

info

Writing Code

To your left, you should see an opened file called `simplebot.py`
Use this file and reference the syntax below to work on this Lab.

Let's start by building a GUI for our chatbot. We'll use ***tkinter***, the standard Python GUI library.

The first step is to import all the necessary modules into our file.
Paste the following code in the top left pane:

```
import tkinter.scrolledtext as tks #creates a scrollable text
        window


from datetime import datetime
from tkinter import *
```

To make a simple GUI for our chatbot, it should have 4 main components:
1. `baseWindow` - the main GUI window that contains everything
2. `chatWindow` - displays the conversation between a user and the chatbot
3. `userEntryBox` - for the user to type their queries for the Chatbot
4. `sendButton` - a button that sends the user query to the Chatbot

Let's use `tkinter` to initialize the `baseWindow` and the other components and place them on the `baseWindow`.

Paste the following code in the top left pane under the import statements:

```
# Create the main application window using Tk()
baseWindow = Tk()

# Set the title of the window
baseWindow.title("The Simple Bot")

# Set the size of the window
baseWindow.geometry("500x250")

# Create the chat window as a ScrolledText widget with "Arial"
        font
chatWindow = tks.ScrolledText(baseWindow, font="Arial")

# Configure tags for message alignment: 'tag-left' for bot
        messages, 'tag-right' for user messages
chatWindow.tag_configure('tag-left', justify='left')
chatWindow.tag_configure('tag-right', justify='right')

# Disable the chat window initially (it should not be editable
        by the user)
chatWindow.config(state=DISABLED)

# Create the send button, with specific font, text, and
        background color
# The 'command' option is commented out. Uncomment it and
        replace 'send' with your send function's name
sendButton = Button(
    baseWindow,
    font=("Verdana", 12, 'bold'),
    text="Send",
    bg="#fd94b4",
    activebackground="#ff467e",
    fg='#ffffff',
    # command=send
)

# Create the user entry box where the user types their messages
userEntryBox = Text(baseWindow, bd=1, bg="white", width=38,
        font="Arial")

# Place the chat window, user entry box, and send button on the
        main window using specific coordinates and sizes
chatWindow.place(x=1, y=1, height=200, width=500)
userEntryBox.place(x=3, y=202, height=27)
sendButton.place(x=430, y=200)

# Start the main event loop to keep the application running and
        responsive
baseWindow.mainloop()
```

Let’s close the GUI and look at how we can attach a function to the send button!

Implementing functions for the GUI

At this point, the `sendButton` doesn't do anything.
Let's define a function called `send` which should do the following:

- Collect the `user_input` from the `textBox`
- Get the `bot_response`
- Insert both of these into the `chatWindow`

Paste the following code in the top left pane, right under the import statements:

```
def send(event):
    chatWindow.config(state=NORMAL)

    user_input = userEntryBox.get("1.0",'end-2c')
    user_input_lc = user_input.lower()
    bot_response = get_bot_response(user_input_lc)

    create_and_insert_user_frame(user_input)
    create_and_insert_bot_frame(bot_response)

    chatWindow.config(state=DISABLED)
    userEntryBox.delete("1.0","end")
    chatWindow.see('end')
```

Once we have the `user_input` and `bot_response`, we'll write a couple of functions to insert them into the `chatWindow`

Paste the following code in the top left pane, right above the `send` function:

```
def create_and_insert_user_frame(user_input):
  userFrame = Frame(chatWindow, bg="#d0ffff")
  Label(
      userFrame,
      text=user_input,
      font=("Arial", 11),
      bg="#d0ffff").grid(row=0, column=0, sticky="w", padx=5,
        pady=5)
  Label(
      userFrame,
      text=datetime.now().strftime("%H:%M"),
      font=("Arial", 7),
      bg="#d0ffff"
  ).grid(row=1, column=0, sticky="w")

  chatWindow.insert('end', '\n ', 'tag-right')
  chatWindow.window_create('end', window=userFrame)


def create_and_insert_bot_frame(bot_response):
  botFrame = Frame(chatWindow, bg="#ffffd0")
  Label(
      botFrame,
      text=bot_response,
      font=("Arial", 11),
      bg="#ffffd0",
      wraplength=400,
      justify='left'
  ).grid(row=0, column=0, sticky="w", padx=5, pady=5)
  Label(
      botFrame,
      text=datetime.now().strftime("%H:%M"),
      font=("Arial", 7),
      bg="#ffffd0"
  ).grid(row=1, column=0, sticky="w")

  chatWindow.insert('end', '\n ', 'tag-left')
  chatWindow.window_create('end', window=botFrame)
  chatWindow.insert(END, "\n\n" + "")
```

Now, we want to bind the `send` function to the `sendButton`. We can also bind the `Enter` key to the window so we don't have to click on the button every time.

Replace the `sendButton` initialization in the top left pane with the following code:

```
sendButton = Button(
    baseWindow,
    font=("Verdana", 12, 'bold'),
    text="Send",
    bg="#fd94b4",
    activebackground="#ff467e",
    fg='#ffffff',
    command=send)
sendButton.bind("<Button-1>", send)
baseWindow.bind('<Return>', send)
```

▼ Code

Your code should look like this:

```
import tkinter.scrolledtext as tks #creates a scrollable text
        window

from datetime import datetime
from tkinter import *


def create_and_insert_user_frame(user_input):
  userFrame = Frame(chatWindow, bg="#d0ffff")
  Label(
      userFrame,
      text=user_input,
      font=("Arial", 11),
      bg="#d0ffff").grid(row=0, column=0, sticky="w", padx=5,
        pady=5)
  Label(
      userFrame,
      text=datetime.now().strftime("%H:%M"),
      font=("Arial", 7),
      bg="#d0ffff"
  ).grid(row=1, column=0, sticky="w")

  chatWindow.insert('end', '\n ', 'tag-right')
  chatWindow.window_create('end', window=userFrame)


def create_and_insert_bot_frame(bot_response):
  botFrame = Frame(chatWindow, bg="#ffffd0")
  Label(
      botFrame,
      text=bot_response,
      font=("Arial", 11),
```

```
        bg="#ffffd0",
        wraplength=400,
        justify='left'
    ).grid(row=0, column=0, sticky="w", padx=5, pady=5)
    Label(
        botFrame,
        text=datetime.now().strftime("%H:%M"),
        font=("Arial", 7),
        bg="#ffffd0"
    ).grid(row=1, column=0, sticky="w")

    chatWindow.insert('end', '\n ', 'tag-left')
    chatWindow.window_create('end', window=botFrame)
    chatWindow.insert(END, "\n\n" + "")


def send(event):
    chatWindow.config(state=NORMAL)

    user_input = userEntryBox.get("1.0",'end-2c')
    user_input_lc = user_input.lower()
    bot_response = get_bot_response(user_input_lc)

    create_and_insert_user_frame(user_input)
    create_and_insert_bot_frame(bot_response)

    chatWindow.config(state=DISABLED)
    userEntryBox.delete("1.0","end")
    chatWindow.see('end')


baseWindow = Tk()
baseWindow.title("The Simple Bot")
baseWindow.geometry("500x250")

chatWindow = tks.ScrolledText(baseWindow, font="Arial")
chatWindow.tag_configure('tag-left', justify='left')
chatWindow.tag_configure('tag-right', justify='right')
chatWindow.config(state=DISABLED)

sendButton = Button(
    baseWindow,
    font=("Verdana", 12, 'bold'),
    text="Send",
    bg="#fd94b4",
    activebackground="#ff467e",
    fg='#ffffff',
    command=send)
```

```
sendButton.bind("<Button-1>", send)
baseWindow.bind('<Return>', send)

userEntryBox = Text(baseWindow, bd=1, bg="white", width=38,
        font="Arial")

chatWindow.place(x=1, y=1, height=200, width=500)
userEntryBox.place(x=3, y=202, height=27)
sendButton.place(x=430, y=200)

baseWindow.mainloop()
```

Now, the only thing left is to define the `bot_response` function!

The bot response function

The `bot_response` function is the logic function for the chatbot. This is where we will define the rules that the chatbot will follow to respond to `user_input`

Paste the following code in the top left pane, right below the import statements:

```
# Generating response
def get_bot_response(user_input):

  bot_response = ""
  if(user_input == "hello"):
    bot_response = "Hi!"
  elif(user_input == "hi" or user_input == "hii" or user_input
       == "hiiii"):
    bot_response = "Hello there! How are you?"
  elif(user_input == "how are you"):
    bot_response = "Oh, I'm great! How about you?"
  elif(user_input == "fine" or user_input == "i am good" or
       user_input == "i am doing good"):
    bot_response = "That's excellent! How can I help you today?"
  else:
    bot_response = "I'm sorry, I don't understand..."

  return bot_response
```

▼ Code

Your code should look like this:

```
import tkinter.scrolledtext as tks #creates a scrollable text
        window

from datetime import datetime
from tkinter import *


# Generating response
def get_bot_response(user_input):

  bot_response = ""
  if(user_input == "hello"):
```

```
    bot_response = "Hi!"
  elif(user_input == "hi" or user_input == "hii" or user_input
        == "hiiii"):
    bot_response = "Hello there! How are you?"
  elif(user_input == "how are you"):
    bot_response = "Oh, I'm great! How about you?"
  elif(user_input == "fine" or user_input == "i am good" or
        user_input == "i am doing good"):
    bot_response = "That's excellent! How can I help you today?"
  else:
    bot_response = "I'm sorry, I don't understand..."

  return bot_response


def create_and_insert_user_frame(user_input):
  userFrame = Frame(chatWindow, bg="#d0ffff")
  Label(
      userFrame,
      text=user_input,
      font=("Arial", 11),
      bg="#d0ffff").grid(row=0, column=0, sticky="w", padx=5,
        pady=5)
  Label(
      userFrame,
      text=datetime.now().strftime("%H:%M"),
      font=("Arial", 7),
      bg="#d0ffff"
  ).grid(row=1, column=0, sticky="w")

  chatWindow.insert('end', '\n ', 'tag-right')
  chatWindow.window_create('end', window=userFrame)


def create_and_insert_bot_frame(bot_response):
  botFrame = Frame(chatWindow, bg="#ffffd0")
  Label(
      botFrame,
      text=bot_response,
      font=("Arial", 11),
      bg="#ffffd0",
      wraplength=400,
      justify='left'
  ).grid(row=0, column=0, sticky="w", padx=5, pady=5)
  Label(
      botFrame,
      text=datetime.now().strftime("%H:%M"),
      font=("Arial", 7),
      bg="#ffffd0"
```

```
  ).grid(row=1, column=0, sticky="w")

  chatWindow.insert('end', '\n ', 'tag-left')
  chatWindow.window_create('end', window=botFrame)
  chatWindow.insert(END, "\n\n" + "")


def send(event):
    chatWindow.config(state=NORMAL)

    user_input = userEntryBox.get("1.0",'end-2c')
    user_input_lc = user_input.lower()
    bot_response = get_bot_response(user_input_lc)

    create_and_insert_user_frame(user_input)
    create_and_insert_bot_frame(bot_response)

    chatWindow.config(state=DISABLED)
    userEntryBox.delete("1.0","end")
    chatWindow.see('end')


baseWindow = Tk()
baseWindow.title("The Simple Bot")
baseWindow.geometry("500x250")

chatWindow = tks.ScrolledText(baseWindow, font="Arial")
chatWindow.tag_configure('tag-left', justify='left')
chatWindow.tag_configure('tag-right', justify='right')
chatWindow.config(state=DISABLED)

sendButton = Button(
    baseWindow,
    font=("Verdana", 12, 'bold'),
    text="Send",
    bg="#fd94b4",
    activebackground="#ff467e",
    fg='#ffffff',
    command=send)
sendButton.bind("<Button-1>", send)
baseWindow.bind('<Return>', send)

userEntryBox = Text(baseWindow, bd=1, bg="white", width=38,
        font="Arial")

chatWindow.place(x=1, y=1, height=200, width=500)
userEntryBox.place(x=3, y=202, height=27)
sendButton.place(x=430, y=200)
```

```
baseWindow.mainloop()
```

This was the last step! We have successfully built our first chatbot! Let's run it and interact with it!

You might need to refresh, your lower left panel.

> important
>
> ## Limitations of this chatbot
>
> This was a simple example of a hard-coded chatbot. This doesn't use any NLP concepts and does simple string matching to generate responses.

Let's see how we can make this better by using the GPT-3.

The bot response function using GPT-3

At this point, you must be familiar with OpenAI GPT-3 Toolkit. We are going to interact with our prompt to create a response. How we are going to interact with it is by changing our `get_bot_response` function.

info

Writing Code

To your left, you should see an opened file called `test.py`
Use this file as demo before we change our code .

Let's see how we can use `openai.Completion.create` to change our simple bot!

In the top left panel, the code file should have the same code as library as we are used to when interacting with GPT-3.

```
import os
import openai
openai.api_key=os.getenv('OPENAI_KEY')
```

Now we are going to create a function `get_bot_response`.The function `get_bot_response(user_input)` generates a bot response by sending a user's input to the OpenAI GPT-3 model, "text-davinci-002", and returns the model's generated text as the bot's response, stripped of leading and trailing whitespace.

```
# Generating response
def get_bot_response(user_input):
    prompt = f"Please provide a response to the following user
        input: '{user_input}'"

    response = openai.Completion.create(model="text-davinci-
        002",
        prompt=prompt,
        max_tokens=150,
        n=1,
        stop=None,
        temperature=0.5,
    )

    bot_response = response.choices[0].text.strip()
    return bot_response
```

After making sure our code runs without errors we are ready to put it in our main code. Please make sure to copy everything in `test.py` and paste it after `from tkinter import *` in our `simplebot.py` (bottom left panel). Also make sure to delete our old `get_bot_response` function.

To run our code and try it: please click the button below. Use the other tab in the bottom-left panel

▼ Code

Your code should look like this:

```
import tkinter.scrolledtext as tks #creates a scrollable text
        window
import os
import openai

openai.api_key=os.getenv('OPENAI_KEY')

from datetime import datetime
from tkinter import *

# Generating response
def get_bot_response(user_input):
    prompt = f"Please provide a response to the following user
        input: '{user_input}'"

    response = openai.Completion.create(model="text-davinci-
        002",
        prompt=prompt,
        max_tokens=150,
        n=1,
```

```
        stop=None,
        temperature=0.5,
    )

    bot_response = response.choices[0].text.strip()
    return bot_response




def create_and_insert_user_frame(user_input):
  userFrame = Frame(chatWindow, bg="#d0ffff")
  Label(
      userFrame,
      text=user_input,
      font=("Arial", 11),
      bg="#d0ffff").grid(row=0, column=0, sticky="w", padx=5,
        pady=5)
  Label(
      userFrame,
      text=datetime.now().strftime("%H:%M"),
      font=("Arial", 7),
      bg="#d0ffff"
  ).grid(row=1, column=0, sticky="w")

  chatWindow.insert('end', '\n ', 'tag-right')
  chatWindow.window_create('end', window=userFrame)


def create_and_insert_bot_frame(bot_response):
  botFrame = Frame(chatWindow, bg="#ffffd0")
  Label(
      botFrame,
      text=bot_response,
      font=("Arial", 11),
      bg="#ffffd0",
      wraplength=400,
      justify='left'
  ).grid(row=0, column=0, sticky="w", padx=5, pady=5)
  Label(
      botFrame,
      text=datetime.now().strftime("%H:%M"),
      font=("Arial", 7),
      bg="#ffffd0"
  ).grid(row=1, column=0, sticky="w")

  chatWindow.insert('end', '\n ', 'tag-left')
  chatWindow.window_create('end', window=botFrame)
```

```
  chatWindow.insert(END, "\n\n" + "")




def send(event):
    chatWindow.config(state=NORMAL)

    user_input = userEntryBox.get("1.0",'end-2c')
    user_input_lc = user_input.lower()
    bot_response = get_bot_response(user_input_lc)

    create_and_insert_user_frame(user_input)
    create_and_insert_bot_frame(bot_response)

    chatWindow.config(state=DISABLED)
    userEntryBox.delete("1.0","end")
    chatWindow.see('end')



baseWindow = Tk()
baseWindow.title("The Simple Bot")
baseWindow.geometry("500x250")

chatWindow = tks.ScrolledText(baseWindow, font="Arial")
chatWindow.tag_configure('tag-left', justify='left')
chatWindow.tag_configure('tag-right', justify='right')
chatWindow.config(state=DISABLED)

sendButton = Button(
    baseWindow,
    font=("Verdana", 12, 'bold'),
    text="Send",
    bg="#fd94b4",
    activebackground="#ff467e",
    fg='#ffffff',
    command=send)
sendButton.bind("<Button-1>", send)
baseWindow.bind('<Return>', send)

userEntryBox = Text(baseWindow, bd=1, bg="white", width=38,
        font="Arial")

chatWindow.place(x=1, y=1, height=200, width=500)
userEntryBox.place(x=3, y=202, height=27)
sendButton.place(x=430, y=200)
```

```
baseWindow.mainloop()
```

And that's it! We've built our first chatbot using GPT-3! Let's run it and interact with it!

Pushing to GitHub

Pushing to GitHub

For the final step, you must push your work to GitHub.

1. On the left side, you should see a file tree with the `nlp-simplebot` folder.
2. Click on your `simplebot.py` file outside of the folder and copy its contents.
3. Now, go inside the `nlp-simplebot` folder and open the `simplebot.py` file.
4. Paste the contents of your `simplebot.py` file into the `simplebot.py` file within the `nlp-simplebot` folder.
5. In the terminal, run the command `cd nlp-simplebot` to move into the folder connected to GitHub.

- Now we can Commit your changes:

```
git add .
git commit -m "Finished The Simple Bots"
```

- Push to GitHub:

```
git push
```