MLH localhost

Workshop

Basic Training: Intro to Python Skills for AI, Part III

1

Using your Web Browser,
Open this URL:

http://mlhlocal.host/lhd-resources

2

Click on the workshop you're attending, and find:

- Setup Instructions
- The Code Samples
- A demo project
- A Workshop FAQ
- These Workshop Slides
- More Learning Resources

MLH

MAJOR LEAGUE HACKING

Our Mission is to Empower Hackers.

65,000+
HACKERS

12,000+
PROJECTS CREATED

400+
CITIES

We hope you learn something awesome today!

Find more resources: http://mlh.io/

Our unique expression of social good

Capital One is dedicated to providing opportunities and resources that will enable more people to succeed.

Through our Future Edge program, we're investing in and collaborating with leading educational and community organizations across the U.S.—to help more people succeed in the 21st century.

We're empowering families through financial literacy and affordable housing, helping individuals bridge the digital skills gap and showing small businesses how to harness technology to grow and compete.

What will you learn today?

1. How to include and work with a Python library
2. How to debug your code
3. You'll also review material from the first two workshops!

Why does this matter?

1. Many artificial intelligence projects are on the web. You want to be able to share your cool work with others!
2. Learning how to use third party libraries is a skill you'll use in any programming language.
3. Once you learn one programing language, you can learn any.

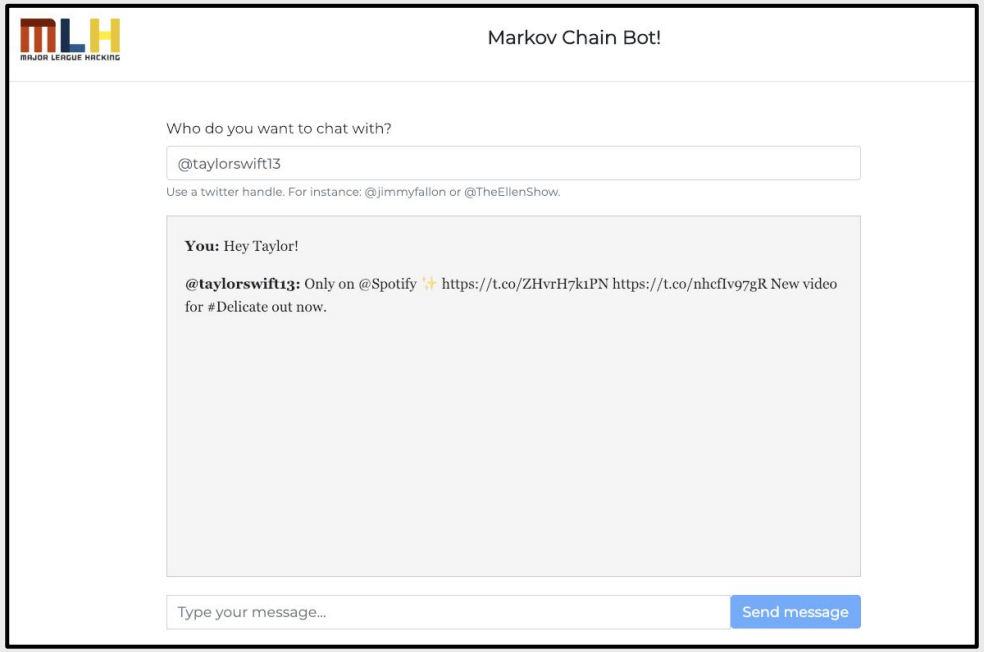
What do you remember from last time?

Discuss for a few minutes with someone around you.

What are you going to build today?

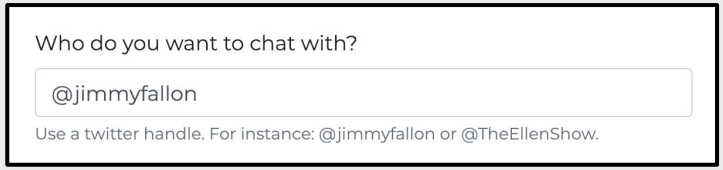
Today you're going to incorporate the markovify library into your web app so that instead of responding with a Tweet, your app responds with a message that sounds like something your celebrity might say!

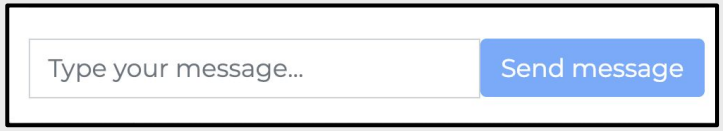
mlhlocal.host/glitch-markov



How does this work?

1. In the first box, the user enters a person's Twitter handle.
2. The user enters a message.
3. The Twitter user and message are sent to the app.
4. The app uses the Twitter API to request 10,000 tweets from that person.
5. The app uses a Python library called Flask to handle the requests and responses.
6. The app uses scraping and text formatting techniques to clean the tweets and pass them to a library called Markovify to generate responses!

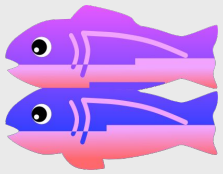




How are you going to rebuild it?

These are the steps you'll need to take:

1. You'll make your own copy of the code on Glitch.
2. A lot of the code is the same from the second workshop, so in this workshop you'll focus only on the parts that are different.
3. You'll incorporate the Markovify library.
4. You'll debug some issues!
5. Then, you'll recap and take a quiz.



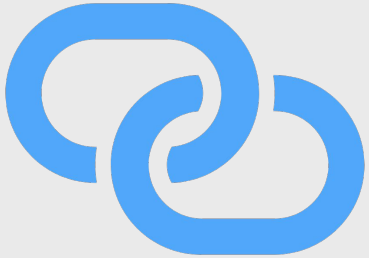
That may sound like a lot of steps. Part of coding/programming is learning how all the different tools work and how they work together.

We know you can do it!

How are you going to build it?

There are lots of tools that can make coding easier!

We're still using Python, Glitch, and Flask.

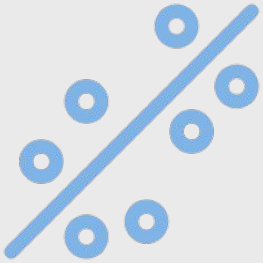


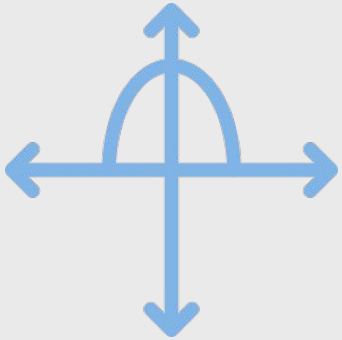
Today, we're adding Markovify. Markovify is a Python library that generates Markov Chains. We'll learn more about Markov chains later. For now, you need to know that the Markov Chain is what generates the fake responses from your Twitter celebrity!

What's a Markov chain?

In artificial intelligence, there are many different models.

- In math, you've seen different kinds of models.
- Some graphs you've worked with are linear (straight lines), some are quadratic (parabolas), etc
- A Markov chain is a complex model that is used to predict values in the future based on a given data set.
- We use this model to generate fake responses in our app!





Key Terms
model: a mathematical way to predict future values

A really simple example of a Markov Chain

Pretend there are only two possible types of weather - sunny or cloudy. That's it, two types.

Let's say that you want to be able to guess if the weather tomorrow will be sunny or cloudy.

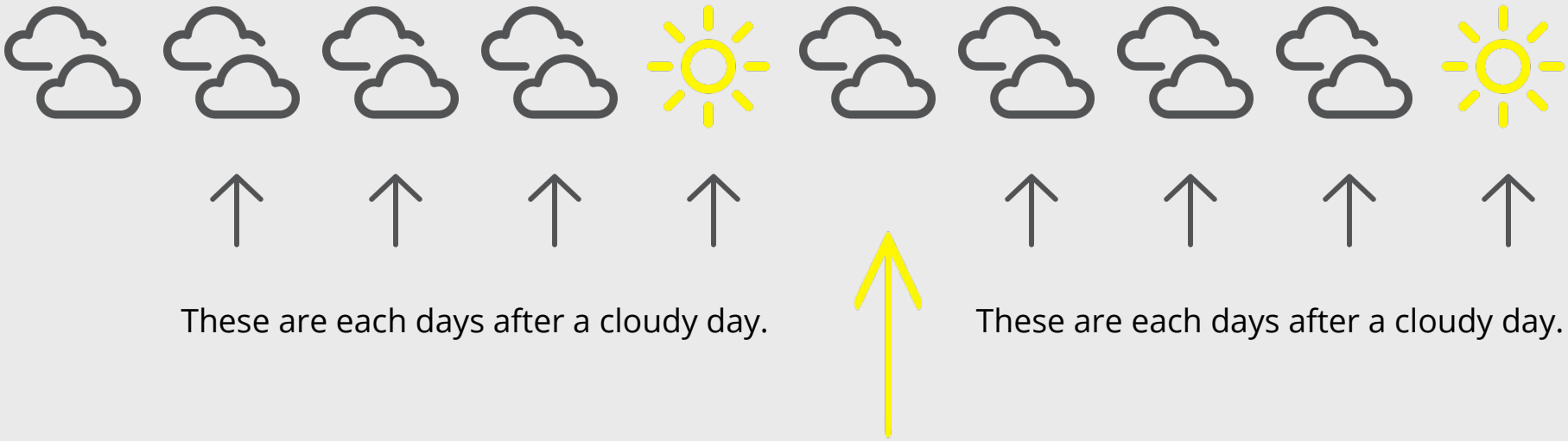
You look back through weather reports for the last year and write down whether every day was sunny or cloudy. You do some math, and you learn that over the last year, if any day was cloudy, 25% of the time the next day was sunny and 75% of the time the next day was cloudy.

You can use this information to make a table of what you think weather would look like for several days:

Thanks to Devin Soni for this explanation!

A really simple example of a Markov Chain

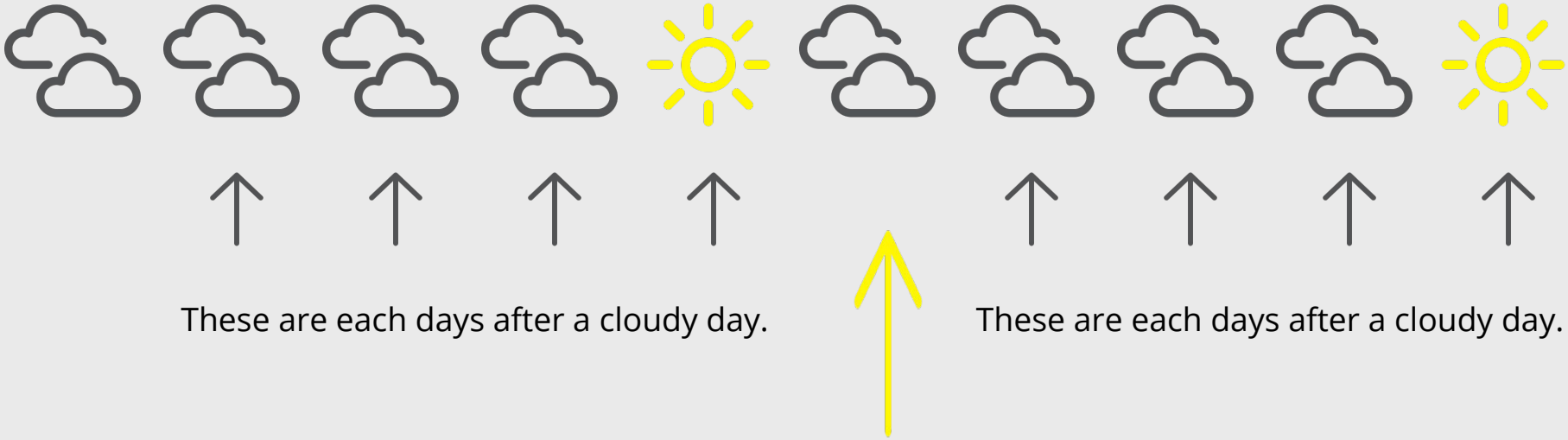
Let's test this out for ourselves!



A really simple example of a Markov Chain

In total, there are 8 days that come after cloudy days.

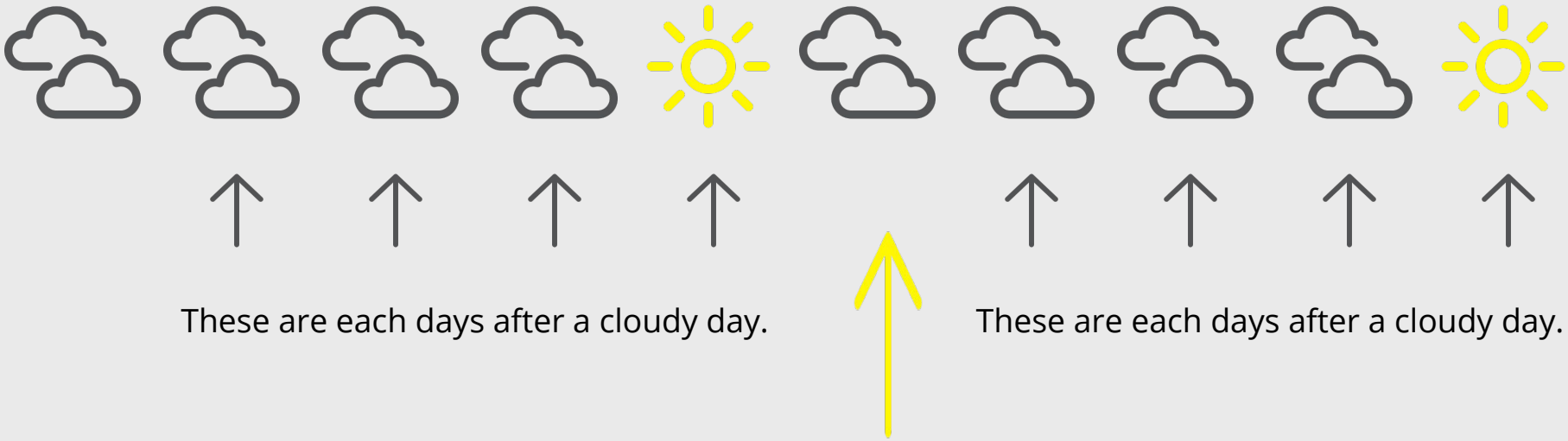
How many of those days are cloudy? How many are sunny?



A really simple example of a Markov Chain

6 of the days after cloudy days are also cloudy.

2 of the days after cloudy days are sunny.

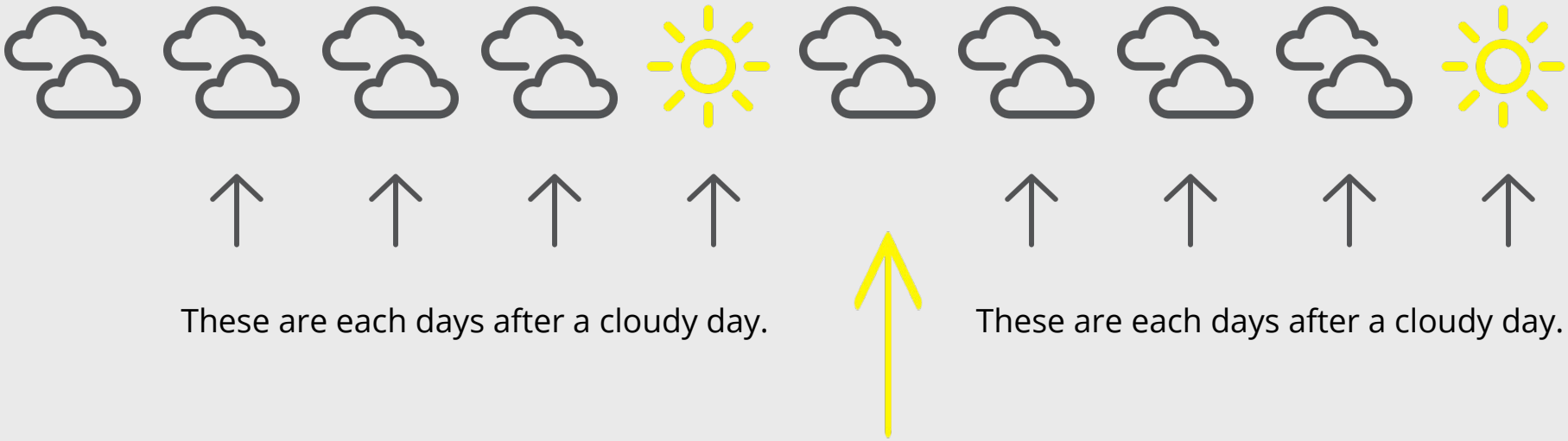


A really simple example of a Markov Chain

2 sunny days / 8 days after cloudy days = 25%

6 cloud days / 8 days after cloud days = 75%

We did it correctly!



A really simple example of a Markov Chain

But! The weather is actually way more complicated than just cloudy or sunny, right? Imagine if we added partly cloudy, and rainy. Now you'd have four different types of weather to keep track of! Calculating that by hand would be a nightmare!

Instead of doing this by hand, we'll use a Python library called Markovify!

README.md

pypi v0.7.1 | build passing | coverage 100% | python 2.7 | 3.4 | 3.5 | 3.6

Markovify

Markovify is a simple, extensible Markov chain generator. Right now, its main use is for building Markov models of large corpora of text, and generating random sentences from that. But, in theory, it could be used for other applications.

In our example, we were using Markov Chains to analyze weather (sunny or cloudy). You can use this for ANYTHING you can count.

We'll use Markov to predict what words come after other words based on a person's Tweets!

What's Markovify?

Markovify is a Python library written and maintained by GitHub user jsvine.

- Markovify can be added to any Python project.
- Markovify takes a large input of text and runs the same sort of calculations on it that we learned in the last few slides (but it can do it for thousands of different options rather than just cloudy and sunny).
- Markovify calculates how often one word appears after the next in the text input. Simply put, it makes a list of each word in the text. Then, it makes a list of what words come next, and how many times each pattern repeats (like we did with the clouds).
- It uses those percentages to generate fake sentences based on how likely one word is to follow another.
- It's not perfect, and that's why our app has some strange output sometimes!

Some things to keep in mind

Writing code is tricky. Pay attention to the following best practices

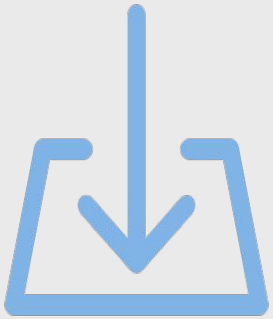
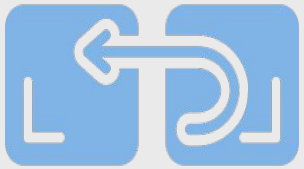
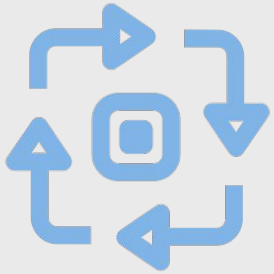
1. The majority of the time, when something isn't working with your code, it's as simple as a typo. Make sure that you've typed everything correctly before moving on to other debugging techniques.
2. Glitch has a really cool rewind feature. If your code was working before, and it's not working now, use the rewind feature to back up.
3. It's okay to make mistakes! You're learning something new, and that's really tricky.

Let's get started!

Table of Contents

What Python Skills Did We Learn Last Time?

1. **Regular expressions** – a programming concept used to work with text
2. **Imports** – using data or functions from other sources (like libraries or other files in our project)

The best way to learn to code is to CODE, not to read about it, so let's dive in!

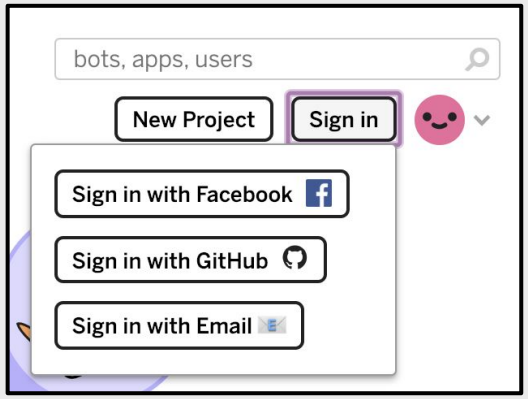
Table of Contents

Glitch

http://mlhlocal.host/glitch

1. Navigate to the URL above.

2. Click **Sign in**. Choose how you want to log in.



Get the code!

mlhlocal.host/glitch-python-iii

Go to the URL above. Click **Remix Your Own**

Get the code!

1. Add the the following code to `.env file`

```
MLH_TWITTER_API=https://localhost-python-abstraction.glitch.me
TWITTER_FETCHER=scraper
FLASK_ENV=development
FLASK_PORT=5000

```

In the last workshop, you used Twitter scraping and data cleaning to send a single Tweet back to the user.

But that's not very interesting! Let's do something cooler in this workshop.

Let's explore the files

new-python-iii-starter | Show Live | README.md

Share

+ New File

- assets
- static/logo.svg
- static/script.js
- static/style.css
- templates/homepage.html
- .env.example
- LICENSE
- README.md
- config.py
- content_moderator.py
- glitch.json
- main.py
- markov.py
- mlh_twitter_api.py
- requirements.txt
- resources.md
- twitter_scraper_fetcher.py

Markdown

MLH Localhost <> Capital One

This project contains the starter code for the MLH Localhost Workshop Intro to Python I

Your Project

Click `Show` above and it will launch the app in a new browser tab. It starts by running the `main.py` program.

← README.md

That's this file!

← twitter_scraper_fetcher.py

A Python file that contains functions which scrape data from Twitter.

← config.py

Code Review

These are the files we'll be working with today:

`mlh_twitter_api.py`: This is a file that will allow us to use the Twitter API to return way more Tweets than we've returned in the past. We'll learn why this matters!

`markov.py`: This file is where we'll write the code to use the Markovify library to create bot responses!

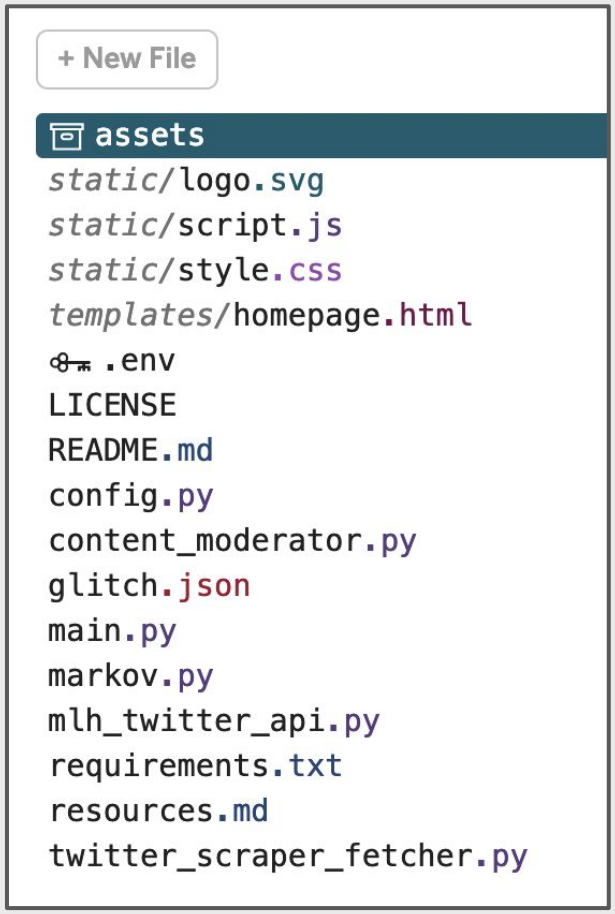


Table of Contents

Let's write some code!

Open **`markov.py`**. There are three places in this file where you'll need to write some code. Let's identify them now!

```
1                    ## write code here!!
2  import config
3  from mlh_twitter_api import get_user_tweets as fetch
4  from twitter_scraper_fetcher import *
5  import re
```

You'll write one line of code on Line 1.

```
18 def generate_bot_answer_with_text_model(twitter_handle, user_question, text_model):
19     pass
20     ## write code here
21
22
23
24
```

You'll write this function.

```
28  # build the markov chain based on the text we read
29  # we use the markovify library to do this step
30  def generate_bot_answer(twitter_handle, user_question):
31      pass
32
33      ## write code here
34
```

And this one too!

Write code: `markov.py`

On line 1, we need to allow our file to use the **`markovify`** library. How do you think you'd add that code?

```
1   ## write code here
2   import config
3   from mlh_twitter_api import get_user_tweets as fetch
4   from twitter_scraper_fetcher import *
5   import re
6
```

Let's write some code!

Did you do it correctly?

```
import markovify
import config
from mlh_twitter_api import get_user_tweets as fetch
from twitter_scraper_fetcher import *
import re

```

Are you ready for the hardest part of this whole series?

We know you can do it!

Let's write some code!

```
19  def generate_bot_answer_with_text_model(twitter_handle, user_question, text_model):
20      pass
21      ## write code here
```

On **Line 14**, we have started the definition of a function called **`generate_bot_answer_with_text_model()`**. This function needs to do several things:

- Create a variable called **`bot_answer`** and set its value to **`None`**.
- Split the **`user_question`** input into a list of words.
- Retrieve one word from that list.
- Pass that word into the **`make_sentence_with_start()`** function from Markovify, call that function on the **`text_model`** input, and save the result to **`bot_answer`**.
- Return **`bot_answer`**.

Let's write some code!

```
14  def generate_bot_answer_with_text_model(...):
15    bot_answer = None
16
17
18
19
20
```

1. On **Line 15**, start by creating the **`bot_answer`** variable and set its value to **`None`** and delete the keyword **pass**

Let's write some code!

```
14  def generate_bot_answer_with_text_model(...):
15    bot_answer = None
16
17    word_list = user_question.split(' ')
18
19
20
```

2. On **Line 17**, split the **`user_question`** string into a list using the **`.split()`** method. You can put `' '` inside `.split()`. Make sure there is a space between the quotation marks. Save the output to a variable called **`word_list`**.

Let's write some code!

```
14  def generate_bot_answer_with_text_model(...):
15    bot_answer = None
16
17    word_list = user_question.split(' ')
18    random_word = random.choice(word_list)
19
20
```

3. On **Line 18**, create a variable called **`random_word`**. Set it equal to a random word from the word list, using the **`random.choice()`** method.

Let's write some code!

```
14  def generate_bot_answer_with_text_model(...):
15    bot_answer = None
16
17    word_list = user_question.split(' ')
18    random_word = random.choice(word_list)
19    bot_answer = text_model.make_sentence_with_start()
20
```

4. On **Line 19**, call the **`make_sentence_with_start()`** method on **`text_model`**. Save the output to **`bot_answer`**. .

Let's write some code!

```
14  def generate_bot_answer_with_text_model(...):
15    bot_answer = None
16
17    word_list = user_question.split(' ')
18    random_word = random.choice(word_list)
19    bot_answer = text_model.make_sentence_with_start(random_word,
    strict=False)
20
21    return bot_answer
```

5. Still on **Line 19**, we're going to add 2 parameters (inputs) to **`make_sentence_with_start()`**: **`random_word, strict=False`**

6. On **Line 21**, return **`bot_answer`**.

Let's write some code!

```
def generate_bot_answer_with_text_model(twitter_handle, user_question, text_model):
  bot_answer = none

  word_list = user_question.split(' ')
  random_word = random.choice(word_list)
  bot_answer = text_model.make_sentence_with_start(random_word, strict=False)

  return bot_answer
```

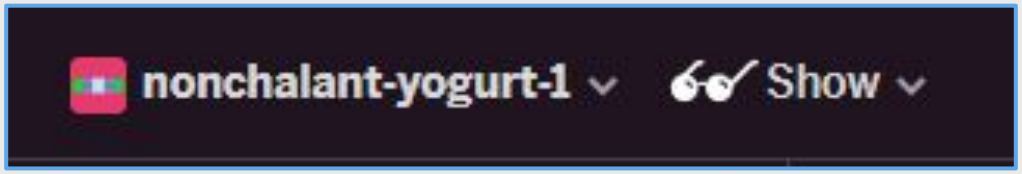
Woah! What did we just do?

- We used a function from Markovify called **`make_sentence_with_start()`** and called it on **`text_model`**.
- **`text_model`** is the output of feeding our celebrity's Tweets into Markovify.
- Now, we're asking Markovify to create a sentence, and we want it to base that sentence on a random word from the user's question.

Let's try it out!

Let's Try it Out!

In the top left hand corner of your browser, click **Show.**

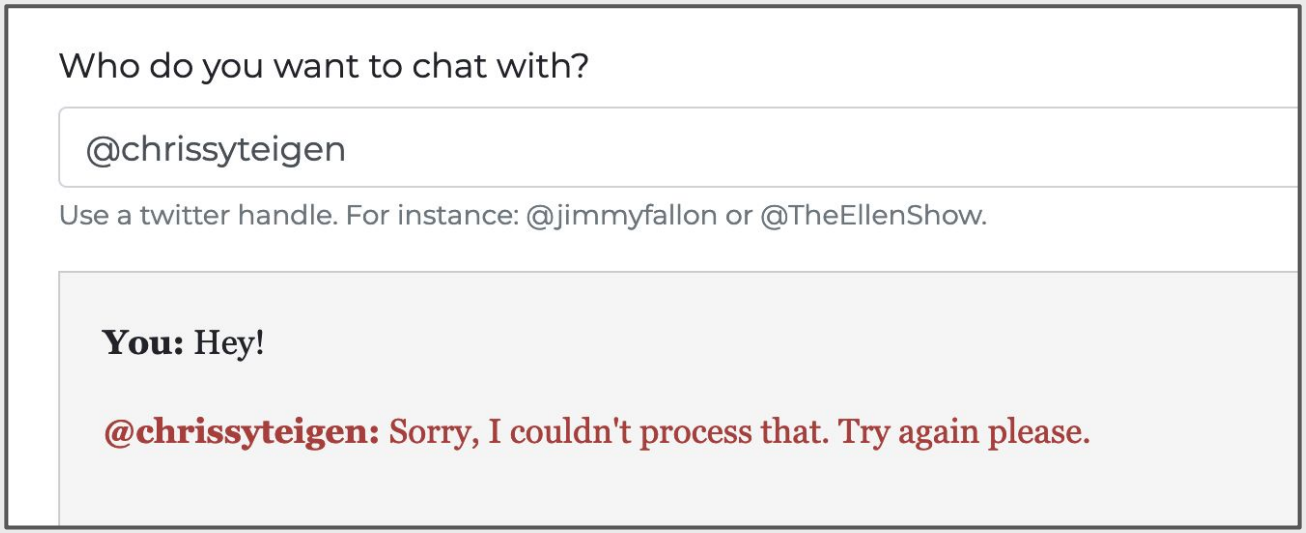


Try it in the browser tab that opens!

Woah! What went wrong?

Woah, what went wrong?

Did you get an error?



Let's investigate!

Let's investigate.

```
14  def generate_bot_answer_with_text_model(twitter_handle, user_question, text_model):
15    bot_answer = none
16
17    word_list = user_question.split(' ')
18    random_word = random.choice(word_list)
19    bot_answer = text_model.make_sentence_with_start(random_word, strict=False)
20    print("Print me if this function runs!")
21    return bot_answer
```

When you have bugs in your program, you can investigate by adding **`print()`** statements.

We want to make sure our **`generate_bot_answer_with_text_model()`** function is working.

Let's add a print statement on **Line 20** that says **`"Print me if this function runs!"`**

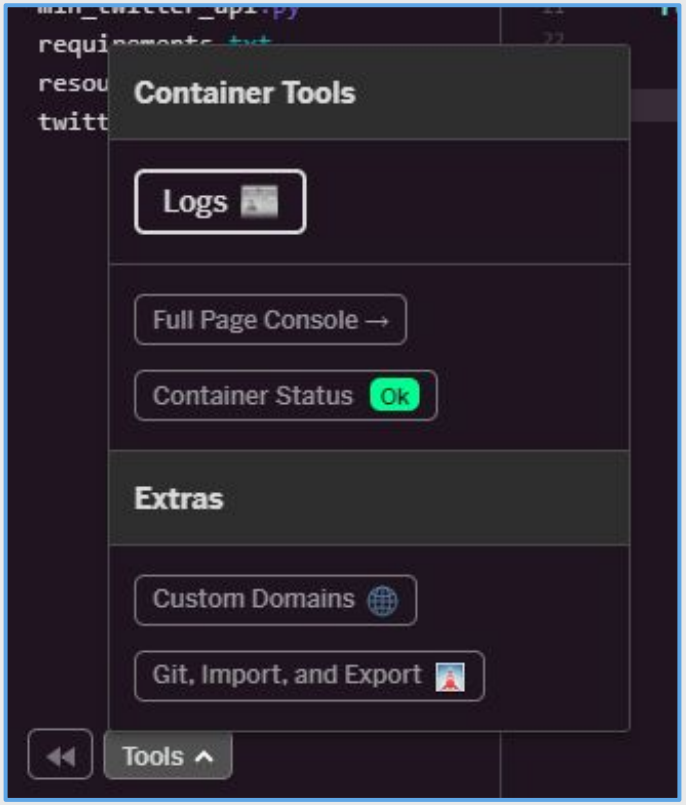
Let's investigate.

```
14  def generate_bot_answer_with_text_model(twitter_handle, user_question, text_model):
15    bot_answer = none
16
17    word_list = user_question.split(' ')
18    random_word = random.choice(word_list)
19    bot_answer = text_model.make_sentence_with_start(random_word, strict=False)
20    print("Print me if this function runs!")
21    return bot_answer
```

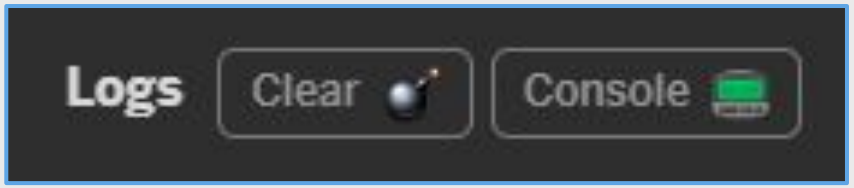
This will print in your Glitch logs. Let's open the logs!

Let's investigate.

1. At the bottom of the page, click **Tools** then **Logs**.

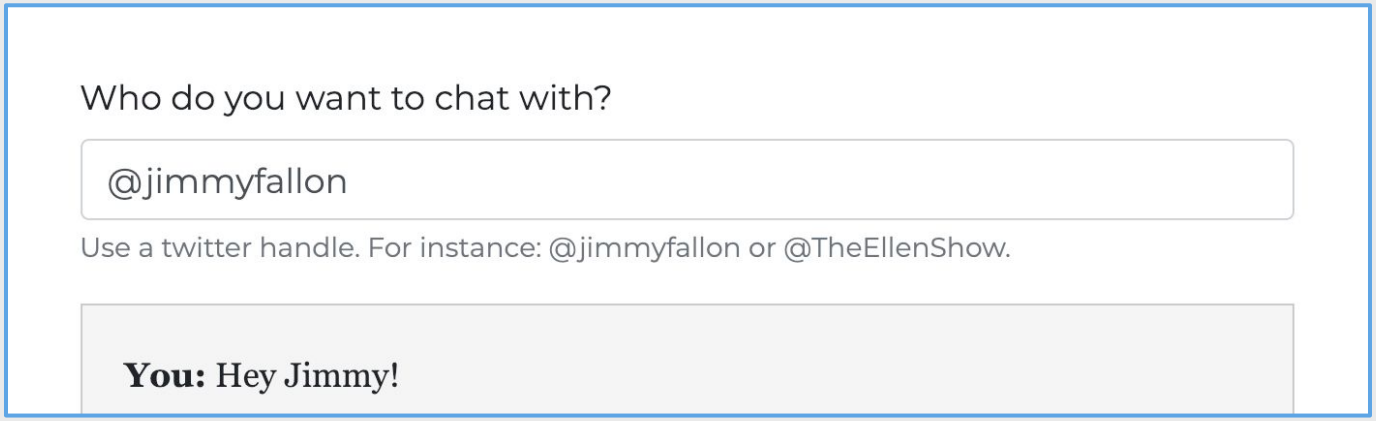


2. When the logs open, click **Clear**.

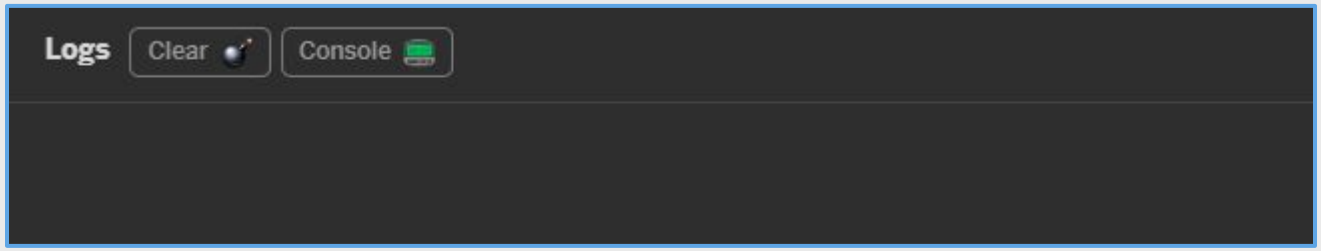


Let's Investigate!

Go back to the live app and try it again!

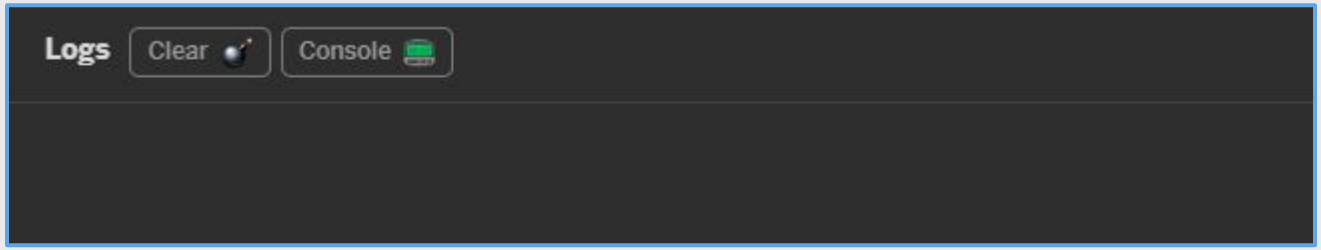


Then, go back to Glitch and read your logs. Did your statement print?



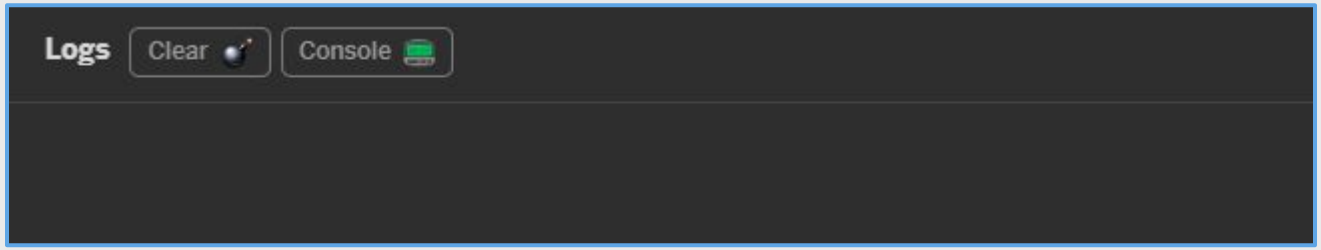
Let's Investigate!

Nothing is printing! Does anyone know why?



Let's Investigate!

Nothing is printing! Does anyone know why?



It's because we wrote the function, but never called it! Let's fix that now.

Let's write some code!

Let's go back to **`markov.py`**.

Notice that around **Line 9**, it says we should be using the markovify library? I don't see us using it anywhere! Let's consult the documentation.

```
8   # build the markov chain based on the text we read
9   # we use the markovify library to do this step
10  def generate_bot_answer(twitter_handle, user_question):
11      pass
12      # write code here
13
```

Markovify Documentation

Basic Usage

```
import markovify

# Get raw text as string.
with open("/path/to/my/corpus.txt") as f:
    text = f.read()

# Build the model.
text_model = markovify.Text(text)

# Print five randomly-generated sentences
for i in range(5):
    print(text_model.make_sentence())

# Print three randomly-generated sentences of no more than 140 characters
for i in range(3):
    print(text_model.make_short_sentence(140))
```

This is the "Basic Usage" documentation from Markovify. Remember, we use Markovify to generate a Markov Chain model. Which line of code do you think is doing that? We need to add it to our code!

Markovify Documentation

Basic Usage

```
import markovify

# Get raw text as string.
with open("/path/to/my/corpus.txt") as f:
    text = f.read()

# Build the model.
text_model = markovify.Text(text)

# Print five randomly-generated sentences
for i in range(5):
    print(text_model.make_sentence())

# Print three randomly-generated sentences of no more than 140 characters
for i in range(3):
    print(text_model.make_short_sentence(140))
```

That's the one! Let's add it to our function.

Let's write some code!

```
10  def generate_bot_answer(twitter_handle, user_question):
11      tweets = get_user_tweets(twitter_handle)
12
13
14
15
16
17
18
19
```

1. On **Line 11**, delete **`pass`**.
2. Then, use the **`get_user_tweets()`** function to save tweets to a variable called **`tweets`**.

Let's write some code!

```
10  def generate_bot_answer(twitter_handle, user_question):
11      tweets = get_user_tweets(twitter_handle)
12      clean_tweets = clean_tweets_data(tweets)
13      text = ''.join(map(str, clean_tweets))
14
15
16
17
18
19
```

3. On **Line 12**, use the **`clean_tweets_data()`** function to clean the Tweets. Save this to a variable called **`clean_tweets`**.

4. On **Line 13**, use the **`join()`** function to convert the clean tweets into a string. Save this to a variable called **`text`**.

Let's write some code!

```
10  def generate_bot_answer(twitter_handle, user_question):
11      tweets = get_user_tweets(twitter_handle)
12      clean_tweets = clean_tweets_data(tweets)
13      text = ''.join(map(str, clean_tweets))
14      text_model = markovify.Text(text)
15
16
17
18
19
```

5. On **Line 14**, use the **`.Text()`** method from the markovify library. Save the output to a variable called **`text_model`**. What should we pass to **`.Text()`**?

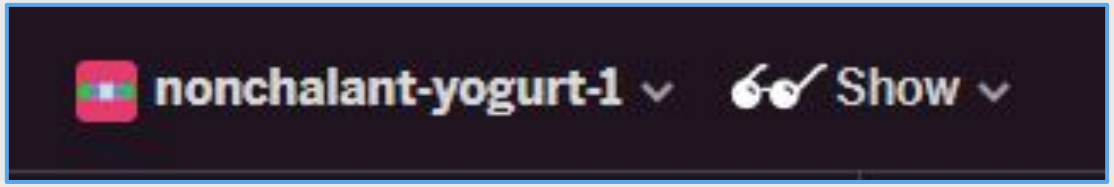
Let's write some code!

```
10  def generate_bot_answer(twitter_handle, user_question):
11      tweets = scrape(twitter_handle)
12      clean_tweets = clean_tweets_data(tweets)
13      text = ''.join(map(str, clean_tweets))
14      text_model = markovify.Text(text)
15      bot_answer =
16  generate_bot_answer_with_text_model(twitter_handle, user_question,
17  text_model)
18      return bot_answer
19
```

6. Now that we have created the text model, we need to call the function we wrote before! Call it on **Line 16** in a return statement and then **return bot_answer**

Let's Try it Out!

In the top left hand corner of your browser, click **Show Live**



Try it in the browser tab that opens!

Let's Investigate!

Your statement still didn't print. That's because we also have to edit `main.py`!

```python
from flask import Flask, render_template
from flask_socketio import SocketIO
from twitter_scraper_fetcher import *
from mlh_twitter_api import moderate
import json
import config
import random

app = Flask(__name__)
socketio = SocketIO(app)

# Renders UI
@app.route("/")
def home():
  return render_template("homepage.html")

# Chat API - WebSocket
@socketio.on("send question")
def generate_message(body, methods=["POST"]):
  question = body["message"]
  twitter_handle = body["username"]

  # Call get_user_tweets() from twitter_scraper_fetcher.py to scrape some tweets
  tweets = get_user_tweets(twitter_handle)
  try:
    cleaned_tweets = clean_tweets_data(tweets)
    # Get a random tweet from the list of tweets
    bot_answer = random.choice(cleaned_tweets)
    bot_answer = moderate(bot_answer)

    # Send the answer to the app, to display to the user
    answer = {"username": twitter_handle, "message": bot_answer}
    socketio.emit("bot answer", answer)
  except:
    bot_answer = "Sorry, I couldn't process that. Try again please."
    socketio.emit("error", {"username": twitter_handle, "message": bot_answer})

if __name__ == "__main__":
    socketio.run(app)
```

Let's write some code!

1. Let's import **`markov.py`** into **`main.py`** so we can use our generate bot answer function! It should look like **Line 8.**

```
1   from flask import Flask, render_template
2   from flask_socketio import SocketIO
3   from twitter_scraper_fetcher import *
4   from content_moderator import moderate
5   import json
6   import config
7   import random
8   from markov import *
```

Let's write some code!

2. Now, scroll down to **Line 25**.
3. On line 25, call **`generate_bot_answer()`** and save it's output to a variable called **`bot_answer`**.
4. Add the following code to line 26:

```
23      try:
24          # Use our new functions to get tweets!
25          bot_answer = generate_bot_answer(twitter_handle, question)
26          moderated_answer = moderate(bot_answer)
```

We have a little more work to do!
You've got this!

Let's write some code!

1. Return to **`markov.py`**. Add the code below:

```
def generate_bot_answer_with_text_model(...):
  bot_answer = None

  word_list = user_question.split(' ')
  random_word = random.choice(word_list)
  bot_answer = text_model.make_sentence_with_start(random_word, strict=False)

  if bot_answer == None:
      bot_answer = text_model.make_sentence(test_output=False)

  return bot_answer
```

Now it should work every time!

Let's try it out!

Try again!

Go back to the live app and try it again!

Who do you want to chat with?

@jimmyfallon

Use a twitter handle. For instance: @jimmyfallon or @TheEllenShow.

You: This is seriously cool!

@jimmyfallon: Plus, a new take on a gift I received from my agents at CAA.

You did it!

But! We're going to add one more thing.

AI requires a huge data set!

In order for Markovify to generate really good responses, we want to use the Twitter API instead of scraping for Tweets.

1. Change line 33 (it might have moved) to say **`fetch`** instead of **`get_user_tweets`**.

```
30  def generate_bot_answer(twitter_handle, user_question):
31      tweets = fetch(twitter_handle)
32      clean_tweets = clean_tweets_data(tweets)
33      text = ''.join(map(str, clean_tweets))
34      text_model = markovify.Text(text)
35
36      return generate_bot_answer_with_text_model(twitter_handle,
    user_question, text_model)
37
38
```

Now try your app again, and enjoy your AI powered Twitter conversations!

Table of Contents

What did you learn today?

We created a fun quiz to test your knowledge and see what you learned from this workshop.

http://mlhlocal.host/quiz

Table of Contents

Next Steps

Where to go from here...

1. Take a course on Codecademy or Udacity.
2. Try out Code Wars
3. Stay tuned for more opportunities from Capital One!

MLH localhost

Workshop

Basic Training: Intro to Python Skills for AI, Part III

v.0.0.1