a workshop by deltaonealpha

Connecting Python & Telegram

Hmm... Telegram... Python?

Yes, Telegram. It is one of the world's leading messaging apps.

Well, "one of the leading apps". Why Telegram of all?

Well, for starters, it is one of the very few messaging companies that provide open-source API access to everyone. All of Telegram (except the server software) is completely open source!

In fact, the official Telegram app on mobile is rather called a 'Telegram client'. As its open source, there are literally a gazillion Telegram clients out there.

You don't like your client? Code one yourself, it is that flexible!

Python... bruh?

Why would one even connect the two?

Python is a leading language today. Being extremely flexible and one of the easiest languages to learn, Python has a lot going for it, and connecting the just makes a lot of sense.

Let me give you an example.

My father's friend lost a lot of money because his restaurant's manager logged meagre sales and pocketed much of the profit.

On hearing this, I started to think, what if every purchase got logged on its own, in a place inaccessible for everyone but the owner?

Pondering on a way to solve this, I created DBFA, a scalable billing framework written in Python3. It can manage your store, employees, payments, inventory, orders, deliveries, invoicing, and a lot more.

A core feature of DBFA is complete Telegram logging. As soon as a bill is issued, it immediately logs the sales activity in the store owner's Telegram account as a message from a bot, with all communications over TSL-encrypted HTTPS. Not only this, it even logs every event where store data is being accessed.

Expanding on this, I even added inline button-based two-factor-authentication as text inputs can be super manipulative.

Yeah, all using Telegram and Python! What I did is nothing compared to what you all can do. Yeah, **you!**

Sounds nice?

Let's plan this workshop then!

Requisites:

- Basic Python skills.
- A Telegram account.
- pip install requests

Topics:

- Creating a Telegram bot
- A bit on Telegram's WEB BOT API
- Connecting and sending a message
  - Understanding the code
- Receiving messages from the bot
  - Understanding the code
- A small project combining all this
- **EXTRA RESOURCE:** Marking received messages as read and a dive into BOT API v2 and inline stuff!

NOTE: *This workshop will be a quick one, so that it can be made quickly without getting boring.*

For the inquisitive ones out there, there will be some references and resources attached at the end so you can tinker a bit more.

BOTS EVERYWHERE

Creating a bot in Telegram!

Telegram has these 'bots', or simple chats which can be handled autonomously. They're basically a way for developers and service providers to automate interactions.

To interface with a bot, we first need to create one, and get its credentials.

Making a bot is quite easy. Ironically enough, you create bots via a bot called **'BotFather'.**

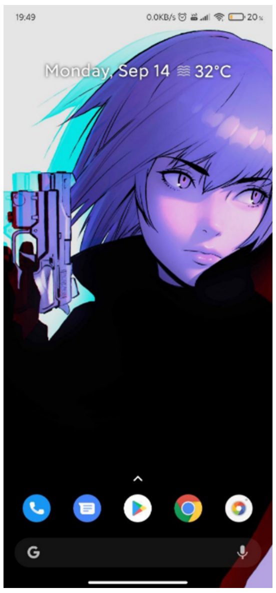
https://t.me/BotFather

Its Telegram's official bot.

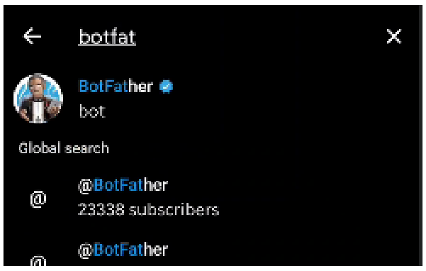
(Please turn over)

Let us summarize this into some quick steps:

- **Get your phone ready**



- **Open Telegram and search for 'BotFather'**


botfat

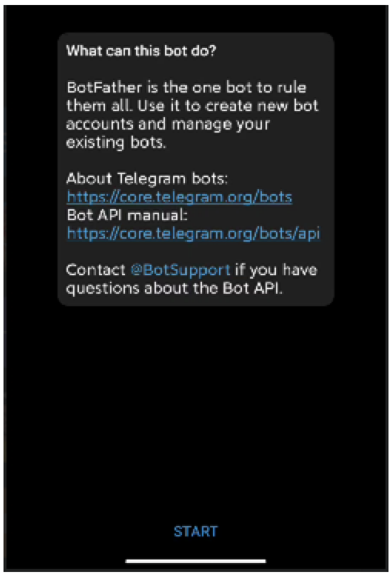
BotFather
bot

Global search

@BotFather
23338 subscribers

@BotFather

- **Now hit 'START'**


What can this bot do?

BotFather is the one bot to rule them all. Use it to create new bot accounts and manage your existing bots.

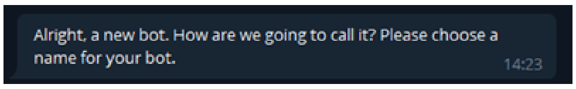
About Telegram bots:
https://core.telegram.org/bots
Bot API manual:
https://core.telegram.org/bots/api

Contact @BotSupport if you have questions about the Bot API.

- **Now type:**

```
/newbot
```

Reply from BotFather:

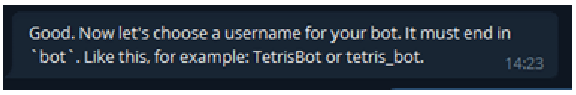

Alright, a new bot. How are we going to call it? Please choose a name for your bot. 14:23

"Alright, a new bot. How are we going to call it? Please choose a name for your bot."

- **Now type the name for your bot (display name):**

```
EXAMPLE:  FartingSpider Reborn
```

Reply from BotFather:


Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 14:23

"Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: `'TetrisBot'` or `'tetris_bot'`."

- **Now reply with your desired username for your bot:**

NOTE: *Your bot's username must end in 'bot' or '_bot'*

`Delta` ✗

`Delta bot` ✗

`Deltabot`	✅
`Delta_bot`	✅

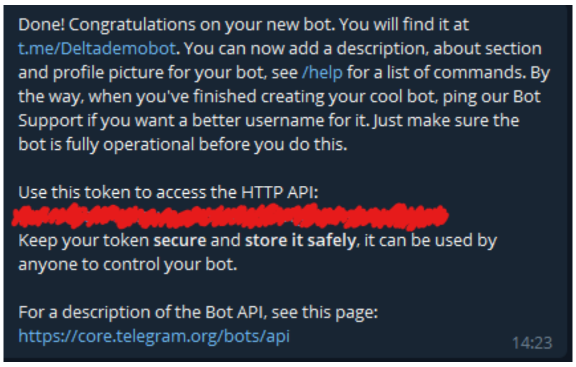
- **Reply from BotFather:**

  !!!!

  NOTE: *The HTTP API key provided to you by BotFather should be kept a secret.*

  API keys are extremely crucial. If it gets out, anyone and everyone can send you messages via the bot, misuse the bot and even delete it!

  Even I've doodled over my API key!



Done! Congratulations on your new bot. You will find it at t.me/Deltademobot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
[redacted]
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page:
https://core.telegram.org/bots/api
14:23

That's it! You just created your very first Telegram bot!

SENDING MESSAGES

Time to experiment!

Telegram's WEB API is remarkably simple, yet powerful, allowing you to essentially send and retrieve messages just by visiting a URL!

We will now try out an example:

- Randomly ping this bot on Telegram: https://t.me/get_id_bot

  and save the 'chat ID' it gives you.
- Then, head over here: **https://repl.it/@deltaonealpha/BasicTelegramBotSendText**

Try running this repl.it by entering your bot's access token and the chat ID you received by pinging the *get_id* bot.

```
--
*get_id* is a wonderfull bot created by Fredy Kardian Udo (t.me/fredykardian)
--
```

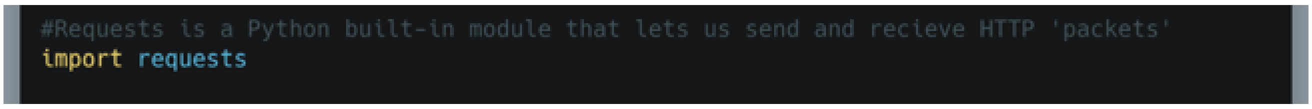
When you are done with trying this out, turn over.

DISECTING THE CODE

Now that we have experienced how seamlessly the WEB API works (I hope you did not skip), let us trace down every line:

This is *requests*, an alternative to Python's built-in URLLIB. This needs to be installed via pip (command in workshop requisites section) if you're running this off your own PC.

Running this code on repl.it doesn't require you to pip.



```python
#Requests is a Python built-in module that lets us send and recieve HTTP 'packets'
import requests
```

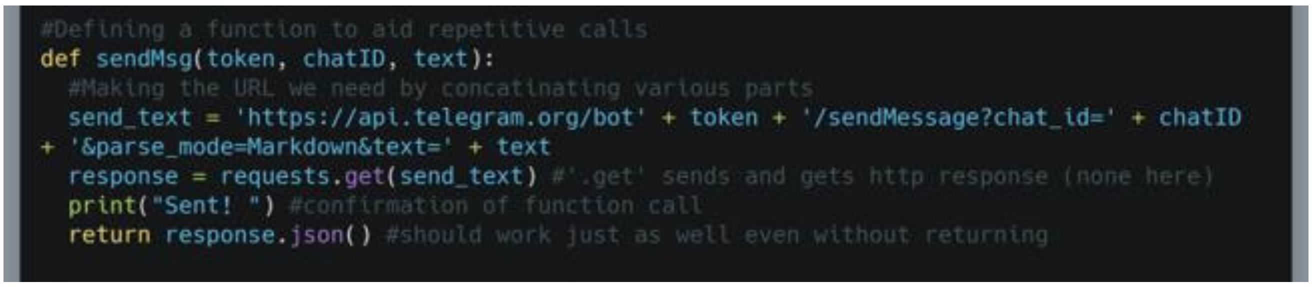
Now we define a function, *sendMsg()*. To call this function, we will need to pass three parameters with it, your bot's token, your chat ID and the text you need to send. As simple as that!

send_text concatenates your token, chat ID and message with the base URL of Telegram's API. This includes the method *sendMessage*, which is used to... well, send messages.

Now we define and call a variable, *response*, which allows us to use the *requests* module's *get* function to access the URL we stored in the variable *send_text*, in the above step.

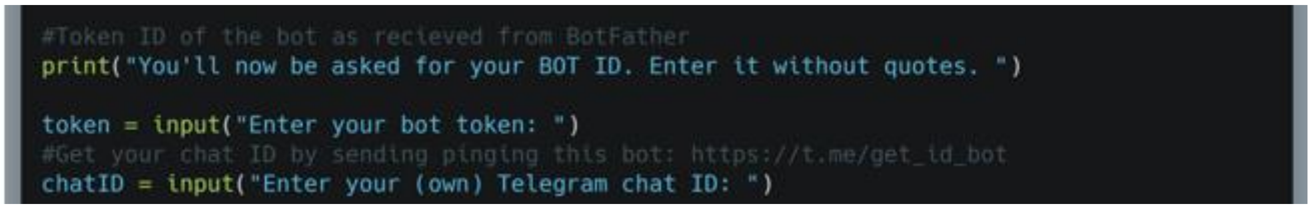
Now we just return *response.json* in an effort to improve our code and unify things.

And.... that's it!



```python
#Defining a function to aid repetitive calls
def sendMsg(token, chatID, text):
  #Making the URL we need by concatinating various parts
  send_text = 'https://api.telegram.org/bot' + token + '/sendMessage?chat_id=' + chatID
+ '&parse_mode=Markdown&text=' + text
  response = requests.get(send_text) #'.get' sends and gets http response (none here)
  print("Sent! ") #confirmation of function call
  return response.json() #should work just as well even without returning
```

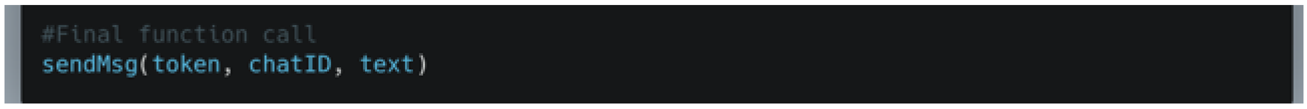
What follows is just a bunch of *input()* statements to receive values from the user to pass-on to the function.



```python
#Token ID of the bot as recieved from BotFather
print("You'll now be asked for your BOT ID. Enter it without quotes. ")

token = input("Enter your bot token: ")
#Get your chat ID by sending pinging this bot: https://t.me/get_id_bot
chatID = input("Enter your (own) Telegram chat ID: ")
```

Then we create a text variable with out text, and simple pass it to the function!



```python
#Final function call
sendMsg(token, chatID, text)
```

RECEIVING MESSAGES

Remember using *sendMessage* in the URL when sending a message via your bot?

To receive messages, we use *getUpdates*. Telegram handles receiving messages brilliantly! Whenever you call *getUpdates* with your bot token, it returns a .JSON with all your queued messages.

Why do I say "queued"? Because your messages don't clear till you issue a "receipt" to Telegram of the same.

Each message comes with an *updateID*, which you can return to Telegram to clear that from the queue. For example, if you have 10 messages, simply passing the *updateID* of the latest message clears all 10 messages.

Yeah, this might sound complex, but this small demonstration on repl.it should make it seem all easy-peasy!

Send a few texts to your BOT via Telegram, then visit this link, and enter your BOT's token:

https://repl.it/@deltaonealpha/TelegramGetUpdates#main.py

Worked marvellously, right?

Code dissection follows.

DISECTING THE CODE

Again, we import *requests* and *json*. Since *json* is a part of Python's standard library, you won't need to install it when using this code on your local installation.

```
import requests, json
```

Now we take an input for the BOT token, and concatenate it with the required URL, issuing *getUpdates*. Then we define a variable *updates*, as a .JSON returned by requests.

```
token = input("Enter your complete bot token: ")
url = f'https://api.telegram.org/bot{token}/getUpdates'
updates = requests.post(url).json()
```

And now? We simple slice the text out, and print!

```
print("\nText recieved:\n--------------")
for update in updates["result"]:
  print(update["message"]["text"])
```

And... that's it!

Its PROJECT TIME!!

Now since we are all aware of basic Telegram BOT controls, why not make a 5-minute project integrating all this?

In this mini-project, we will be asking the user for their chat ID and bot token, and telling them to send a “hello” to our bot. If the bot detects “hello”, it will respond with a greeting and if it doesn’t, it will send a different reply ;)

Let’s start!

We start by importing *requests* and *json* to handle sending and receiving messages, and then *time* to include delays in our code.

```
import requests, json, time
```

Now we take inputs from the user:

```
token = input("Enter your bot token: ")

#Get your chat ID by sending pinging this bot: https://t.me/get_id_bot
chatID = input("Enter your (own) Telegram chat ID: ")
```

Now, the user is instructed to send “hello” to their Telegram bot and then enter any key to proceed.

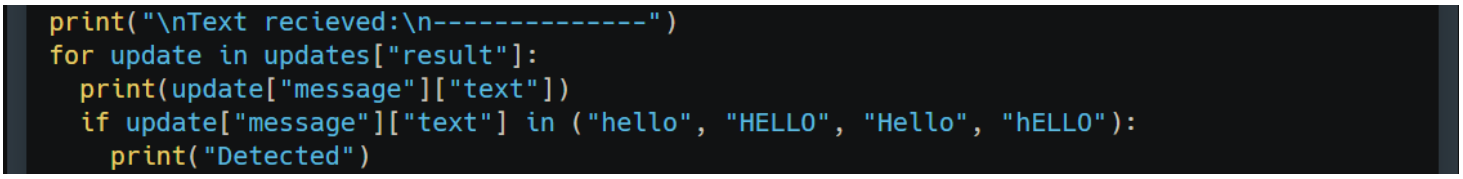
(We have to wait for a confirmation else the code will immediately proceed to replying when the user hasn’t even sent anything :D)

```
print("Now open your Telegram application and send 'hello' to your Telegram Bot.")
time.sleep(5) #Waiting
waitkey = input("Input any *key* once done.")
```

Now, we simply concatenate the inputs into a URL and define a variable *updates* as a request to this URL:

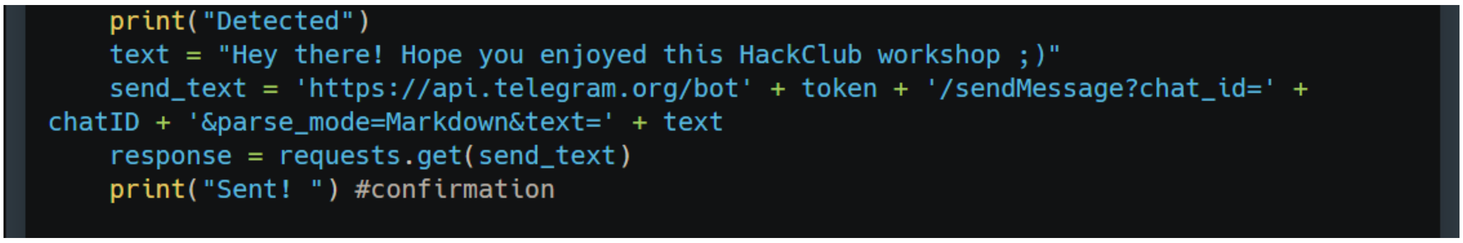
```
url = f'https://api.telegram.org/bot{token}/getUpdates'
updates = requests.post(url).json()
```

Then we print the slice the .JSON received, print the received text and put it in an *if...else* block to compare it and detect whether the message is "hello" (or its case variations) or something entirely different!



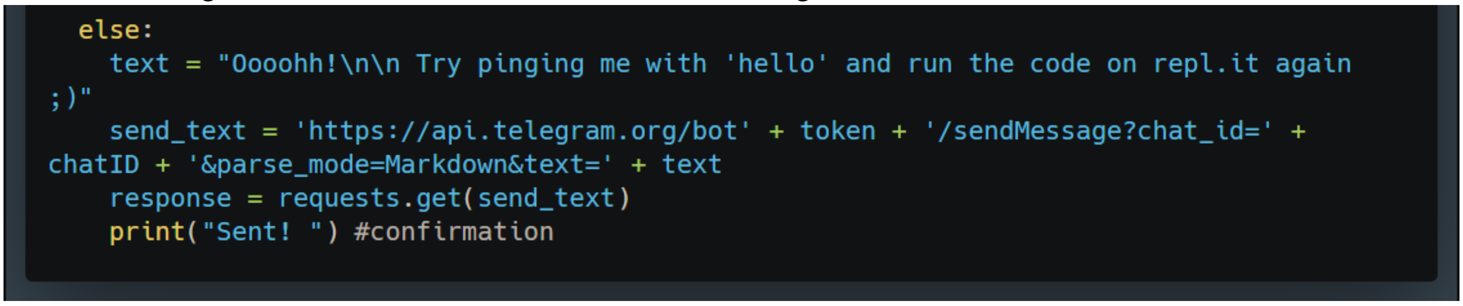
```
print("\nText recieved:\n--------------")
for update in updates["result"]:
  print(update["message"]["text"])
  if update["message"]["text"] in ("hello", "HELLO", "Hello", "hELLO"):
    print("Detected")
```

If the message is *"hello"*, we send a greeting back!



```
    print("Detected")
    text = "Hey there! Hope you enjoyed this HackClub workshop ;)"
    send_text = 'https://api.telegram.org/bot' + token + '/sendMessage?chat_id=' +
chatID + '&parse_mode=Markdown&text=' + text
    response = requests.get(send_text)
    print("Sent! ") #confirmation
```

If the message is not "hello", we still send a message back, but a different one:



```
  else:
    text = "Oooohh!\n\n Try pinging me with 'hello' and run the code on repl.it again
;)"
    send_text = 'https://api.telegram.org/bot' + token + '/sendMessage?chat_id=' +
chatID + '&parse_mode=Markdown&text=' + text
    response = requests.get(send_text)
    print("Sent! ") #confirmation
```

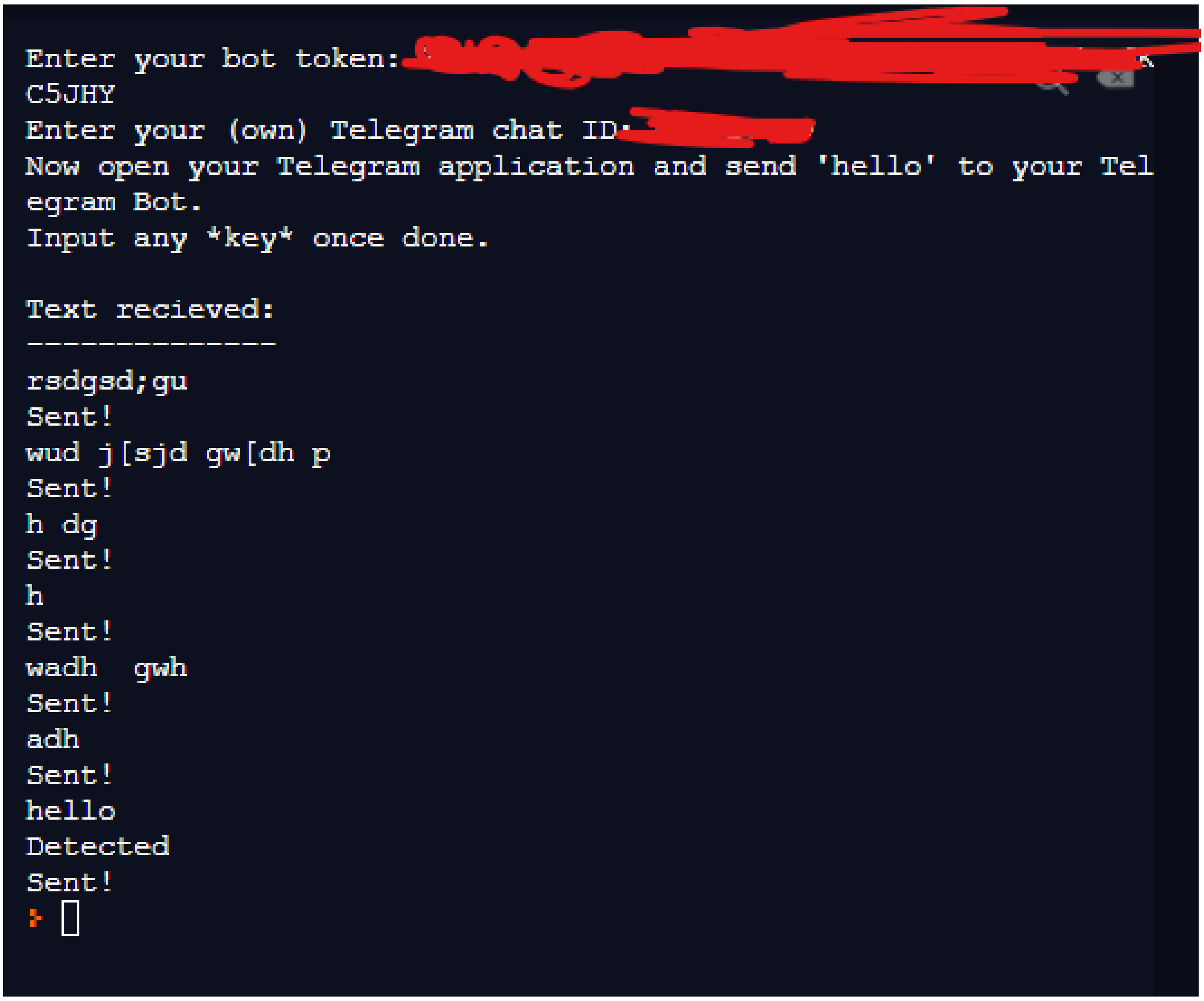
And... that's it!

P.S. You can send multiple messages, one *"hello"*, and maybe one not that, and the bot will reply to each!

You can try this on a repl.it if you feel too lazy to code:
https://repl.it/@deltaonealpha/TelegramWorkshopSampleProject#main.py

(Images on the next page)

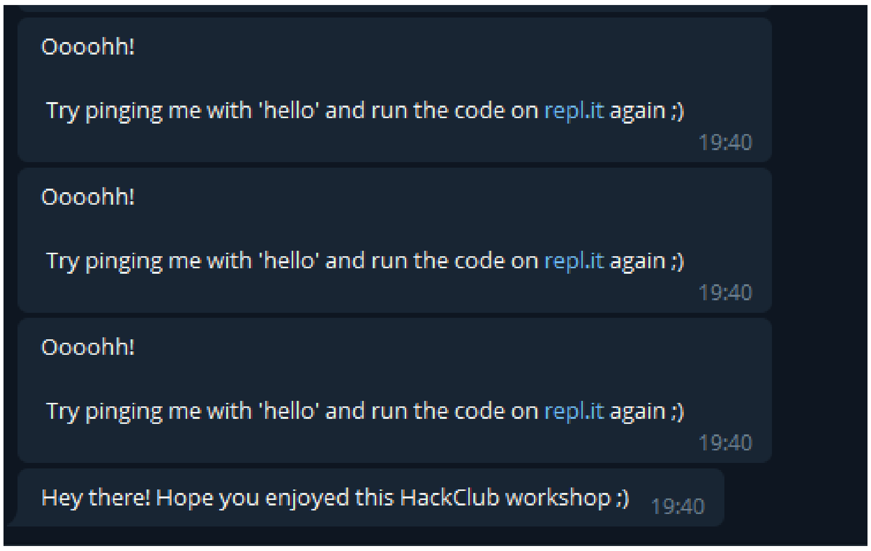
repl.it shell



```
Enter your bot token:[redacted]
C5JHY
Enter your (own) Telegram chat ID:[redacted]
Now open your Telegram application and send 'hello' to your Tel
egram Bot.
Input any *key* once done.

Text recieved:
---------------
rsdgsd;gu
Sent!
wud j[sjd gw[dh p
Sent!
h dg
Sent!
h
Sent!
wadh  gwh
Sent!
adh
Sent!
hello
Detected
Sent!
> 
```

Telegram chat window



```
Oooohh!

 Try pinging me with 'hello' and run the code on repl.it again ;)
                                                            19:40
Oooohh!

 Try pinging me with 'hello' and run the code on repl.it again ;)
                                                            19:40
Oooohh!

 Try pinging me with 'hello' and run the code on repl.it again ;)
                                                            19:40
Hey there! Hope you enjoyed this HackClub workshop ;)   19:40
```

Goodbye!

That was all for this workshop. This is the first ever workshop that I've made in my life and the first time when I've tried to teach something to such a wide and diverse community!

I really hope this workshop was fun. If you have any doubts, contact me on HackClub's Slack: `@deltaonealpha` (https://hackclub.slack.com/team/U01AVFQUCAD)

(P.S For those who'd like to stick around longer, there are a few external resources linked below as a bonus, which allow you to make this part of something even bigger! Hack on!)

Bonus Resources

TIME TO CALL HIM

While our code works marvellously, we still have a major problem to fix before we move on to making this a part of something big.

Issuing Read Receipts to Telegram

If you don't clear messages received from *getUpdates*, they'll keep coming alongside newer messages when you make subsequent calls. To avoid this, we just have to let Telegram know that we've received the message(s).

We pass the *updateId* with of the latest received message while using *getUpdates*, and Telegram send us only subsequent messages.

The *updateID* can be found in the .JSON returned and can be used after slicing.

Resources:

https://www.serverless.com/examples/aws-node-telegram-echo-bot

https://www.andreafortuna.org/2017/11/29/how-to-build-a-simple-echo-bot-on-telegram-using-hook-io-and-python/

https://www.codementor.io/@garethdwyer/building-a-telegram-bot-using-python-part-1-goi5fncay

https://blog.usejournal.com/part-1-how-to-create-a-telegram-bot-in-python-under-10-minutes-145e7f4e6e40

Leveraging BOT API v2: Inline buttons and keyboard!

Resources:

https://python-telegram-bot.readthedocs.io/en/stable/telegram.inlinekeyboardmarkup.html

https://github.com/python-telegram-bot/python-telegram-bot/blob/master/examples/inlinekeyboard.py

https://python-telegram-bot.readthedocs.io/en/stable/telegram.inlinekeyboardbutton.html

https://www.mindk.com/blog/how-to-develop-a-chat-bot/