Lesson 5 –
Sensor data upload to cloud & notification using Python

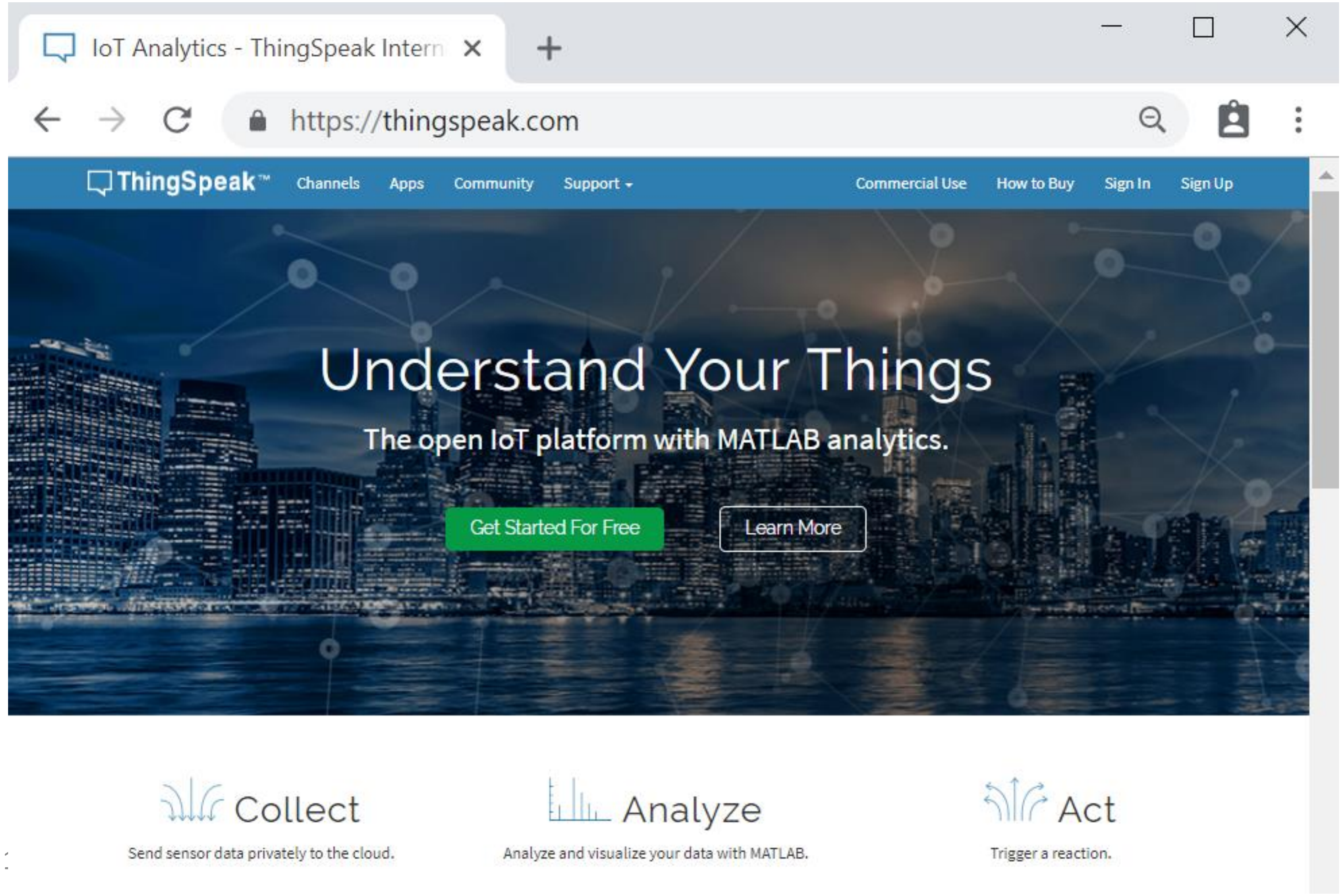
- S.P. Chong

Objectives

- In this lesson, you will learn to set up a **Cloud** platform, such as **Thingspeak**, for sensor data to be uploaded.
- You will then learn to program a RPi (Raspberry Pi) using Python, to **send data** from a sensor to (and to **read data** from) a Cloud platform such as Thingspeak.
- You will next learn to send an event-triggered **notification**, through **Twitter**, for instance.
- Finally, you will learn to use **sockets** for RPi-RPi communication.

Setting up a Cloud platform (e.g. Thingspeak)

- Let’s see how we can set up **Thingspeak**, a cloud platform for uploading sensor data.



Understand Your Things

The open IoT platform with MATLAB analytics.

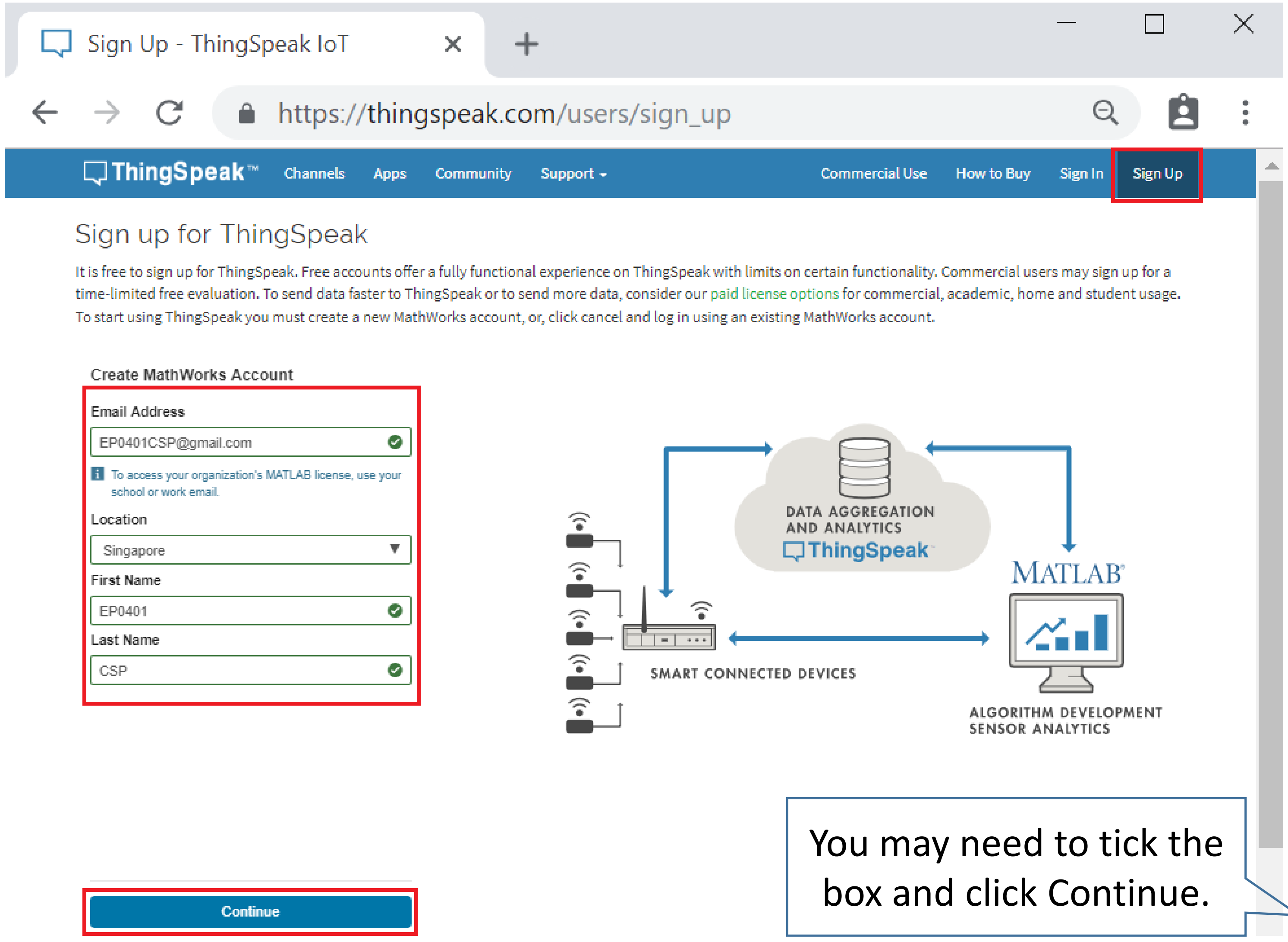
Send sensor data privately to the cloud.

Analyze and visualize your data with MATLAB.

Trigger a reaction.

You can create a Thingspeak account at www.thingspeak.com

Setting up a Cloud platform (e.g. Thingspeak) (cont.)


Sign Up - ThingSpeak IoT

https://thingspeak.com/users/sign_up

ThingSpeak™ Channels Apps Community Support Commercial Use How to Buy Sign In Sign Up

Sign up for ThingSpeak

It is free to sign up for ThingSpeak. Free accounts offer a fully functional experience on ThingSpeak with limits on certain functionality. Commercial users may sign up for a time-limited free evaluation. To send data faster to ThingSpeak or to send more data, consider our paid license options for commercial, academic, home and student usage. To start using ThingSpeak you must create a new MathWorks account, or, click cancel and log in using an existing MathWorks account.

Create MathWorks Account

Email Address: EP0401CSP@gmail.com

To access your organization's MATLAB license, use your school or work email.

Location: Singapore

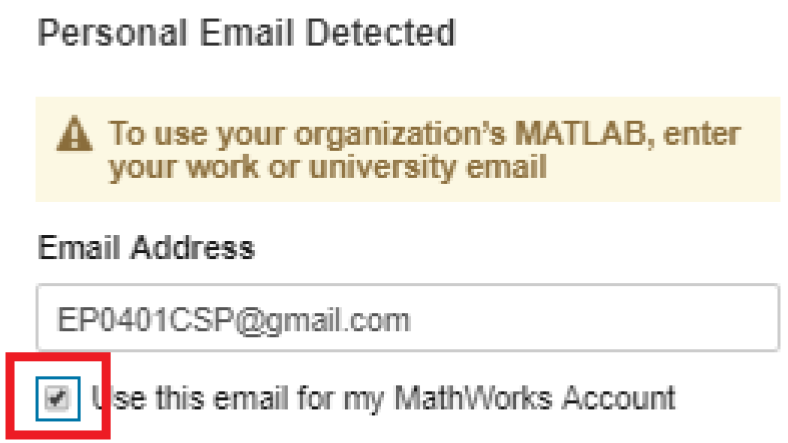
First Name: EP0401

Last Name: CSP

Continue

Click Sign Up, fill in the Email Address etc., and click Continue.

You may need to tick the box and click Continue.

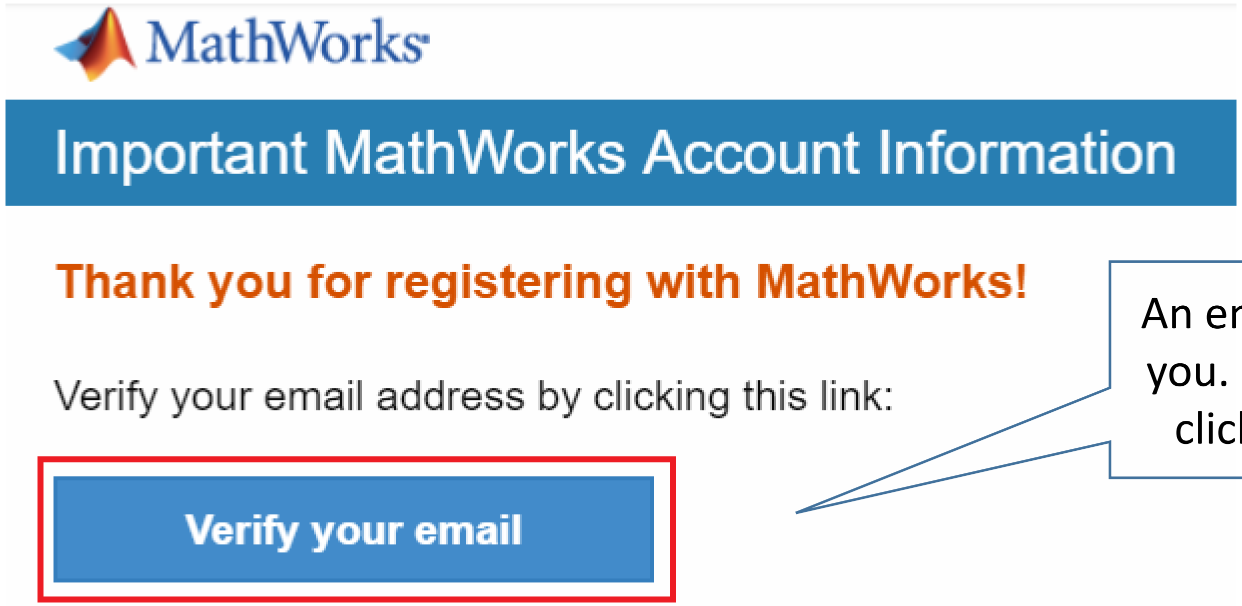

Personal Email Detected

To use your organization's MATLAB, enter your work or university email

Email Address: EP0401CSP@gmail.com

[✓] Use this email for my MathWorks Account

Setting up a Cloud platform (e.g. Thingspeak) (cont.)



MathWorks

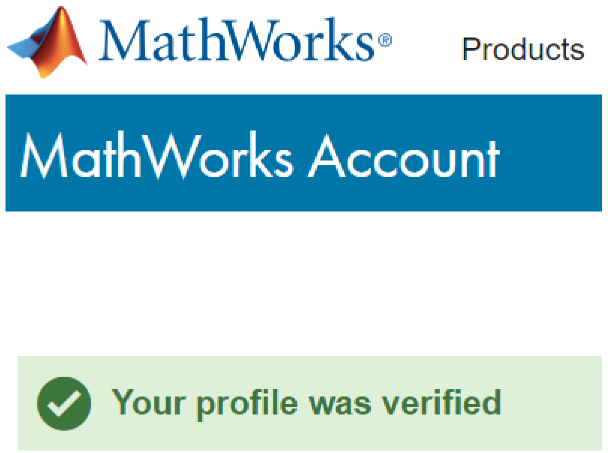
Important MathWorks Account Information

Thank you for registering with MathWorks!

Verify your email address by clicking this link:

Verify your email

An email has been sent to you. Open that email and click Verify your email.



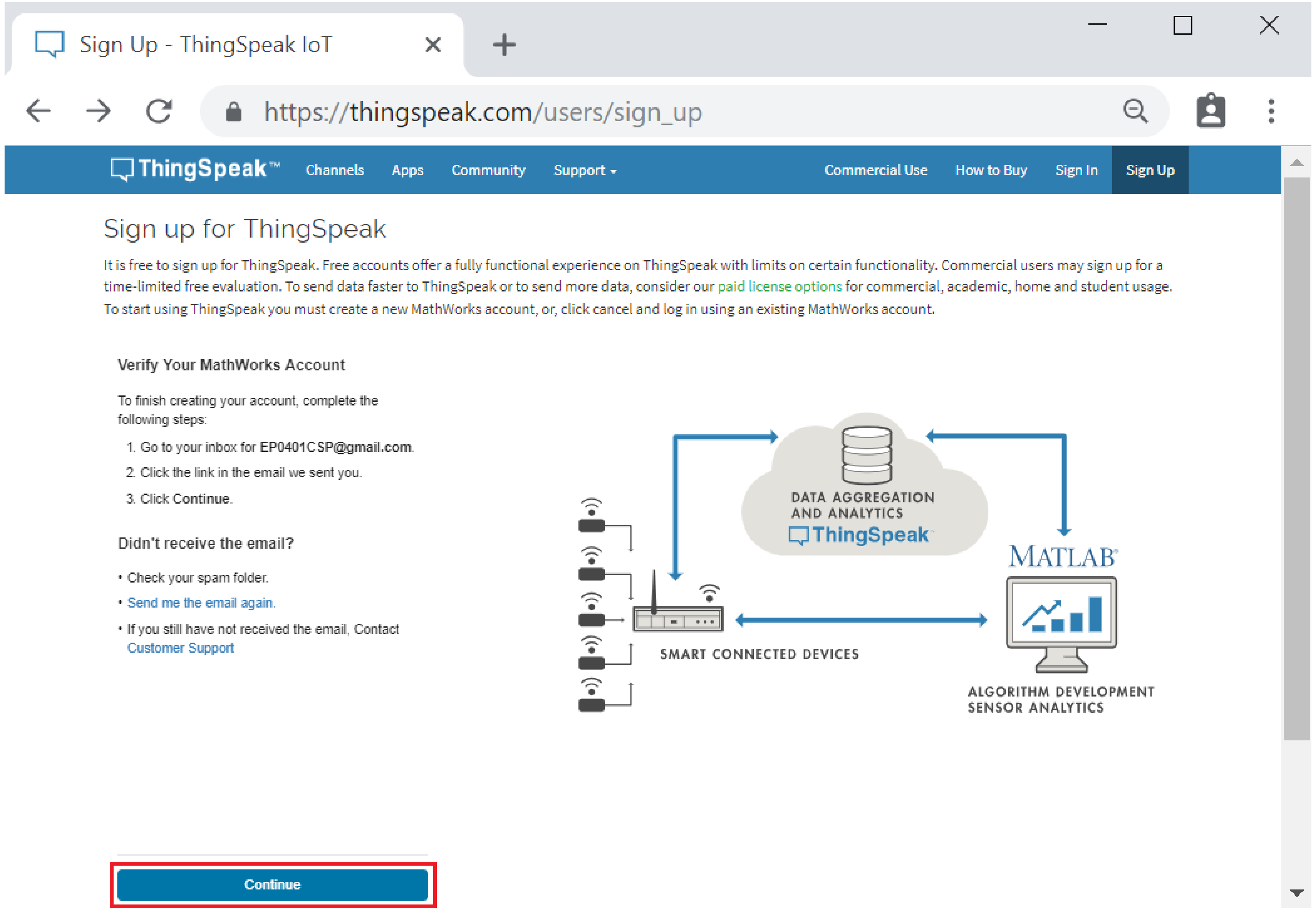
MathWorks® Products

MathWorks Account

Your profile was verified

A new tab will open to inform you "Your profile was verified".

Setting up a Cloud platform (e.g. Thingspeak) (cont.)



Sign up for ThingSpeak

It is free to sign up for ThingSpeak. Free accounts offer a fully functional experience on ThingSpeak with limits on certain functionality. Commercial users may sign up for a time-limited free evaluation. To send data faster to ThingSpeak or to send more data, consider our paid license options for commercial, academic, home and student usage. To start using ThingSpeak you must create a new MathWorks account, or, click cancel and log in using an existing MathWorks account.

Verify Your MathWorks Account

To finish creating your account, complete the following steps:

1. Go to your inbox for EP0401CSP@gmail.com.
2. Click the link in the email we sent you.
3. Click Continue.

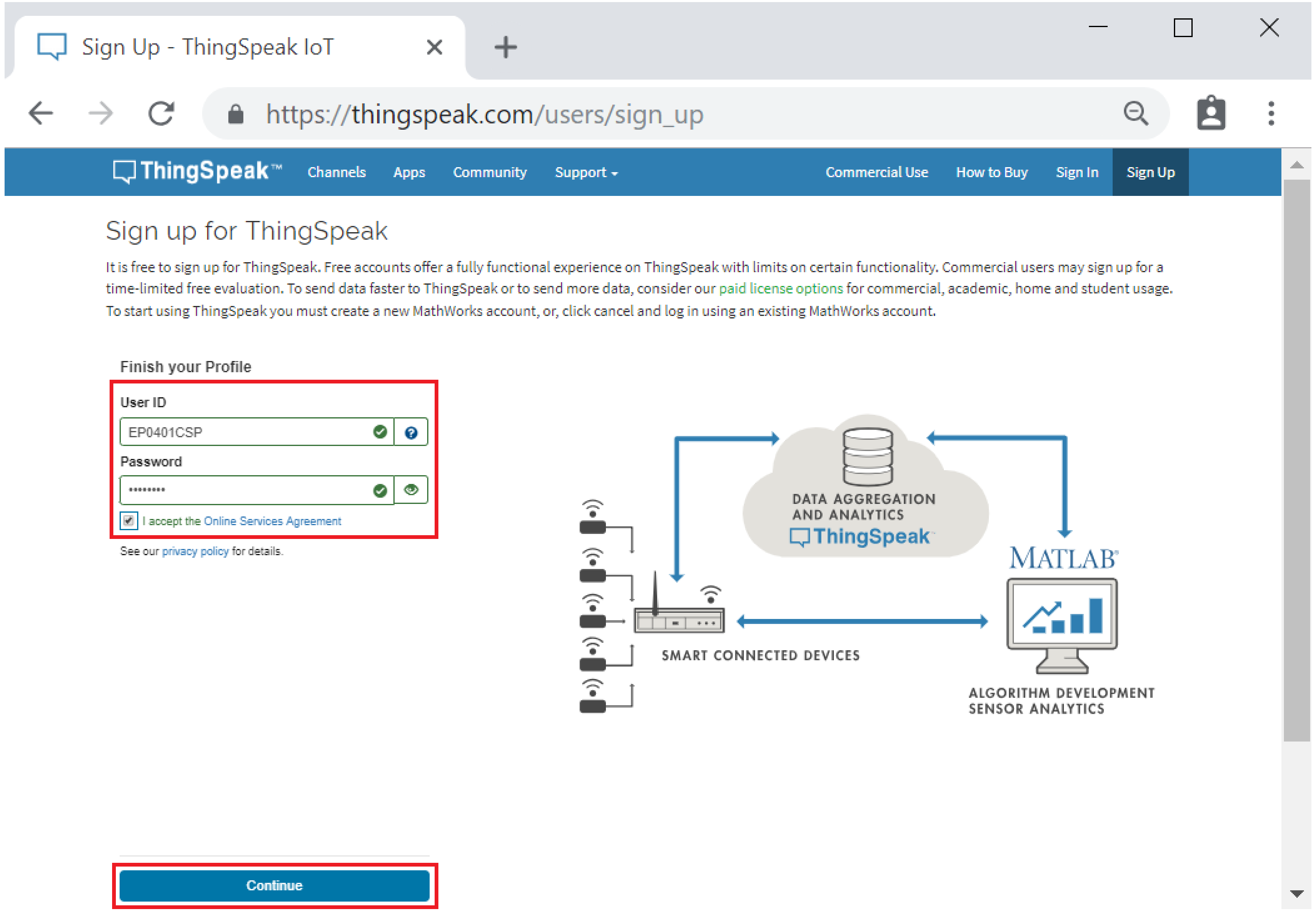
Didn't receive the email?

- Check your spam folder.
- Send me the email again.
- If you still have not received the email, Contact Customer Support

DATA AGGREGATION AND ANALYTICS
ThingSpeak
MATLAB
SMART CONNECTED DEVICES
ALGORITHM DEVELOPMENT SENSOR ANALYTICS
Continue

Back to the Thingspeak tab, click Continue.

Setting up a Cloud platform (e.g. Thingspeak) (cont.)



Sign up for ThingSpeak

It is free to sign up for ThingSpeak. Free accounts offer a fully functional experience on ThingSpeak with limits on certain functionality. Commercial users may sign up for a time-limited free evaluation. To send data faster to ThingSpeak or to send more data, consider our paid license options for commercial, academic, home and student usage. To start using ThingSpeak you must create a new MathWorks account, or, click cancel and log in using an existing MathWorks account.

Finish your Profile

User ID: EP0401CSP

Password: ••••••••

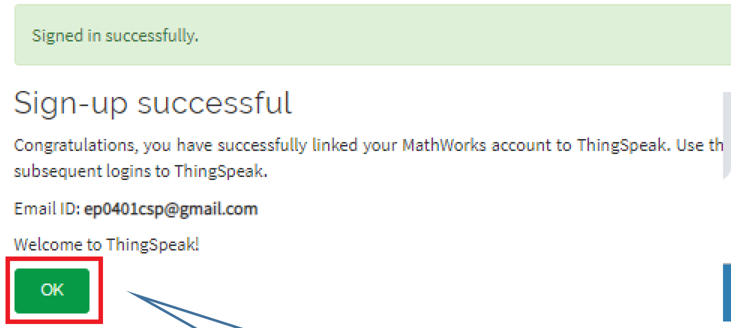
☑ I accept the Online Services Agreement

See our privacy policy for details.

DATA AGGREGATION AND ANALYTICS
ThingSpeak
MATLAB
SMART CONNECTED DEVICES
ALGORITHM DEVELOPMENT SENSOR ANALYTICS
Continue

Enter your User ID and Password, tick the box and click Continue.

Setting up a Cloud platform (e.g. Thingspeak) (cont.)



Signed in successfully.

X

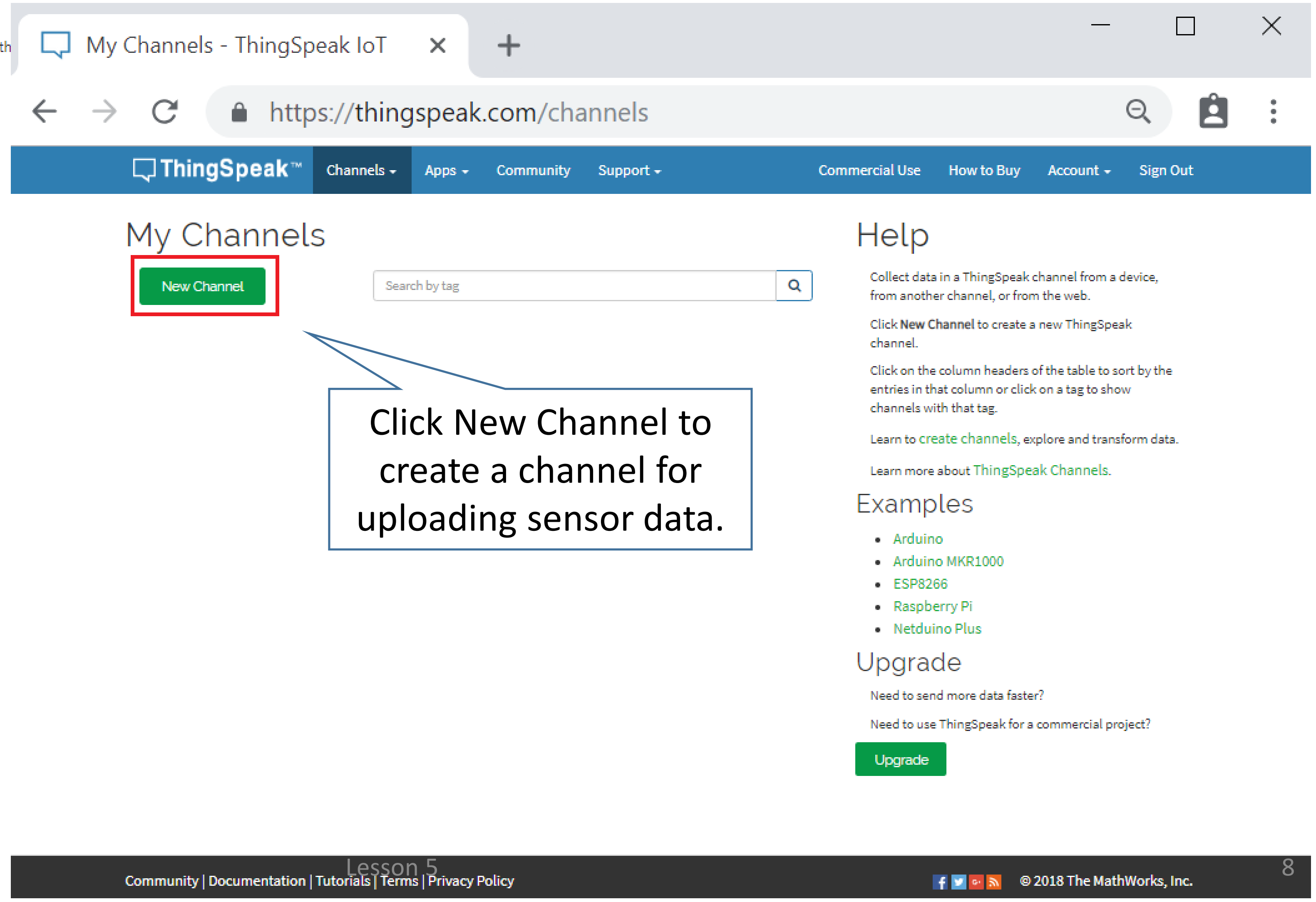
Sign-up successful

Congratulations, you have successfully linked your MathWorks account to ThingSpeak. Use th
subsequent logins to ThingSpeak.

Email ID: ep0401csp@gmail.com

Welcome to ThingSpeak!

Click OK to start using Thingspeak.



My Channels

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

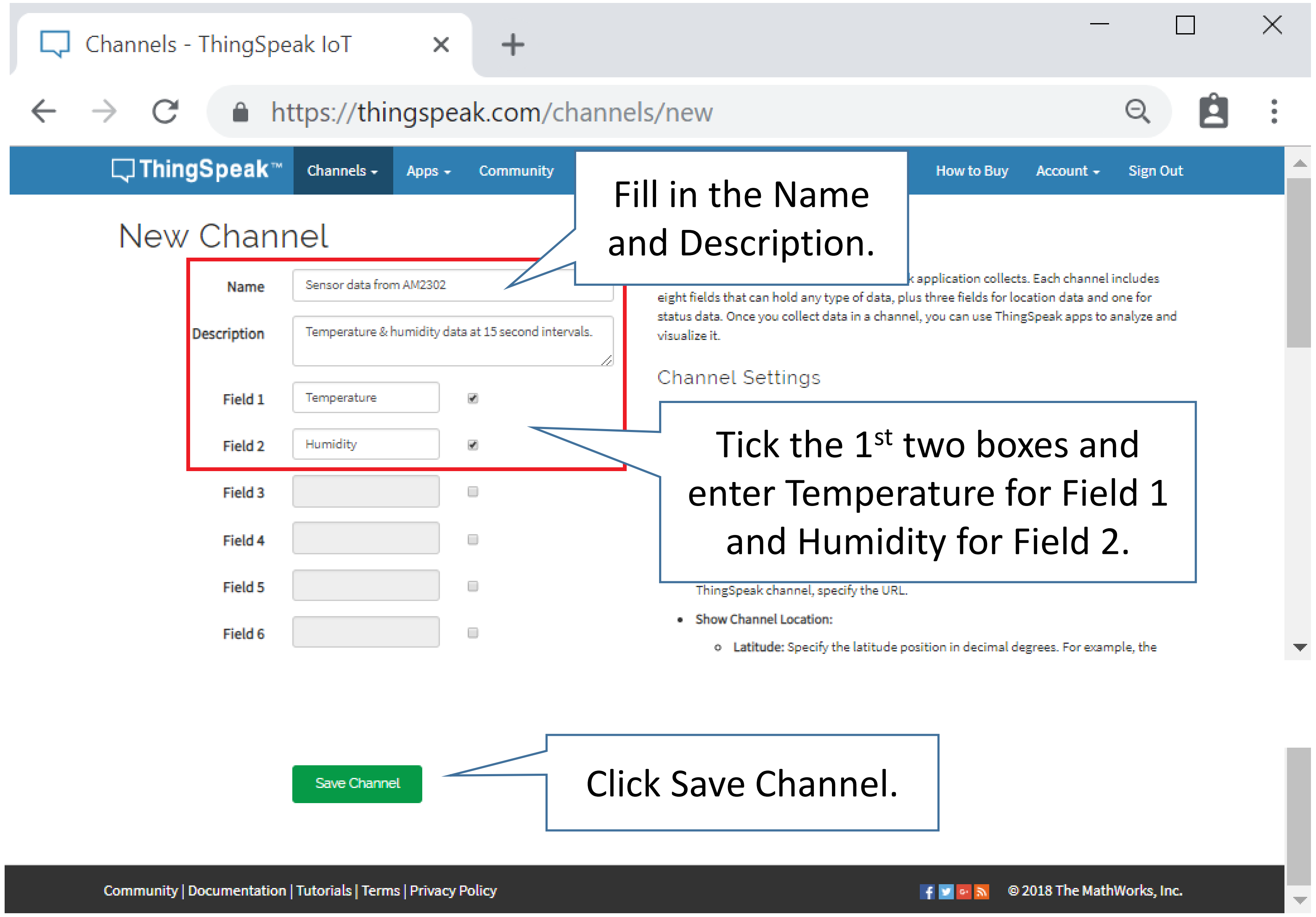
Need to send more data faster?

Need to use ThingSpeak for a commercial project?

Upgrade
Community | Documentation | Tutorials | Terms | Privacy Policy
© 2018 The MathWorks, Inc.

Click New Channel to create a channel for uploading sensor data.

Setting up a Cloud platform (e.g. Thingspeak) (cont.)



Channels - ThingSpeak IoT
https://thingspeak.com/channels/new
ThingSpeak™
Channels
Apps
Community
How to Buy
Account
Sign Out
New Channel
Save Channel
Channel Settings
Community | Documentation | Tutorials | Terms | Privacy Policy
© 2018 The MathWorks, Inc.

Name	Sensor data from AM2302	
Description	Temperature & humidity data at 15 second intervals.	
Field 1	Temperature	✓
Field 2	Humidity	✓
Field 3		
Field 4		
Field 5		
Field 6		

application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

ThingSpeak channel, specify the URL.

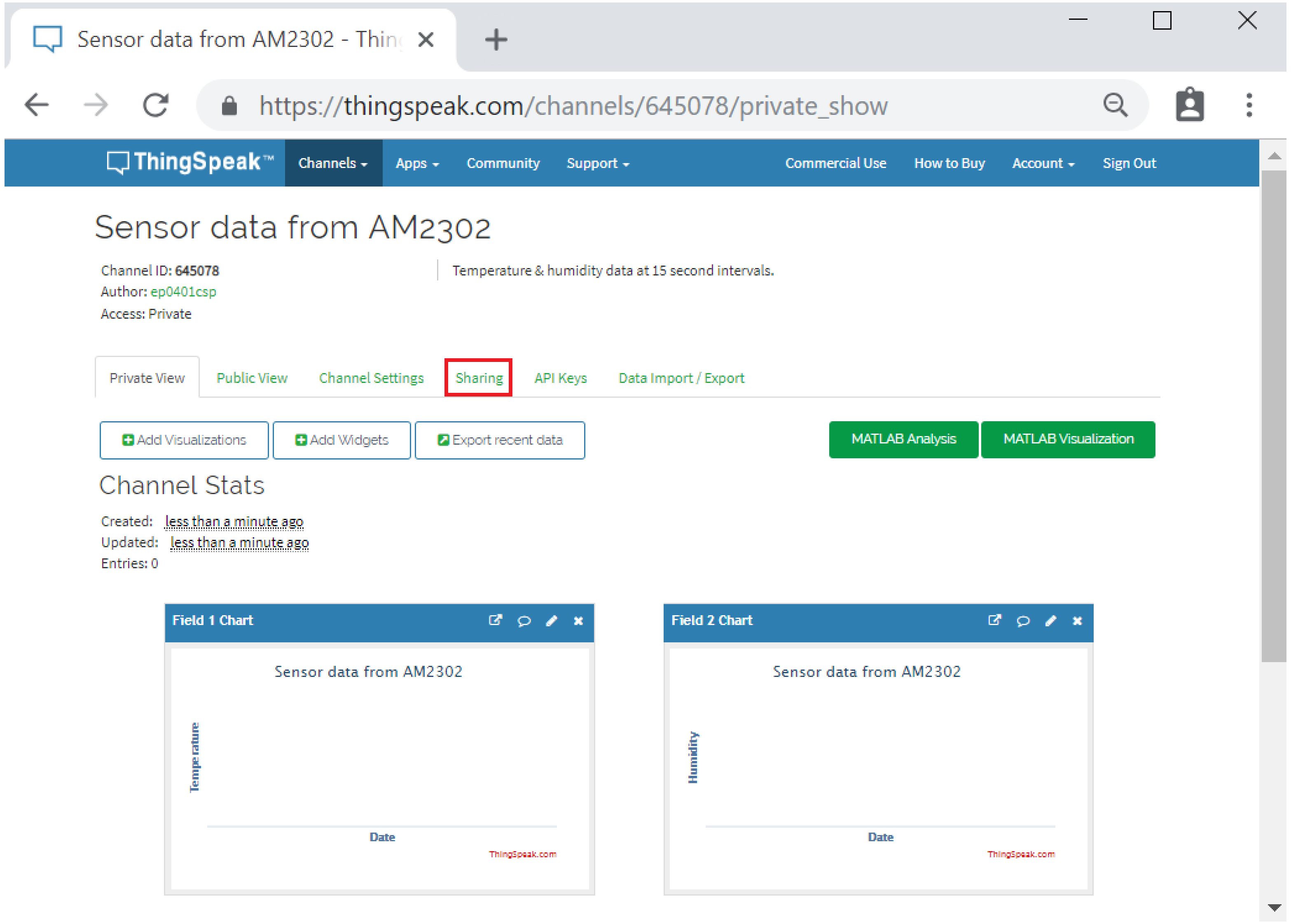
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the

Fill in the Name and Description.

Tick the 1st two boxes and enter Temperature for Field 1 and Humidity for Field 2.

Click Save Channel.

Setting up a Cloud platform (e.g. Thingspeak) (cont.)



Sensor data from AM2302 - Thin

https://thingspeak.com/channels/645078/private_show

Sensor data from AM2302

Channel ID: 645078
Author: ep0401csp
Access: Private

Temperature & humidity data at 15 second intervals.

Channel Stats

Created: less than a minute ago
Updated: less than a minute ago
Entries: 0

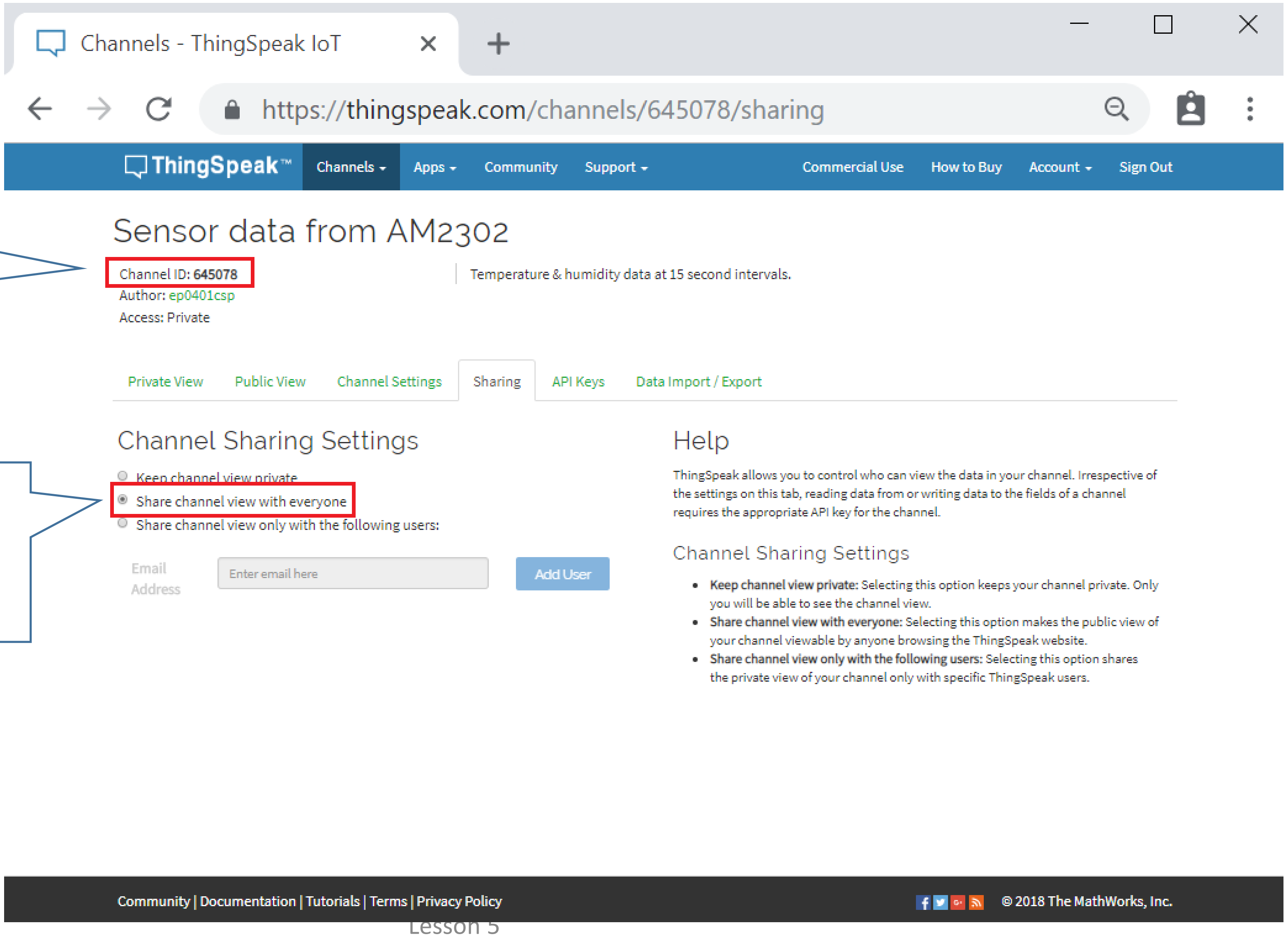
You will see 2 charts (Temperature & Humidity) in the channel just created.

The Access is now Private. To change this, click Sharing.

Setting up a Cloud platform (e.g. Thingspeak) (cont.)

Note the Channel ID, in this case 645078.

Tick the box "Share channel view with everyone".



Sensor data from AM2302

Channel ID: **645078**
Author: ep0401csp
Access: Private

Temperature & humidity data at 15 second intervals.

Channel Sharing Settings

( ) Keep channel view private
(●) Share channel view with everyone
( ) Share channel view only with the following users:

Email Address: Enter email here

Help

ThingSpeak allows you to control who can view the data in your channel. Irrespective of the settings on this tab, reading data from or writing data to the fields of a channel requires the appropriate API key for the channel.

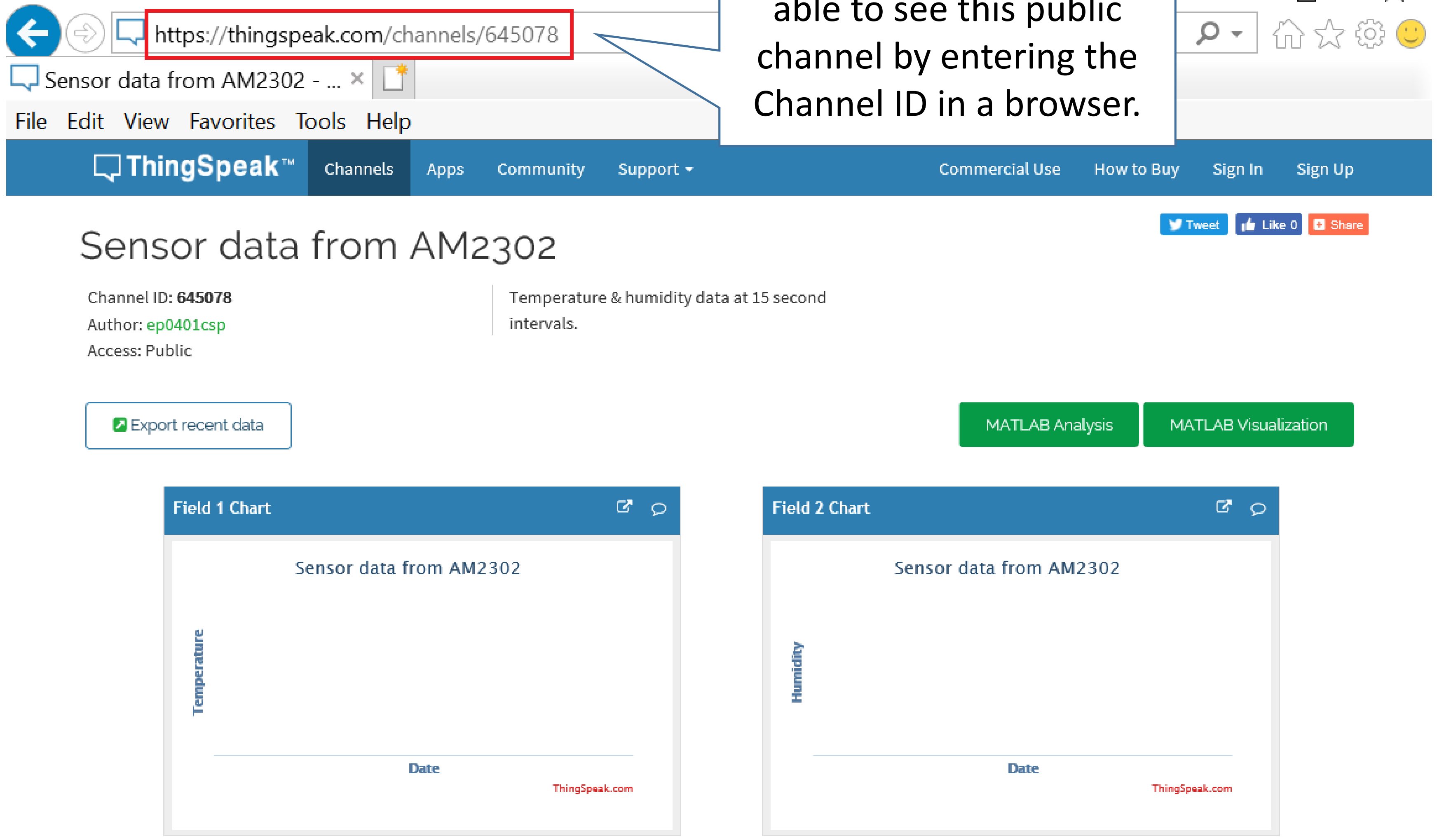
Channel Sharing Settings

- **Keep channel view private:** Selecting this option keeps your channel private. Only you will be able to see the channel view.
- **Share channel view with everyone:** Selecting this option makes the public view of your channel viewable by anyone browsing the ThingSpeak website.
- **Share channel view only with the following users:** Selecting this option shares the private view of your channel only with specific ThingSpeak users.

Community | Documentation | Tutorials | Terms | Privacy Policy
© 2018 The MathWorks, Inc.

Setting up a Cloud platform (e.g. Thingspeak) (cont.)

After this, anyone will be able to see this public channel by entering the Channel ID in a browser.



Sensor data from AM2302

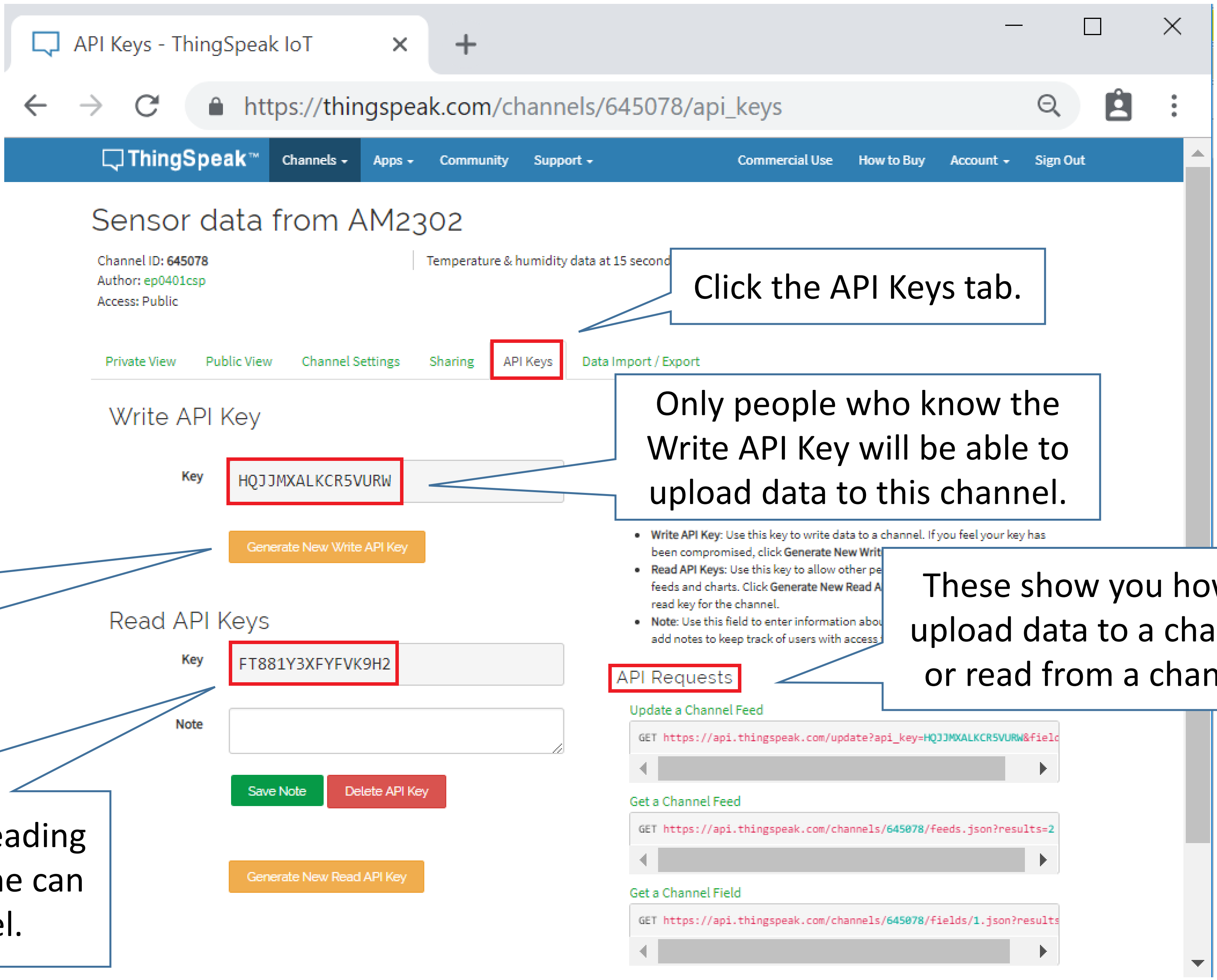
Channel ID: **645078**
Author: ep0401csp
Access: Public

Temperature & humidity data at 15 second intervals.

Sensor data from AM2302

Sensor data from AM2302

Setting up a Cloud platform (e.g. Thingspeak) (cont.)



Sensor data from AM2302

Channel ID: 645078
Author: ep0401csp
Access: Public

Temperature & humidity data at 15 second

Write API Key

Key: HQJJMXALKCR5VURW

Read API Keys

Key: FT881Y3XFYFVK9H2

Note:

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Writ**
- **Read API Keys**: Use this key to allow other pe feeds and charts. Click **Generate New Read A** read key for the channel.
- **Note**: Use this field to enter information abou add notes to keep track of users with access

API Requests

Update a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=HQJJMXALKCR5VURW&fielc
```

Get a Channel Feed

```
GET https://api.thingspeak.com/channels/645078/feeds.json?results=2
```

Get a Channel Field

```
GET https://api.thingspeak.com/channels/645078/fields/1.json?results
```

Click the API Keys tab.

Only people who know the Write API Key will be able to upload data to this channel.

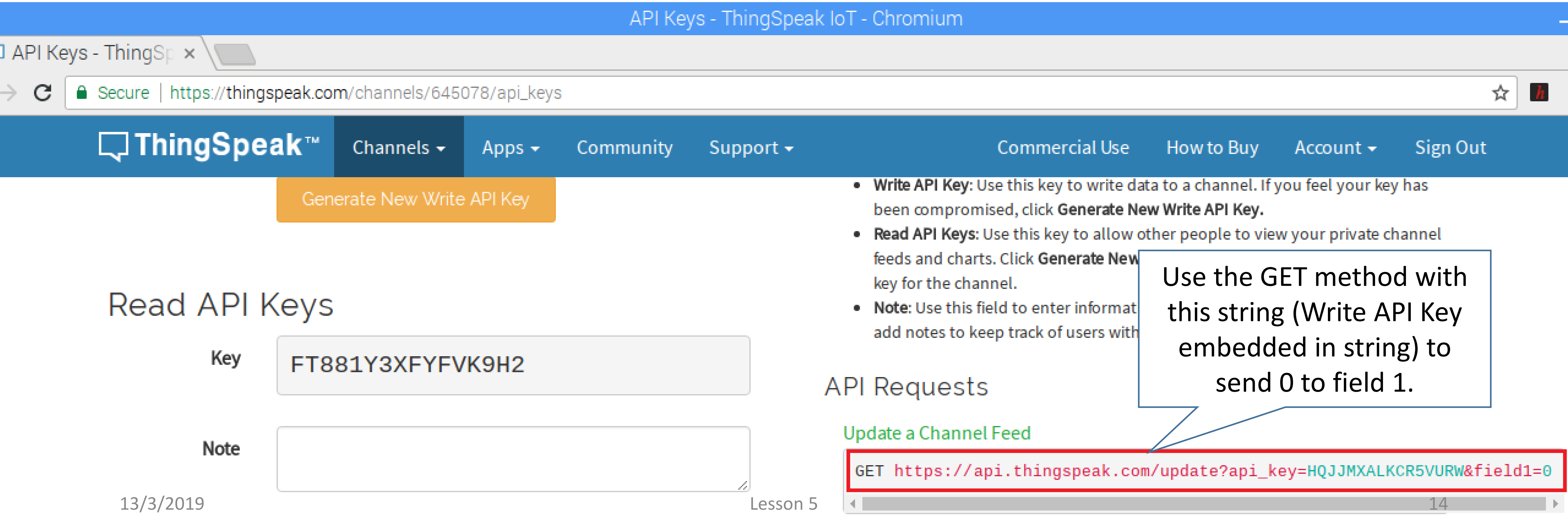
If you think that this "password" has been compromised, you can always generate a new one.

These show you how to upload data to a channel, or read from a channel.

Read API Key is required for reading from a private channel. Anyone can read from a public channel.

Sending data to cloud for storage & visualization

- Let's write a Python program to **send** sensor data (such as those from a temperature & humidity sensor AM2302) to two different fields of a Thingspeak channel.
- Thingspeak gives the **sample code** below for **updating a channel feed**:



API Keys - ThingSpeak IoT - Chromium

API Keys - ThingSp ×

Secure | https://thingspeak.com/channels/645078/api_keys

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New** key for the channel.
- **Note:** Use this field to enter informat add notes to keep track of users with

Read API Keys

Key: FT881Y3XFYFVK9H2

Note:

API Requests

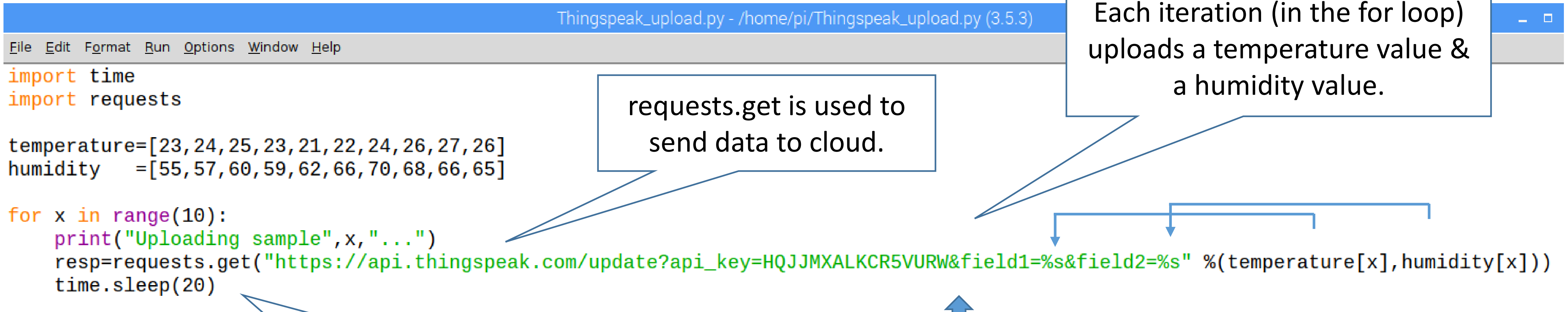
Update a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=HQJJMXALKCR5VURW&field1=0
```

Use the GET method with this string (Write API Key embedded in string) to send 0 to field 1.

Sending data to cloud for storage & visualization (cont.)

- This translates into the Python code below, which uses the **requests** module.
- To keep things simple, two lists (of temperature + humidity values) are used, instead of real readings from the AM2302 sensor.

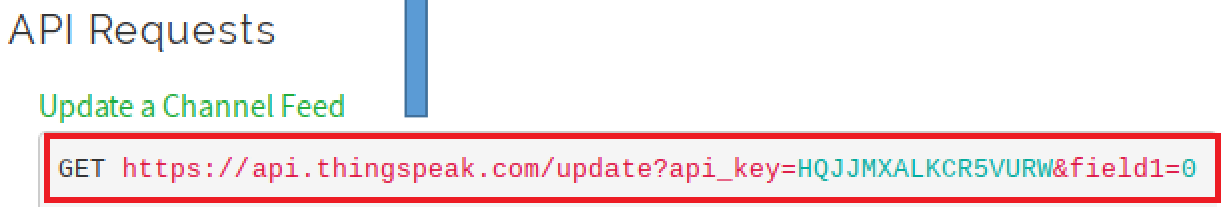

Thingspeak_upload.py - /home/pi/Thingspeak_upload.py (3.5.3)
File Edit Format Run Options Window Help

```
import time
import requests

temperature=[23,24,25,23,21,22,24,26,27,26]
humidity   =[55,57,60,59,62,66,70,68,66,65]

for x in range(10):
    print("Uploading sample",x,"...")
    resp=requests.get("https://api.thingspeak.com/update?api_key=HQJJMXALKCR5VURW&field1=%s&field2=%s" %(temperature[x],humidity[x]))
    time.sleep(20)
```

requests.get is used to send data to cloud.

Each iteration (in the for loop) uploads a temperature value & a humidity value.

An interval of at least 15 seconds is required, for uploading to a free channel.



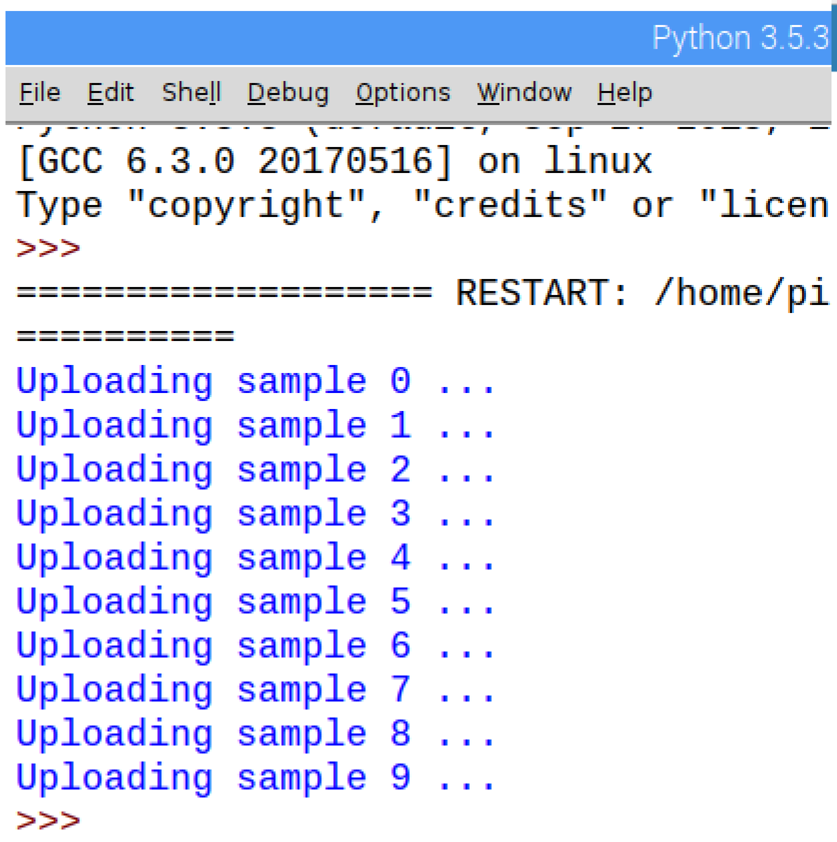
API Requests

Update a Channel Feed

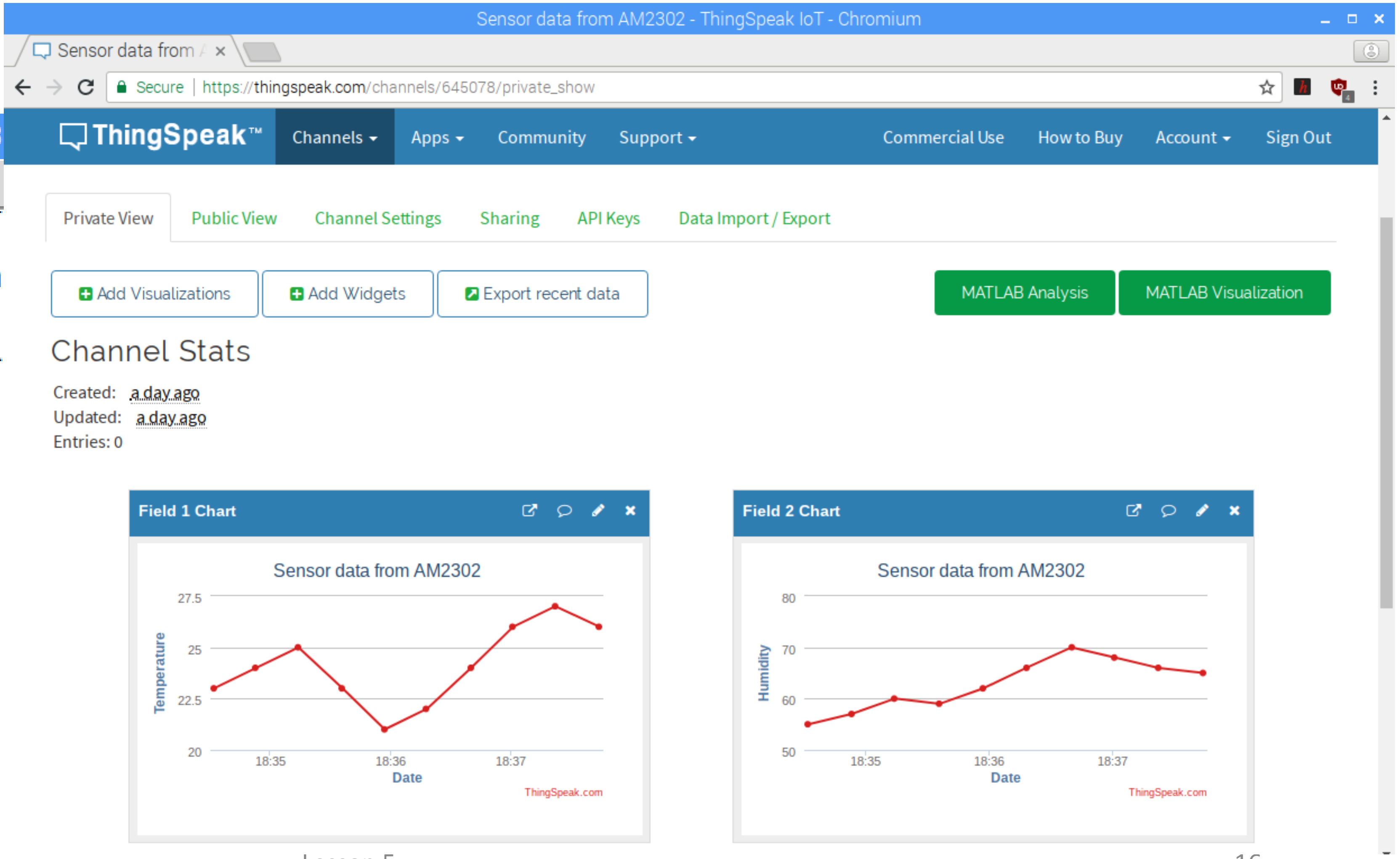
```
GET https://api.thingspeak.com/update?api_key=HQJJMXALKCR5VURW&field1=0
```

Sending data to cloud for storage & visualization (cont.)

- After a while, all 10 sets of values have been uploaded:



```
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "licen
>>>
=================== RESTART: /home/pi
==========
Uploading sample 0 ...
Uploading sample 1 ...
Uploading sample 2 ...
Uploading sample 3 ...
Uploading sample 4 ...
Uploading sample 5 ...
Uploading sample 6 ...
Uploading sample 7 ...
Uploading sample 8 ...
Uploading sample 9 ...
>>>
```



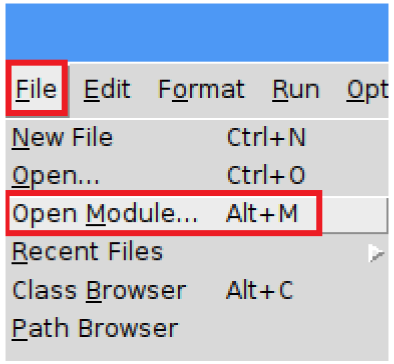
Channel Stats

Created: a day ago
Updated: a day ago
Entries: 0

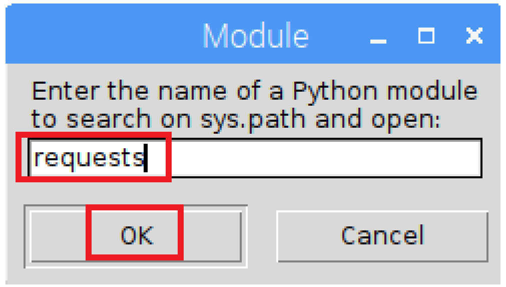
Sending data to cloud for storage & visualization (cont.)

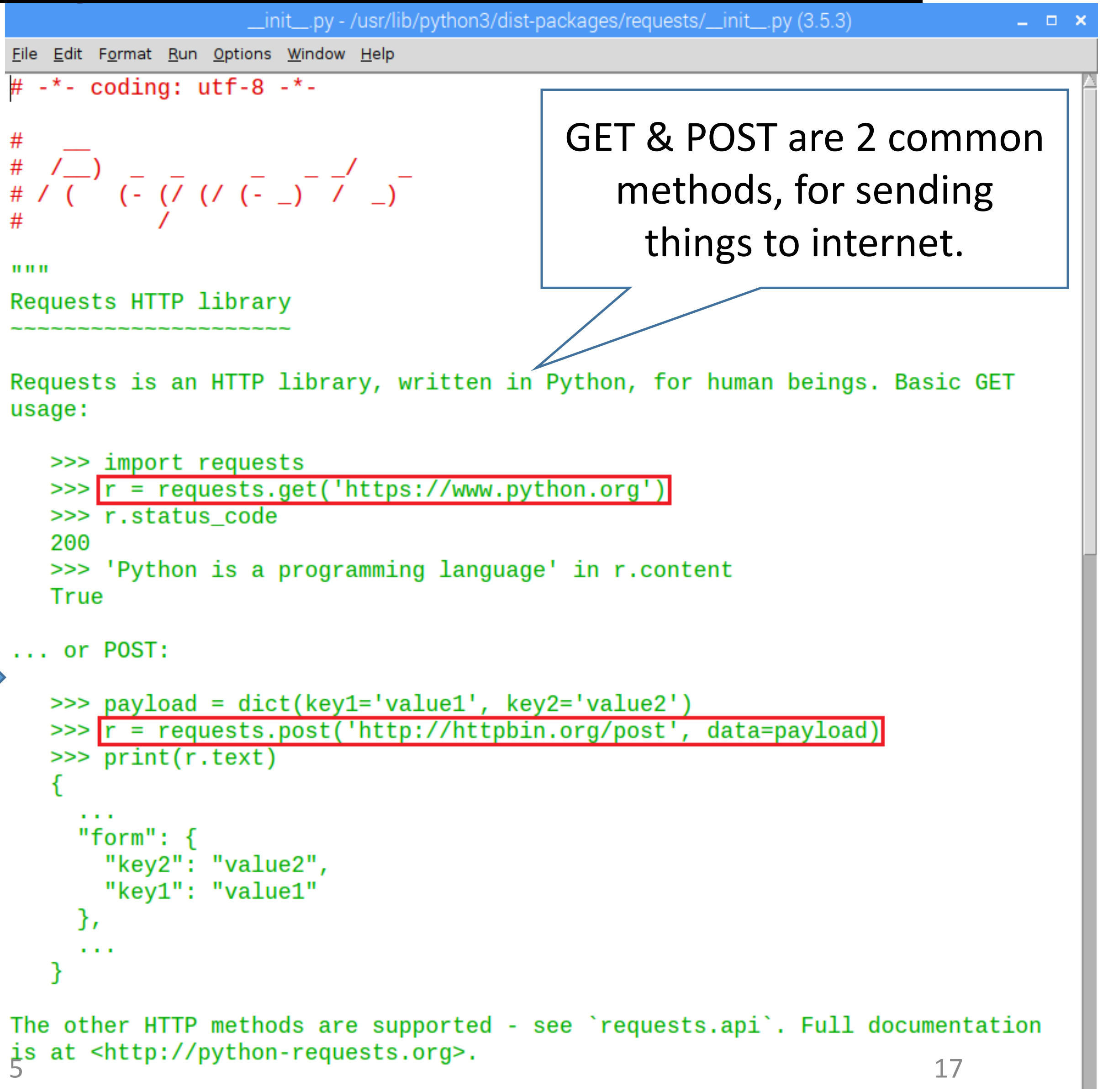
- You can actually open the **request** modules, to see what it contains:

In the Python IDLE, click File -> Open Module.



Enter requests & click OK.


Module
Enter the name of a Python module to search on sys.path and open:
requests
OK
Cancel

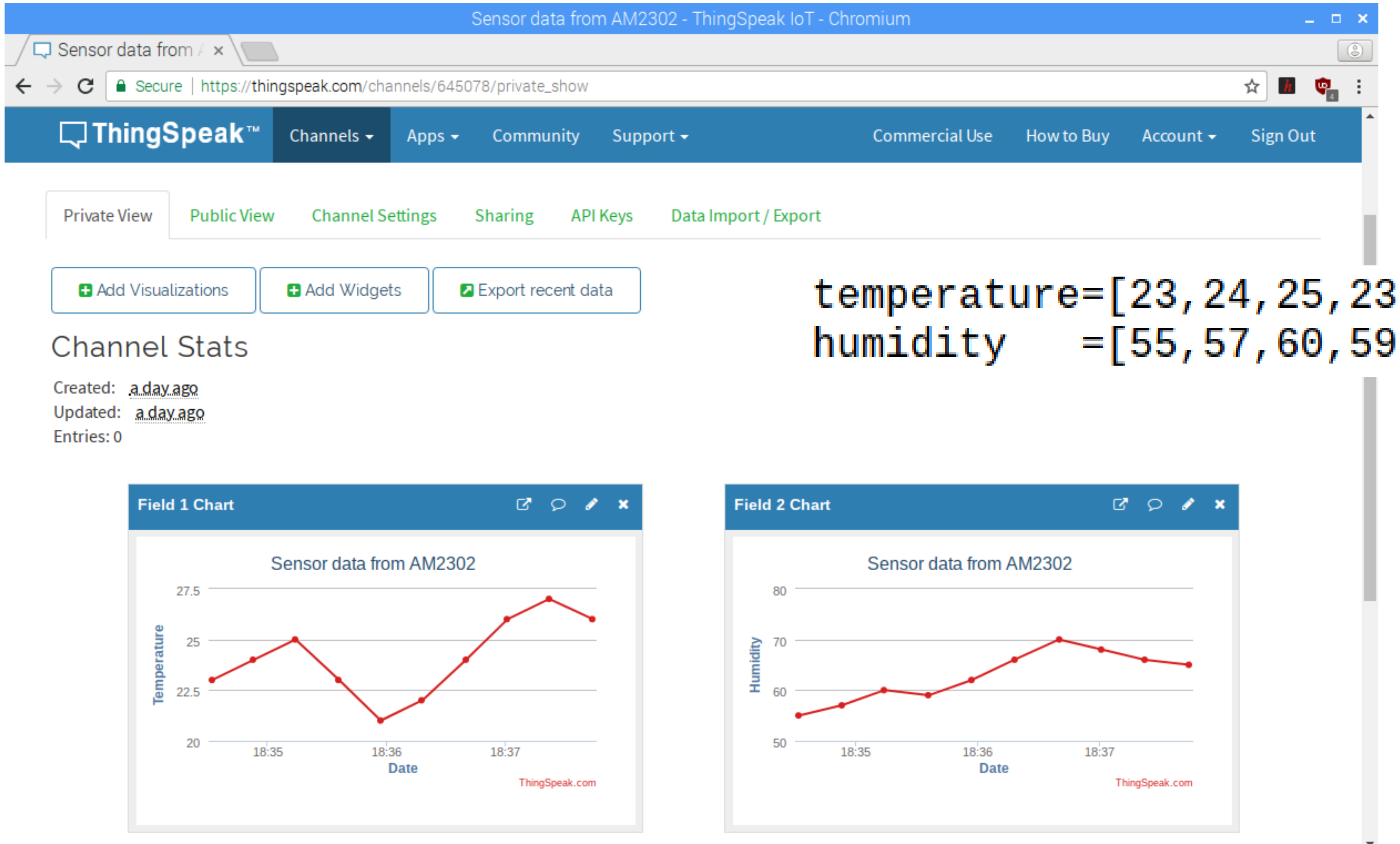

__init__.py - /usr/lib/python3/dist-packages/requests/__init__.py (3.5.3)
File
Edit
Format
Run
Options
Window
Help

```
# -*- coding: utf-8 -*-

#   __
#  /__)  _  _     _   _ _/   _
# / (   (- (/ (/ (- _)  /  _)
#          /

"""
Requests HTTP library
~~~~~~~~~~~~~~~~~~~~~

Requests is an HTTP library, written in Python, for human beings. Basic GET
usage:

   >>> import requests
   >>> r = requests.get('https://www.python.org')
   >>> r.status_code
   200
   >>> 'Python is a programming language' in r.content
   True

... or POST:

   >>> payload = dict(key1='value1', key2='value2')
   >>> r = requests.post('http://httpbin.org/post', data=payload)
   >>> print(r.text)
   {
     ...
     "form": {
       "key2": "value2",
       "key1": "value1"
     },
     ...
   }

The other HTTP methods are supported - see `requests.api`. Full documentation
is at <http://python-requests.org>.
```

GET & POST are 2 common methods, for sending things to internet.

Reading data from cloud

- Let's write a Python program to **read** sensor data (temperature & humidity readings) from two different fields of a Thingspeak channel, based on Thingspeak's sample code.



Channel Stats

Created: a day ago
Updated: a day ago
Entries: 0

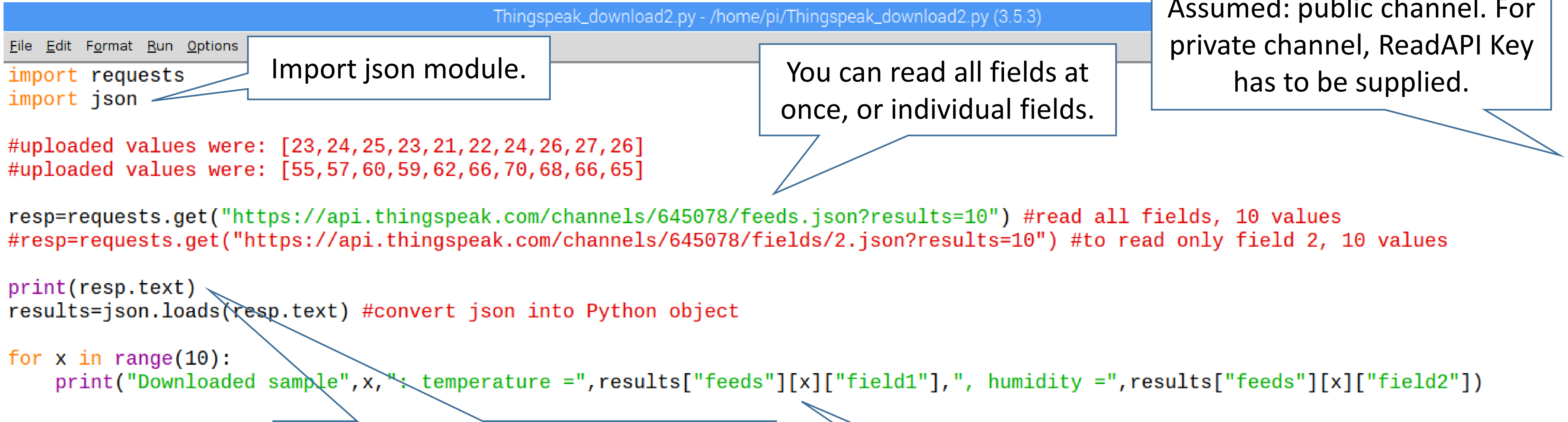
We should be able to read back the values uploaded previously.

```
temperature=[23,24,25,23,21,22,24,26,27,26]
humidity   =[55,57,60,59,62,66,70,68,66,65]
```

Reading data from cloud (cont.)

Ref: https://en.wikipedia.org/wiki/JSON

- The data downloaded is in the form of **json** (JavaScript Object Notation) object – which is very similar to a Python **dictionary** consisting of key-value pairs.


Thingspeak_download2.py - /home/pi/Thingspeak_download2.py (3.5.3)
File Edit Format Run Options

```
import requests
import json

#uploaded values were: [23,24,25,23,21,22,24,26,27,26]
#uploaded values were: [55,57,60,59,62,66,70,68,66,65]

resp=requests.get("https://api.thingspeak.com/channels/645078/feeds.json?results=10") #read all fields, 10 values
#resp=requests.get("https://api.thingspeak.com/channels/645078/fields/2.json?results=10") #to read only field 2, 10 values

print(resp.text)
results=json.loads(resp.text) #convert json into Python object

for x in range(10):
    print("Downloaded sample",x,": temperature =",results["feeds"][x]["field1"],", humidity =",results["feeds"][x]["field2"])
```

Import json module.

You can read all fields at once, or individual fields.

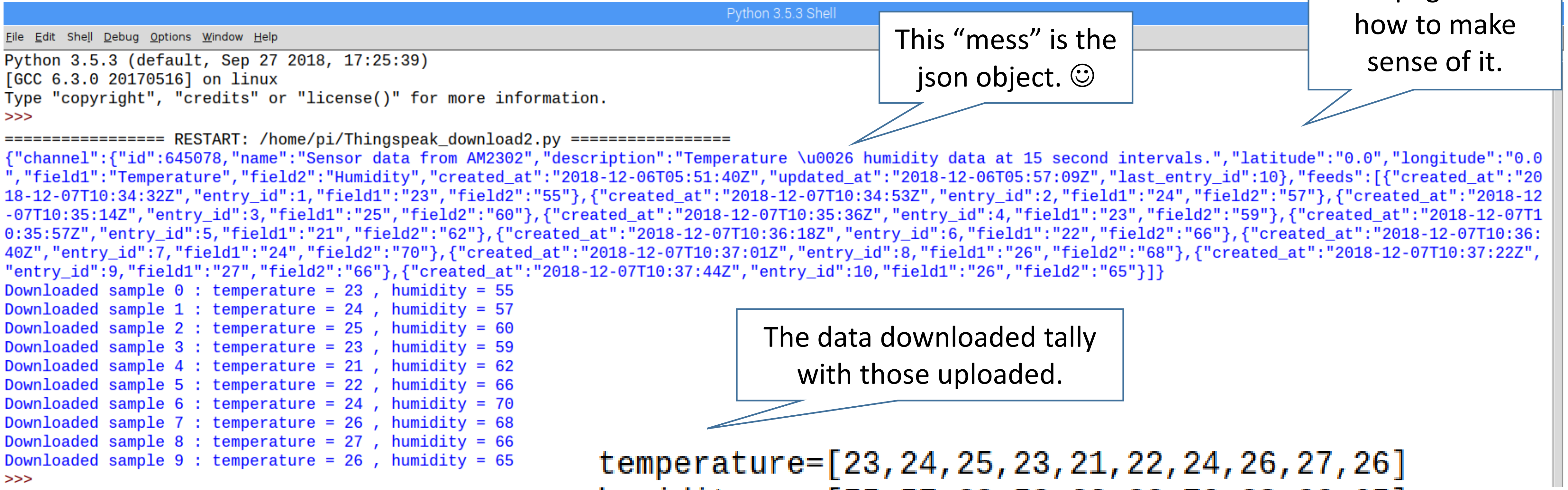
Assumed: public channel. For private channel, ReadAPI Key has to be supplied.

Comment out this line, if you don't want to see what is in the json object.

You will understand this line once we use an online json viewer to see what is inside the "results".

Reading data from cloud (cont.)

- The Python program prints out the ‘raw” json object, and extracts the 10 sets of temperature & humidity values:



```
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: /home/pi/Thingspeak_download2.py =================
```

{"channel":{"id":645078,"name":"Sensor data from AM2302","description":"Temperature \u0026 humidity data at 15 second intervals.","latitude":"0.0","longitude":"0.0","field1":"Temperature","field2":"Humidity","created_at":"2018-12-06T05:51:40Z","updated_at":"2018-12-06T05:57:09Z","last_entry_id":10},"feeds":[{"created_at":"2018-12-07T10:34:32Z","entry_id":1,"field1":"23","field2":"55"},{"created_at":"2018-12-07T10:34:53Z","entry_id":2,"field1":"24","field2":"57"},{"created_at":"2018-12-07T10:35:14Z","entry_id":3,"field1":"25","field2":"60"},{"created_at":"2018-12-07T10:35:36Z","entry_id":4,"field1":"23","field2":"59"},{"created_at":"2018-12-07T10:35:57Z","entry_id":5,"field1":"21","field2":"62"},{"created_at":"2018-12-07T10:36:18Z","entry_id":6,"field1":"22","field2":"66"},{"created_at":"2018-12-07T10:36:40Z","entry_id":7,"field1":"24","field2":"70"},{"created_at":"2018-12-07T10:37:01Z","entry_id":8,"field1":"26","field2":"68"},{"created_at":"2018-12-07T10:37:22Z","entry_id":9,"field1":"27","field2":"66"},{"created_at":"2018-12-07T10:37:44Z","entry_id":10,"field1":"26","field2":"65"}]}

```
Downloaded sample 0 : temperature = 23 , humidity = 55
Downloaded sample 1 : temperature = 24 , humidity = 57
Downloaded sample 2 : temperature = 25 , humidity = 60
Downloaded sample 3 : temperature = 23 , humidity = 59
Downloaded sample 4 : temperature = 21 , humidity = 62
Downloaded sample 5 : temperature = 22 , humidity = 66
Downloaded sample 6 : temperature = 24 , humidity = 70
Downloaded sample 7 : temperature = 26 , humidity = 68
Downloaded sample 8 : temperature = 27 , humidity = 66
Downloaded sample 9 : temperature = 26 , humidity = 65
>>>
```

This “mess” is the json object. ☺

Next page shows how to make sense of it.

The data downloaded tally with those uploaded.

```
temperature=[23,24,25,23,21,22,24,26,27,26]
humidity   =[55,57,60,59,62,66,70,68,66,65]
```

Reading data from cloud (cont.)

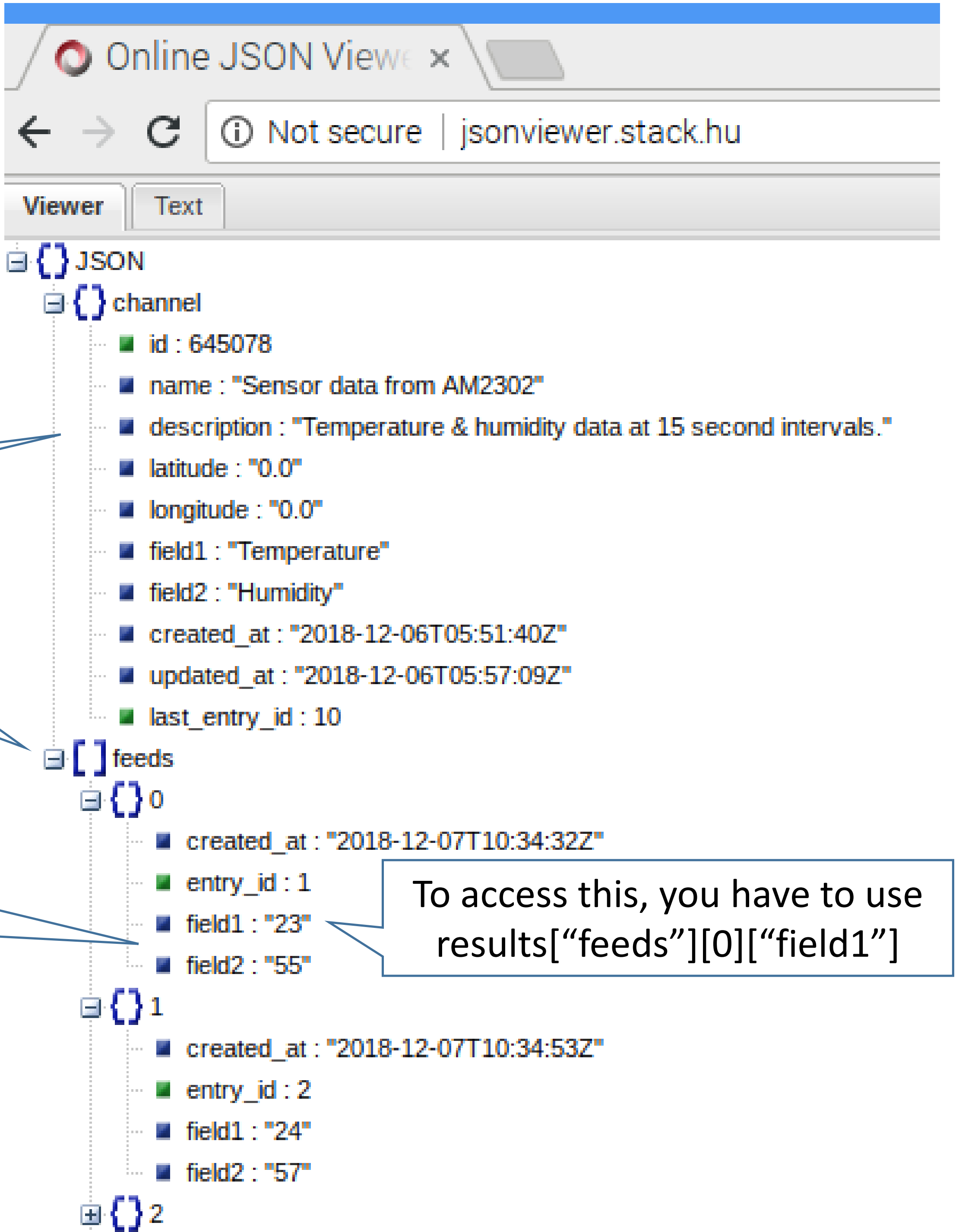
- Using an **online json viewer** (just Google it!), you will be able to make sense of the “mess” (i.e. the json object) on the previous slide.

It consists of channel info…

…and feeds, which consists of a number of data point.

Each data point consists of the timestamp, the entry id and the fields 1 & 2 data (i.e. temperature & humidity).

To access this, you have to use results[“feeds”][0][“field1”]


Online JSON View ×

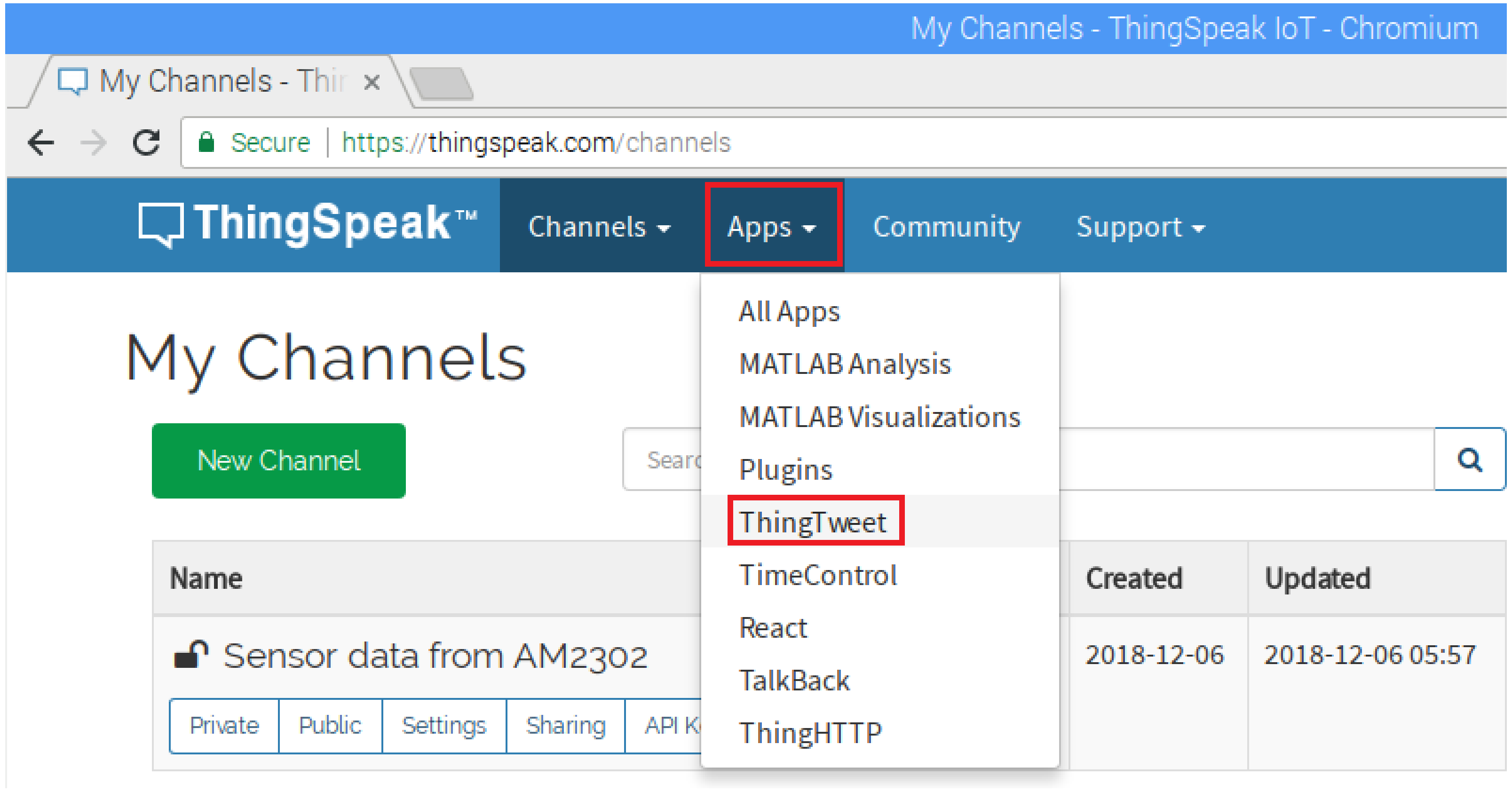
Not secure | jsonviewer.stack.hu

Viewer | Text

```
{} JSON
  {} channel
    id : 645078
    name : "Sensor data from AM2302"
    description : "Temperature & humidity data at 15 second intervals."
    latitude : "0.0"
    longitude : "0.0"
    field1 : "Temperature"
    field2 : "Humidity"
    created_at : "2018-12-06T05:51:40Z"
    updated_at : "2018-12-06T05:57:09Z"
    last_entry_id : 10
  [] feeds
    {} 0
      created_at : "2018-12-07T10:34:32Z"
      entry_id : 1
      field1 : "23"
      field2 : "55"
    {} 1
      created_at : "2018-12-07T10:34:53Z"
      entry_id : 2
      field1 : "24"
      field2 : "57"
    {} 2
```

Sending an event-triggered notification (e.g. a Tweet)

- Let's write a Python program to send a Tweet.
- This can be used to notify someone (a 'follower'), when the temperature or humidity reading is too high, for example.



My Channels - ThingSpeak IoT - Chromium

My Channels - Thir ×

Secure | https://thingspeak.com/channels

ThingSpeak™ Channels ▾ Apps ▾ Community Support ▾

All Apps
MATLAB Analysis
MATLAB Visualizations
Plugins
ThingTweet
TimeControl
React
TalkBack
ThingHTTP

My Channels

New Channel

Searc

Name	Created	Updated
Sensor data from AM2302	2018-12-06	2018-12-06 05:57

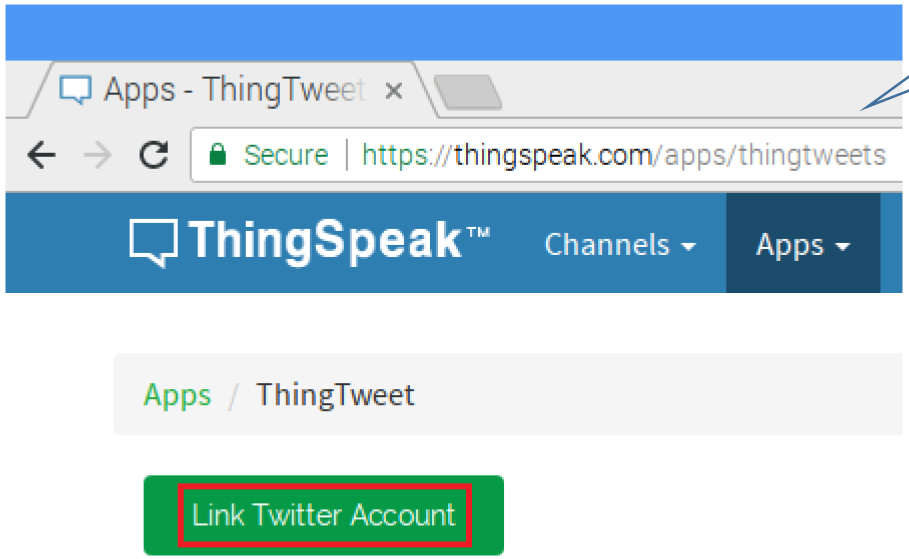
Private | Public | Settings | Sharing | API K

Click Apps -> ThingTweet

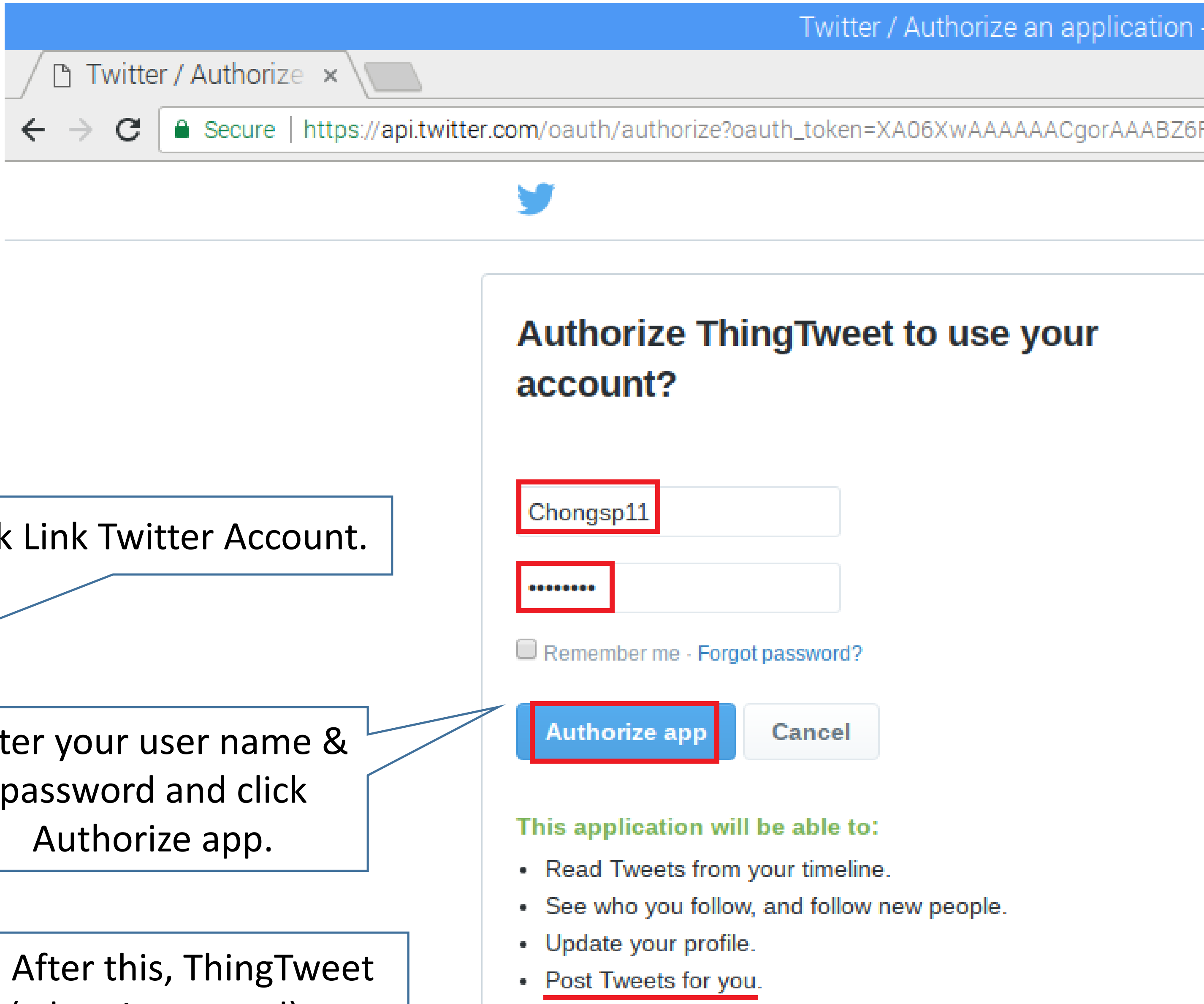
This Thingspeak App can post tweets on your behalf.

Sending an event-triggered notification (e.g. a Tweet) (cont.)

- If you don't have a Twitter account, create one at Twitter.com.
- Alternatively, you can also use the Twitter mobile app to do this.



Click Link Twitter Account.


Twitter / Authorize an application
Twitter / Authorize
Secure | https://api.twitter.com/oauth/authorize?oauth_token=XA06XwAAAAAACgorAAABZ6F

Authorize ThingTweet to use your account?

Chongsp11

••••••••

Remember me · Forgot password?

Authorize app Cancel

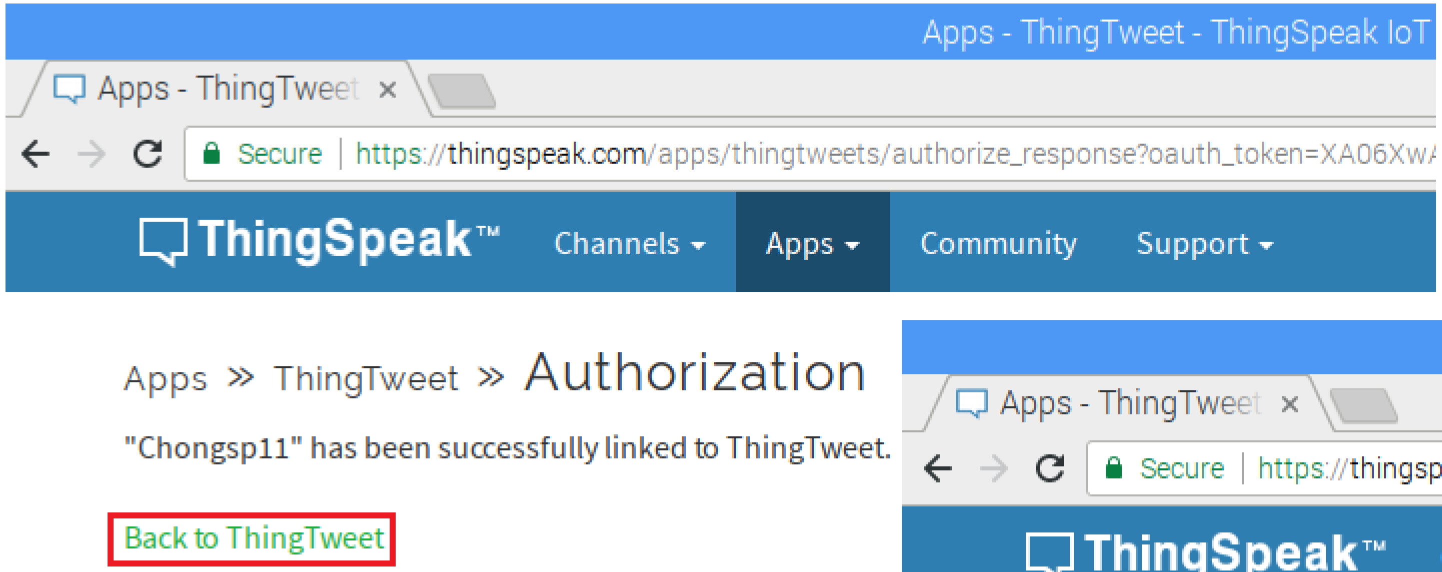
This application will be able to:

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.

Enter your user name & password and click Authorize app.

After this, ThingTweet (when instructed), can post Tweets for you.

Sending an event-triggered notification (e.g. a Tweet)(cont.)



Apps - ThingTweet - ThingSpeak IoT

https://thingspeak.com/apps/thingtweets/authorize_response?oauth_token=XA06XwA

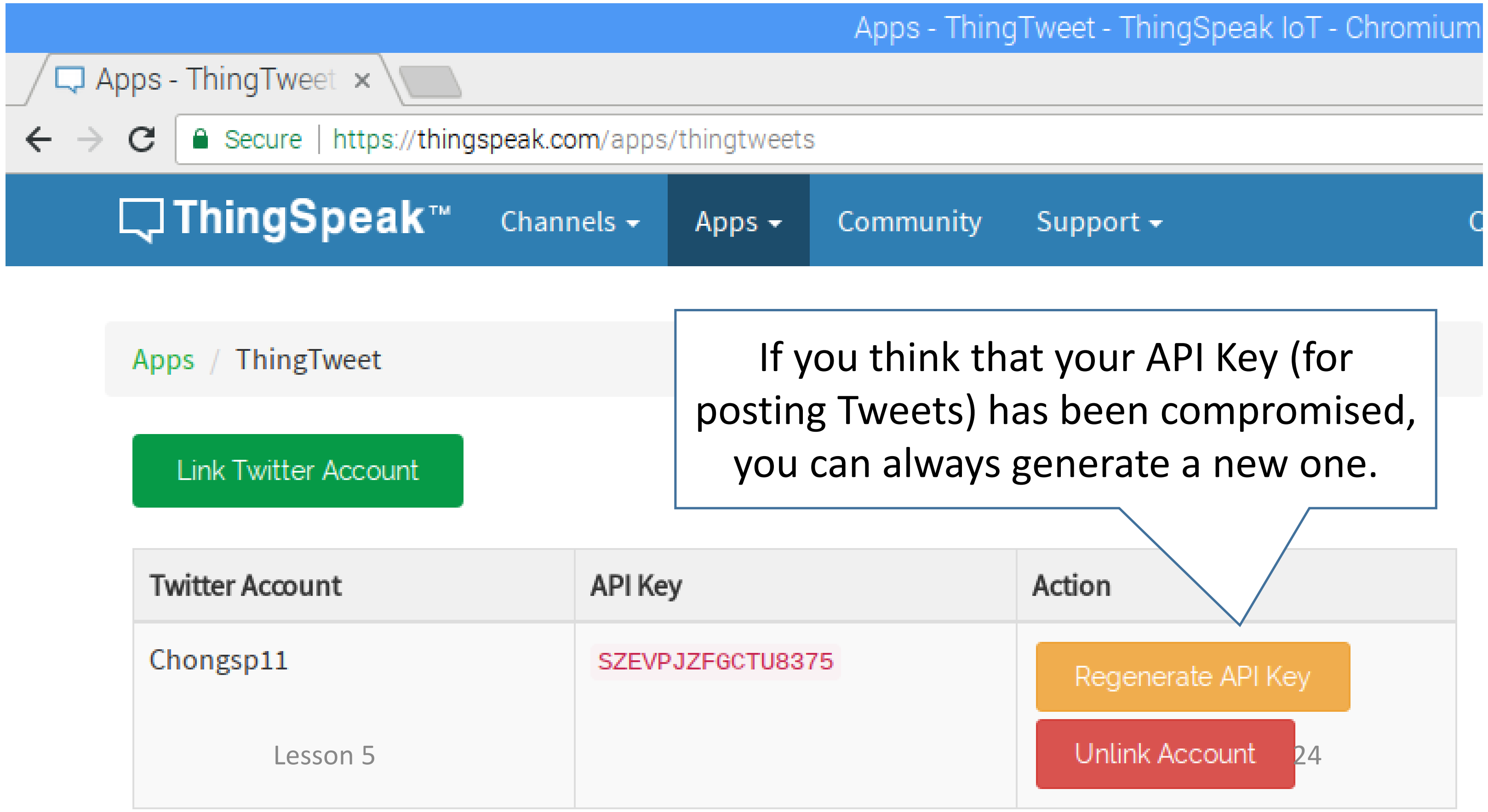
Apps » ThingTweet » Authorization

"Chongsp11" has been successfully linked to ThingTweet.

Back to ThingTweet

Click Back to ThingTweet.

- The Twitter account has been linked to ThingTweet.
- But it can be unlinked.



Apps - ThingTweet - ThingSpeak IoT - Chromium

https://thingspeak.com/apps/thingtweets

Apps / ThingTweet

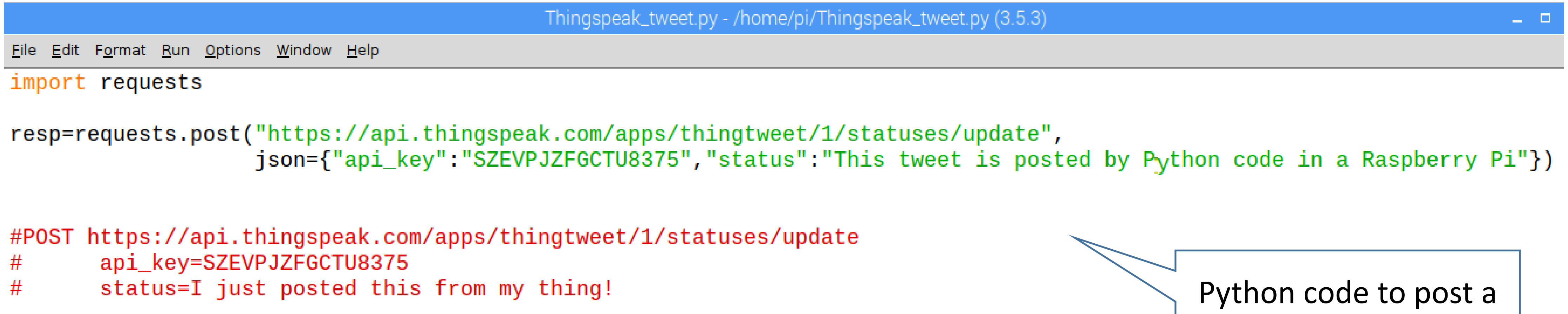
Link Twitter Account

Twitter Account	API Key	Action
Chongsp11	SZEVPJZFGCTU8375	Regenerate API Key Unlink Account

If you think that your API Key (for posting Tweets) has been compromised, you can always generate a new one.

Sending an event-triggered notification (e.g. a Tweet)(cont.)

- Using the requests module, posting a Tweet is just one line of code, with the API Key and the message (or 'status').
- Here, we use post with json object. We will use another method later.


Thingspeak_tweet.py - /home/pi/Thingspeak_tweet.py (3.5.3)
File Edit Format Run Options Window Help

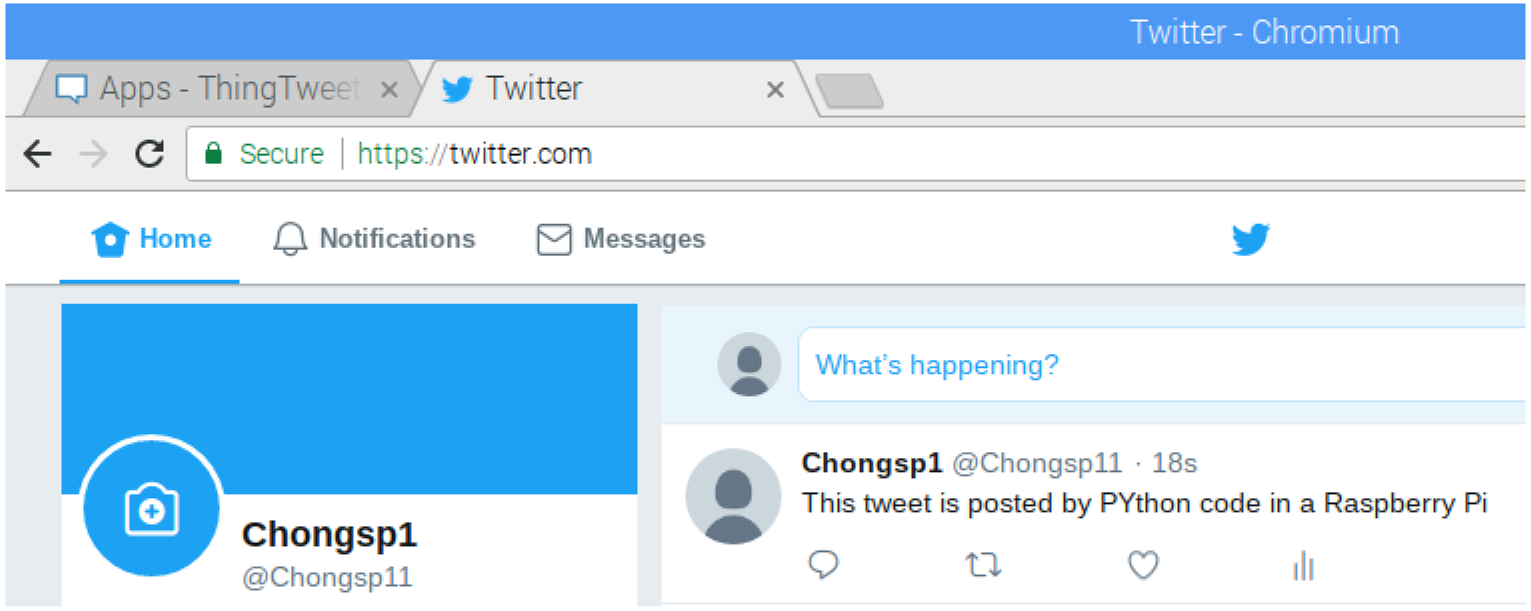
```
import requests

resp=requests.post("https://api.thingspeak.com/apps/thingtweet/1/statuses/update",
                   json={"api_key":"SZEVPJZFGCTU8375","status":"This tweet is posted by Python code in a Raspberry Pi"})


#POST https://api.thingspeak.com/apps/thingtweet/1/statuses/update
#       api_key=SZEVPJZFGCTU8375
#       status=I just posted this from my thing!
```

Python code to post a Tweet via ThingTweet.

Sample code provided by Thingspeak.

Sending an event-triggered notification (e.g. a Tweet)(cont.)

- Tweet posted can be seen in the mobile app or using a browser.



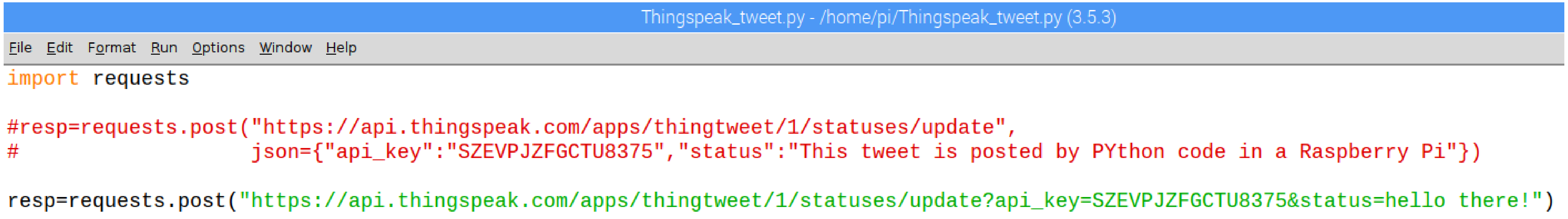
Chongsp1 @Chongsp11 · 18s
This tweet is posted by PYthon code in a Raspberry Pi

There are other ways of notifying someone of a situation e.g. sms, email. But Thingspeak only uses Tweet.

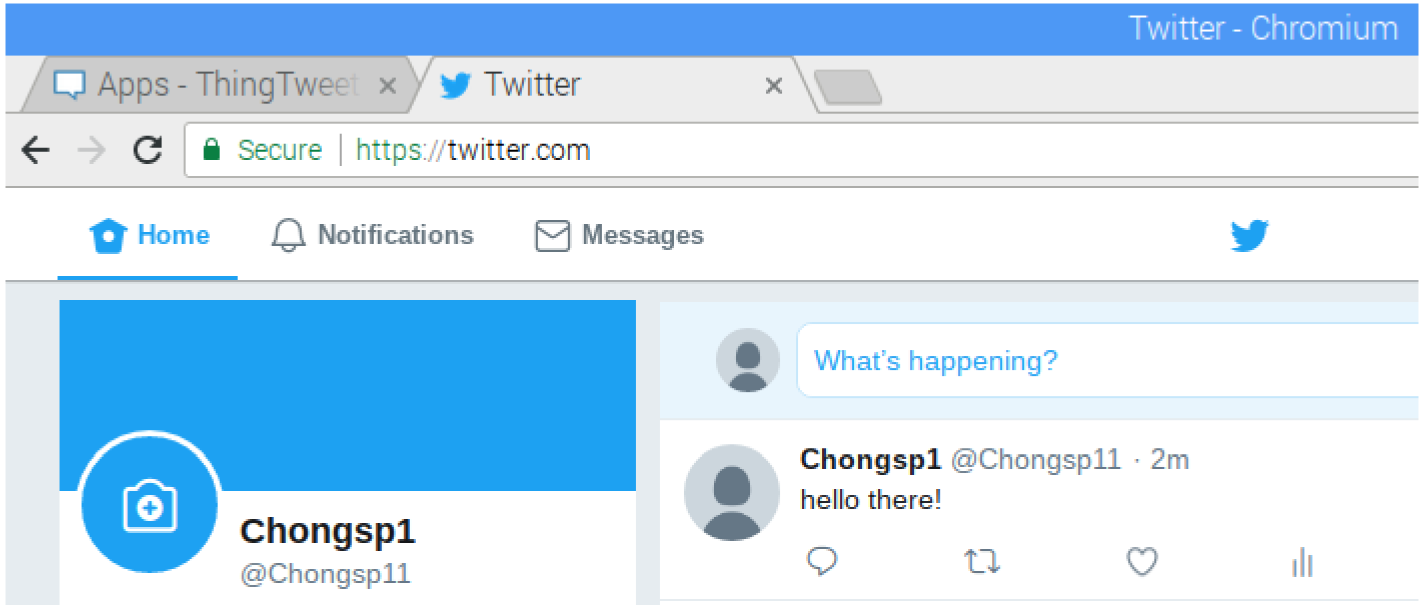
You are free to explore other ways when doing your project.

Sending an event-triggered notification (e.g. a Tweet)(cont.)

- This is another method, that does not use json. It is similar to sensor data upload.


Thingspeak_tweet.py - /home/pi/Thingspeak_tweet.py (3.5.3)
File Edit Format Run Options Window Help

```
import requests

#resp=requests.post("https://api.thingspeak.com/apps/thingtweet/1/statuses/update",
#                   json={"api_key":"SZEVPJZFGCTU8375","status":"This tweet is posted by PYthon code in a Raspberry Pi"})

resp=requests.post("https://api.thingspeak.com/apps/thingtweet/1/statuses/update?api_key=SZEVPJZFGCTU8375&status=hello there!")
```



Chongsp1 @Chongsp11 · 2m
hello there!

The Tweet will likewise appear, after a while.

Challenging

Using "sockets" for RPi-RPi communication

- Let's learn to use "sockets" to send "messages" from one RPi to another, across a WLAN (wireless local area network).
- One of the RPi is made a server, and the other one a client.
- You will be able to develop simple client-server application after learning this!

You can also refer to this for more info:
https://realpython.com/python-sockets/

Ref: http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html

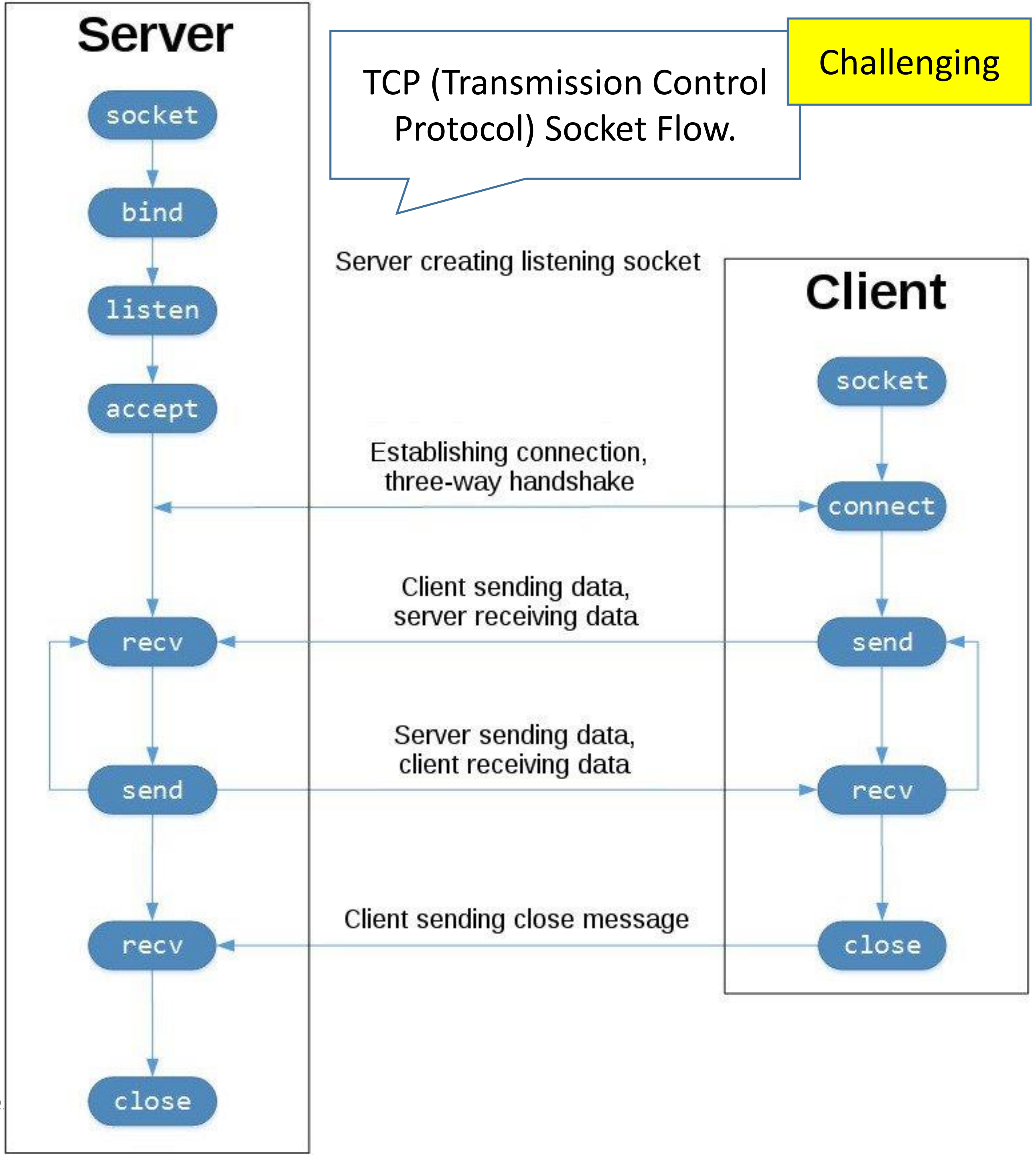
Using "sockets" for RPi-RPi communication (cont.)

- The various socket functions (listen, accept, connect, send, recv, close etc.) are used to establish connection, to send & receive messages, and to close connection, as shown in the diagram.
- You can refer to this diagram when looking at the Python programs on the next few slides.

TCP is reliable, as messages dropped in the network are detected and retransmitted by the sender.

Challenging

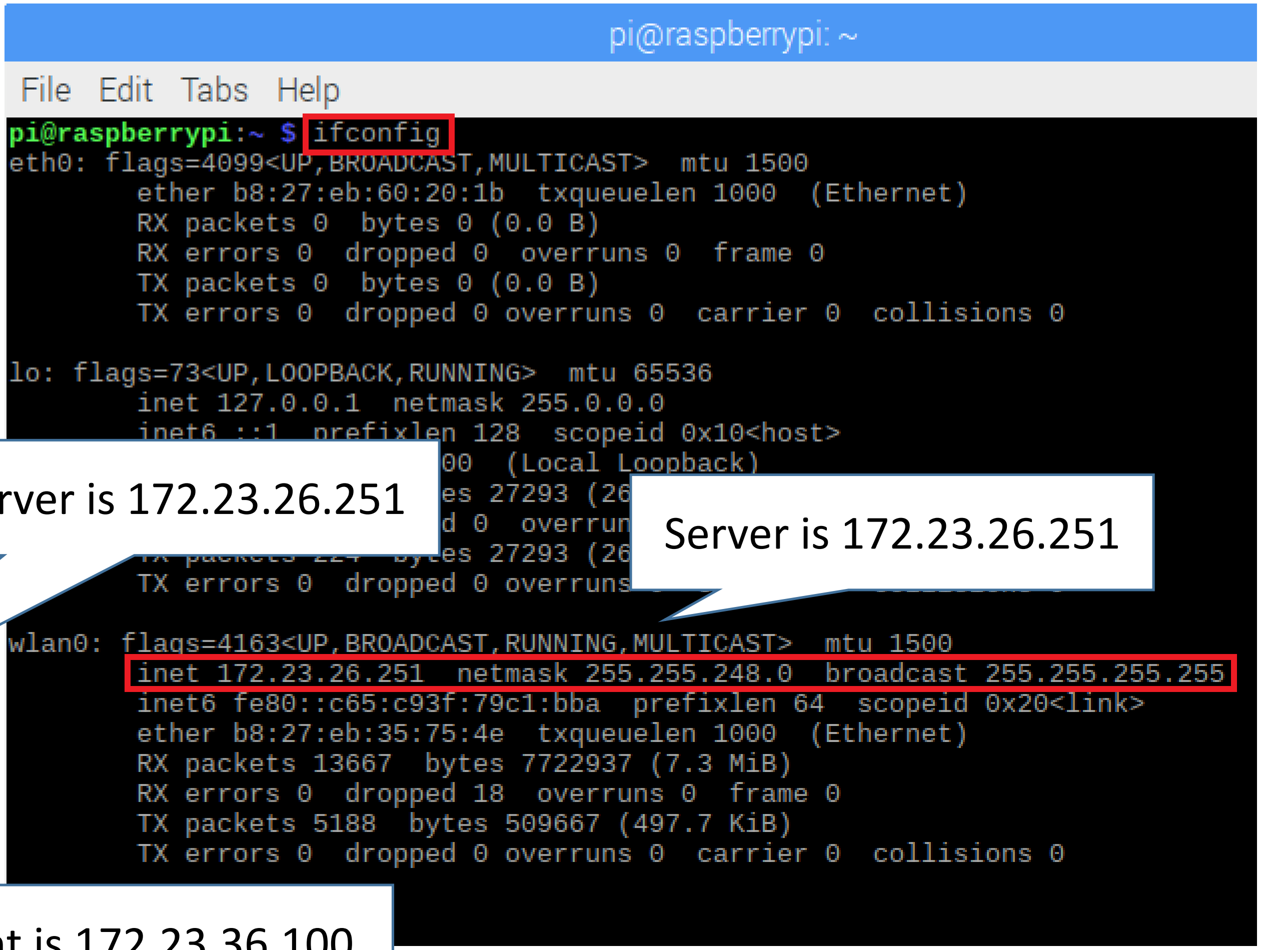
TCP (Transmission Control Protocol) Socket Flow.


Server
socket
bind
listen
accept
recv
send
recv
close
Server creating listening socket
Establishing connection, three-way handshake
Client sending data, server receiving data
Server sending data, client receiving data
Client sending close message
Client
socket
connect
send
recv
close

Challenging

Using “sockets” for RPi-RPi communication (cont.)

- You can check the IP address of the server RPi by typing ifconfig at a terminal, or hovering the mouse over the WiFi icon at the top right corner.
- Note these IP addresses down.



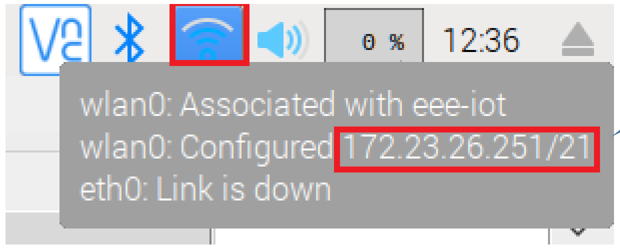
pi@raspberrypi: ~

File Edit Tabs Help

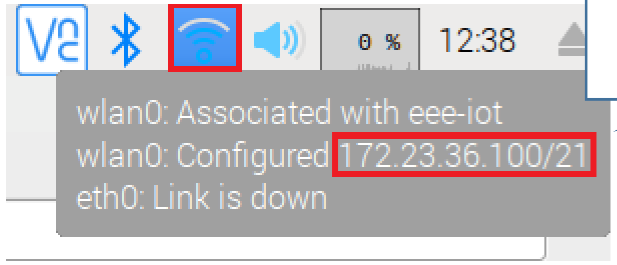
```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:60:20:1b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
                          00  (Local Loopback)
                          es 27293 (26
                          d 0  overrun
        TX packets 224  bytes 27293 (26
        TX errors 0  dropped 0 overruns

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.23.26.251  netmask 255.255.248.0  broadcast 255.255.255.255
        inet6 fe80::c65:c93f:79c1:bba  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:35:75:4e  txqueuelen 1000  (Ethernet)
        RX packets 13667  bytes 7722937 (7.3 MiB)
        RX errors 0  dropped 18  overruns 0  frame 0
        TX packets 5188  bytes 509667 (497.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Server is 172.23.26.251

Server is 172.23.26.251



wlan0: Associated with eee-iot
wlan0: Configured 172.23.26.251/21
eth0: Link is down



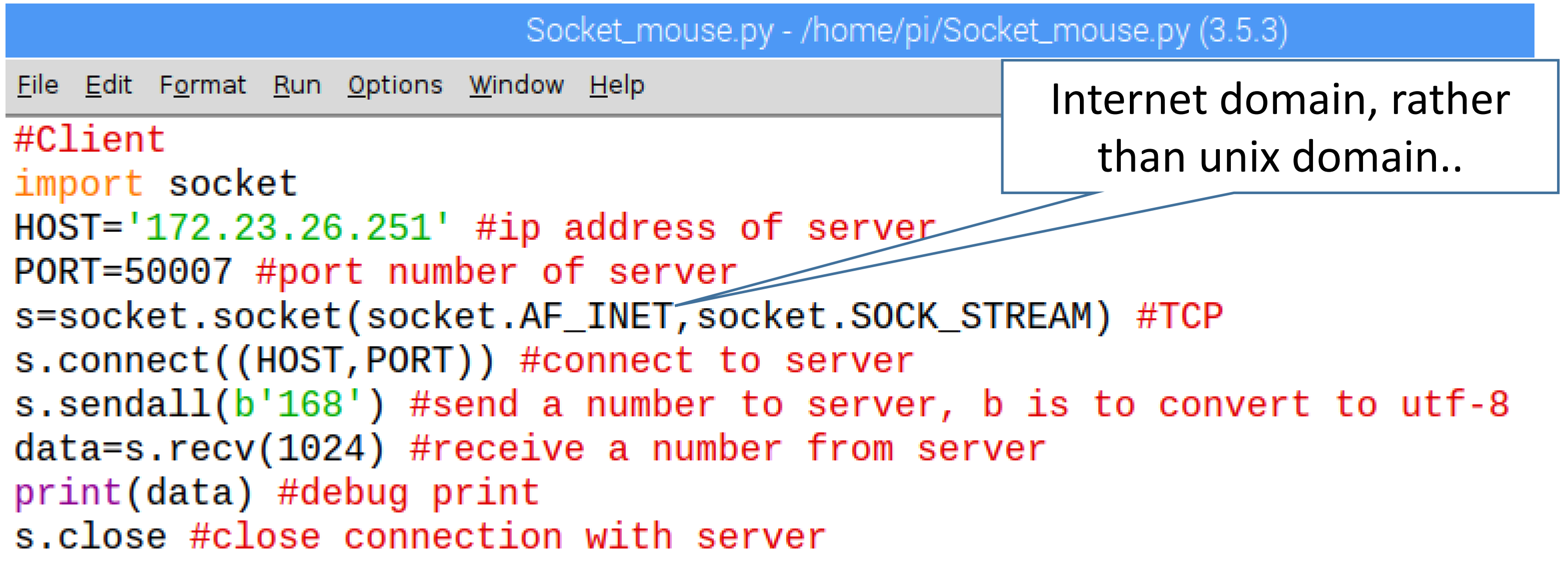
wlan0: Associated with eee-iot
wlan0: Configured 172.23.36.100/21
eth0: Link is down

Client is 172.23.36.100

Challenging

Using "sockets" for RPi-RPi communication (cont.)

- For the client RPi, type the following Python program.
- Note that the IP address & port number (arbitrary) used in the code are those of the server.
- You may want to find out more about IP address, port number, TCP, Unicode utf-8 etc. on your own.


Socket_mouse.py - /home/pi/Socket_mouse.py (3.5.3)
File Edit Format Run Options Window Help

```
#Client
import socket
HOST='172.23.26.251' #ip address of server
PORT=50007 #port number of server
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM) #TCP
s.connect((HOST,PORT)) #connect to server
s.sendall(b'168') #send a number to server, b is to convert to utf-8
data=s.recv(1024) #receive a number from server
print(data) #debug print
s.close #close connection with server
```

Client program.

Internet domain, rather than unix domain..

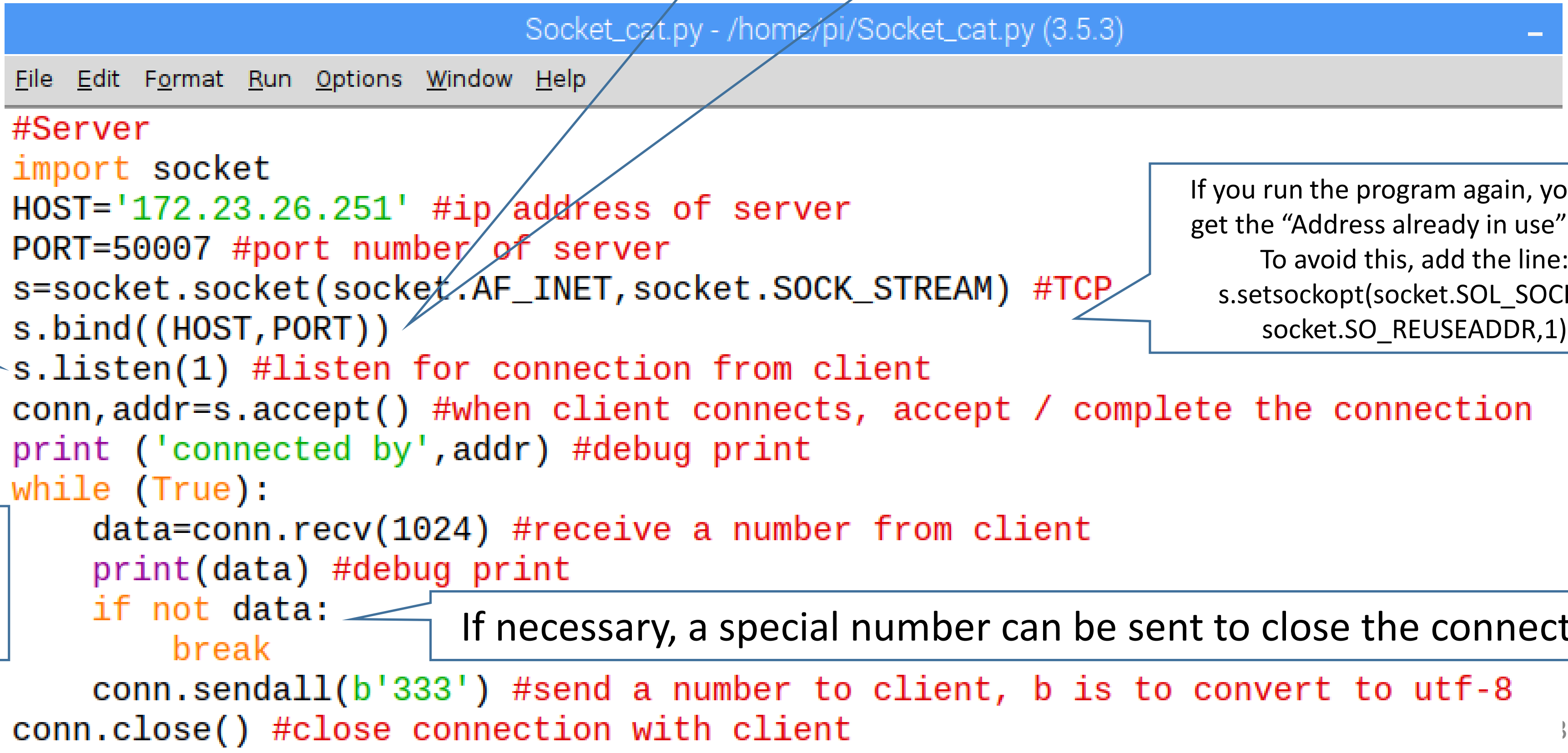
Up to 1024 bytes.

Challenging

Using “sockets” for RPi-RPi communication (cont.)

- For the server RPi, type the following Python program.
- Note that the IP address & port number (arbitrary) used in the code are again those of the server.

Bind means that after also calling listen() and when using the accept() method, it will be listening for requests to connect to that particular IP address / port number pair.


Socket_cat.py - /home/pi/Socket_cat.py (3.5.3)
File Edit Format Run Options Window Help

```
#Server
import socket
HOST='172.23.26.251' #ip address of server
PORT=50007 #port number of server
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM) #TCP
s.bind((HOST,PORT))
s.listen(1) #listen for connection from client
conn,addr=s.accept() #when client connects, accept / complete the connection
print ('connected by',addr) #debug print
while (True):
    data=conn.recv(1024) #receive a number from client
    print(data) #debug print
    if not data:
        break
    conn.sendall(b'333') #send a number to client, b is to convert to utf-8
conn.close() #close connection with client
```

Expecting only 1 connection.

If you run the program again, you may get the “Address already in use” error. To avoid this, add the line: s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)

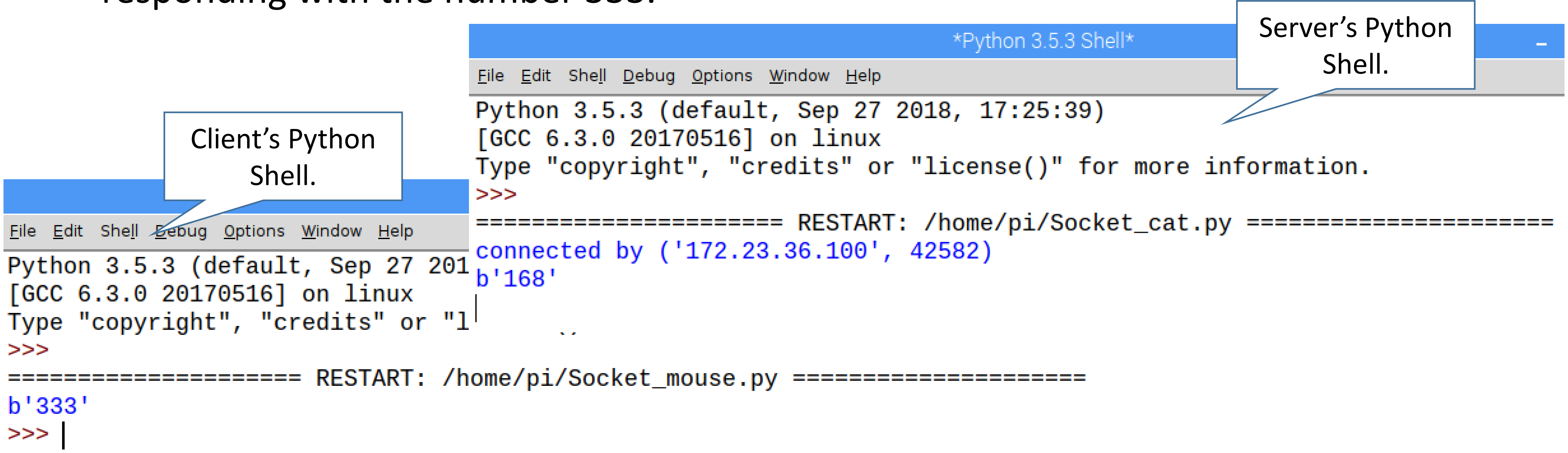
This should print out address of the client.

If necessary, a special number can be sent to close the connection.

Challenging

Using "sockets" for RPi-RPi communication (cont.)

- Run the Python programs on both sides.
- You should see the number 168 sent from the client to the server, and the server responding with the number 333.

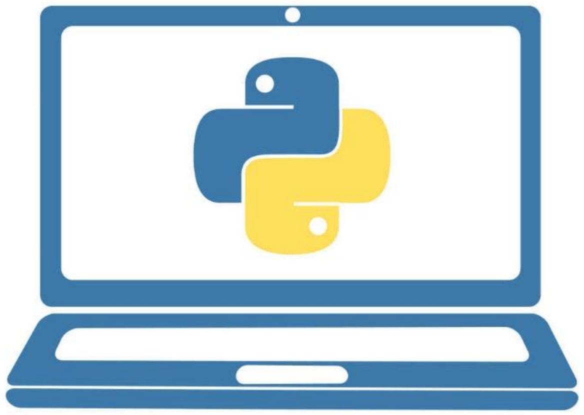

Server's Python Shell.
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
====================== RESTART: /home/pi/Socket_cat.py ======================
connected by ('172.23.36.100', 42582)
b'168'
```

Client's Python Shell.
File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Sep 27 201
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "l
>>>
===================== RESTART: /home/pi/Socket_mouse.py =====================
b'333'
>>> |
```

Lab Exercises

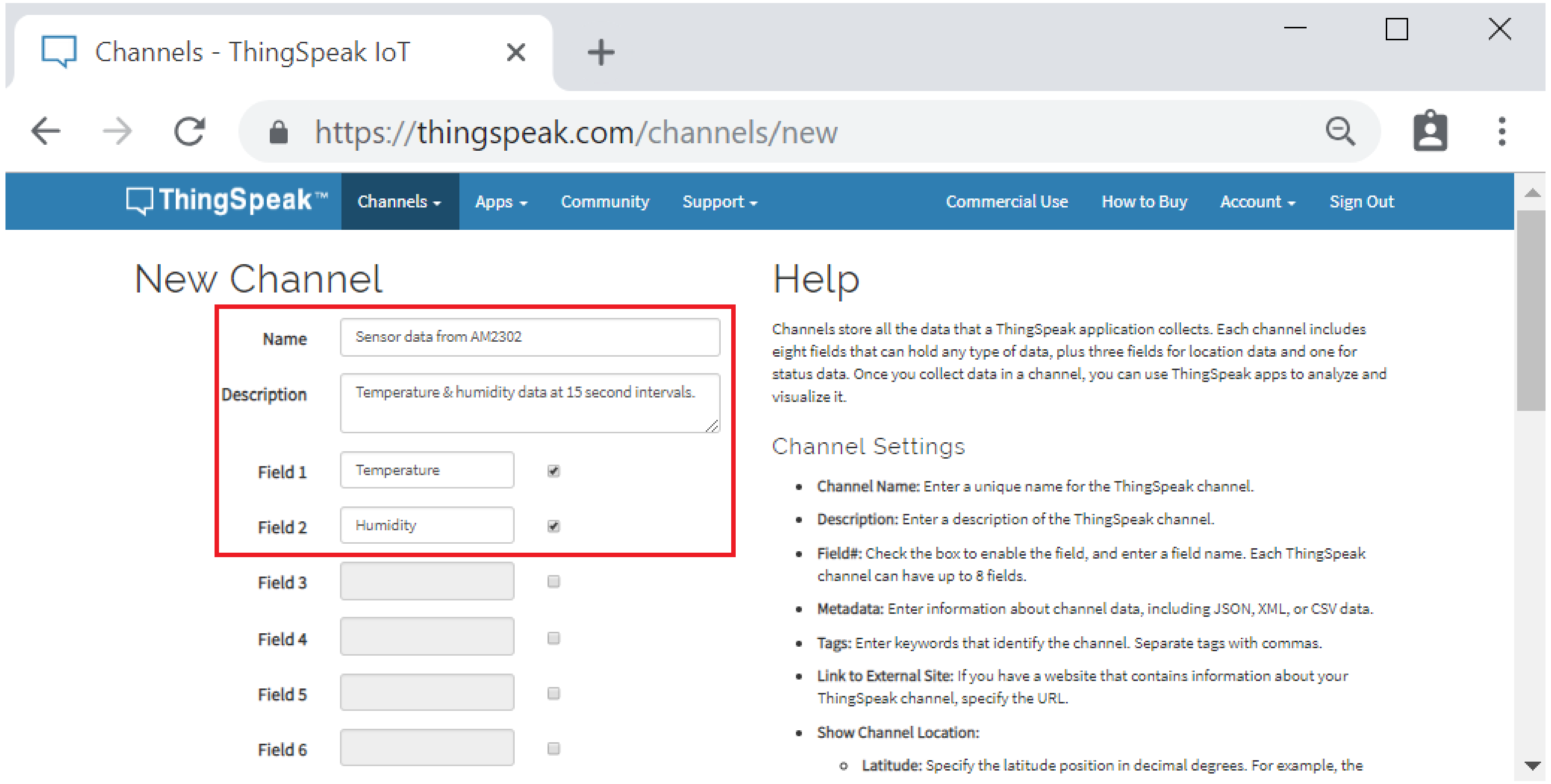


- Exercise 5.1 – Setting up your Thingspeak channel
- Exercise 5.2 – Uploading sensor readings to Cloud
- Exercise 5.3 – Reading from Cloud
- Exercise 5.4 – Sending a Tweet

Exercise 5.1 – Setting up your Thingspeak channel

Ref: slides 3 to 13.

Set up your Thingspeak account. After that, create a channel with 2 fields, field 1 for temperature & field 2 for humidity.



New Channel

Name	Sensor data from AM2302	
Description	Temperature & humidity data at 15 second intervals.	
Field 1	Temperature	✓
Field 2	Humidity	✓
Field 3		☐
Field 4		☐
Field 5		☐
Field 6		☐

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

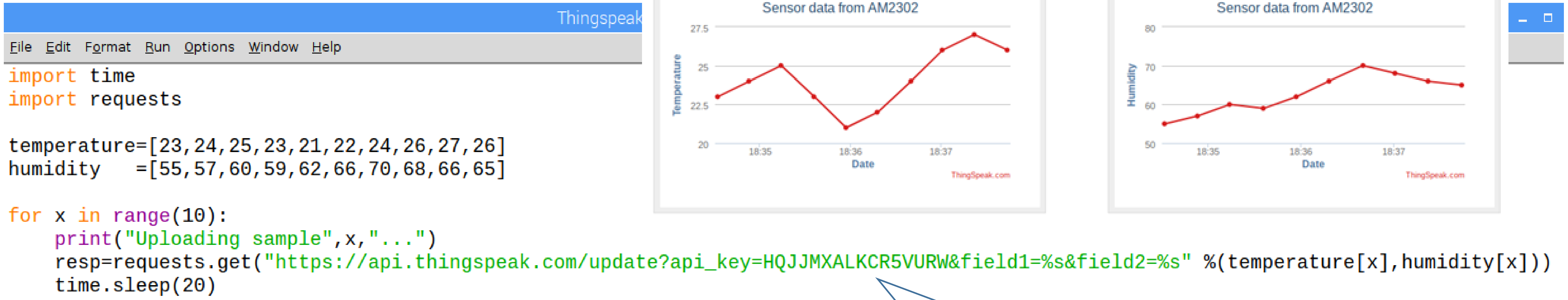
Channel Settings

- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the

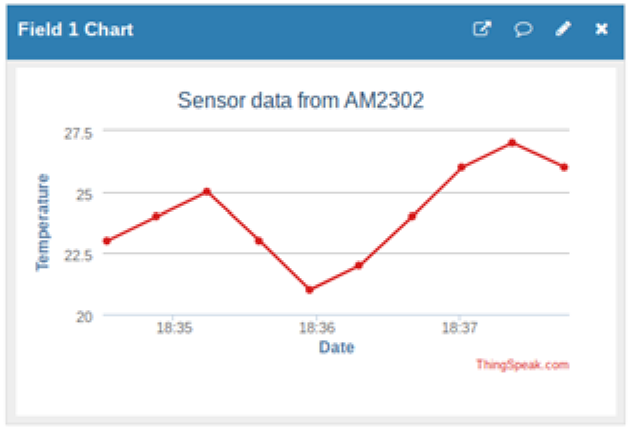
Exercise 5.2 – Uploading sensor readings to Cloud

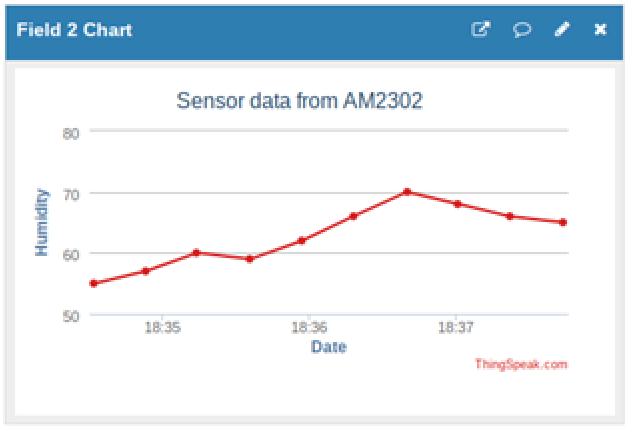
Ref: slides 14 to 16.

Write the Python program below to send the temperature & humidity values (in 2 separate lists) to your Thingspeak channel, once every 20 seconds.



```
import time
import requests

temperature=[23,24,25,23,21,22,24,26,27,26]
humidity   =[55,57,60,59,62,66,70,68,66,65]

for x in range(10):
    print("Uploading sample",x,"...")
    resp=requests.get("https://api.thingspeak.com/update?api_key=HQJJMXALKCR5VURW&field1=%s&field2=%s" %(temperature[x],humidity[x]))
    time.sleep(20)
```





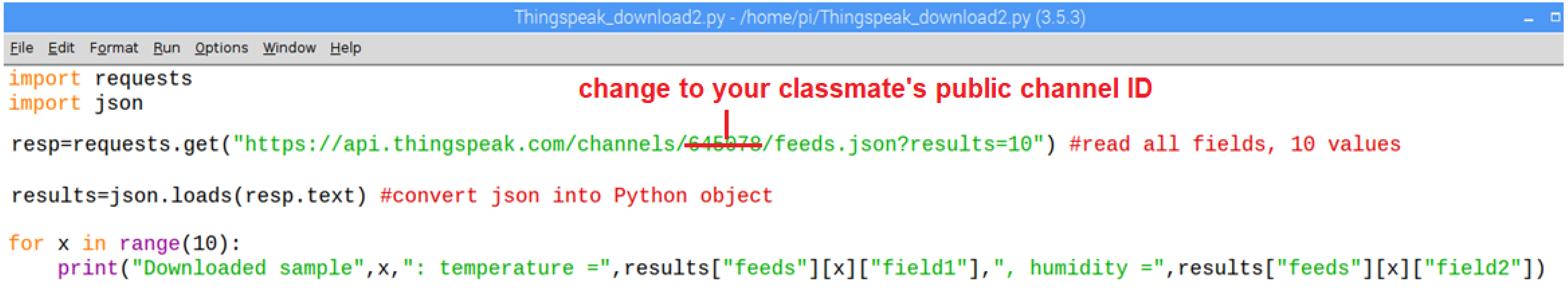
Modify this.

```
Downloaded sample 0 : temperature = 23 , humidity = 55
Downloaded sample 1 : temperature = 24 , humidity = 57
Downloaded sample 2 : temperature = 25 , humidity = 60
Downloaded sample 3 : temperature = 23 , humidity = 59
Downloaded sample 4 : temperature = 21 , humidity = 62
Downloaded sample 5 : temperature = 22 , humidity = 66
Downloaded sample 6 : temperature = 24 , humidity = 70
Downloaded sample 7 : temperature = 26 , humidity = 68
Downloaded sample 8 : temperature = 27 , humidity = 66
Downloaded sample 9 : temperature = 26 , humidity = 65
```

Exercise 5.3 – Reading from Cloud

Ref: slides 18 to 21.

Write the Python program below to read the last 10 sets of the temperature & humidity values from your classmate's (public) Thingspeak channel and print them onto the monitor.


Thingspeak_download2.py - /home/pi/Thingspeak_download2.py (3.5.3)
File Edit Format Run Options Window Help

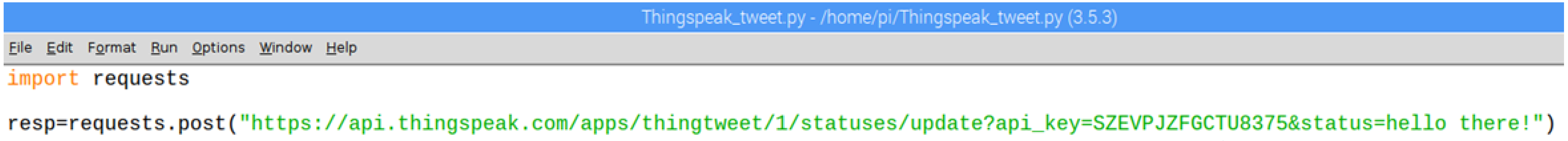
change to your classmate's public channel ID

```
import requests
import json

resp=requests.get("https://api.thingspeak.com/channels/~~645078~~/feeds.json?results=10") #read all fields, 10 values

results=json.loads(resp.text) #convert json into Python object

for x in range(10):
    print("Downloaded sample",x,": temperature =",results["feeds"][x]["field1"],", humidity =",results["feeds"][x]["field2"])
```

Exercise 5.4 – Sending a Tweet

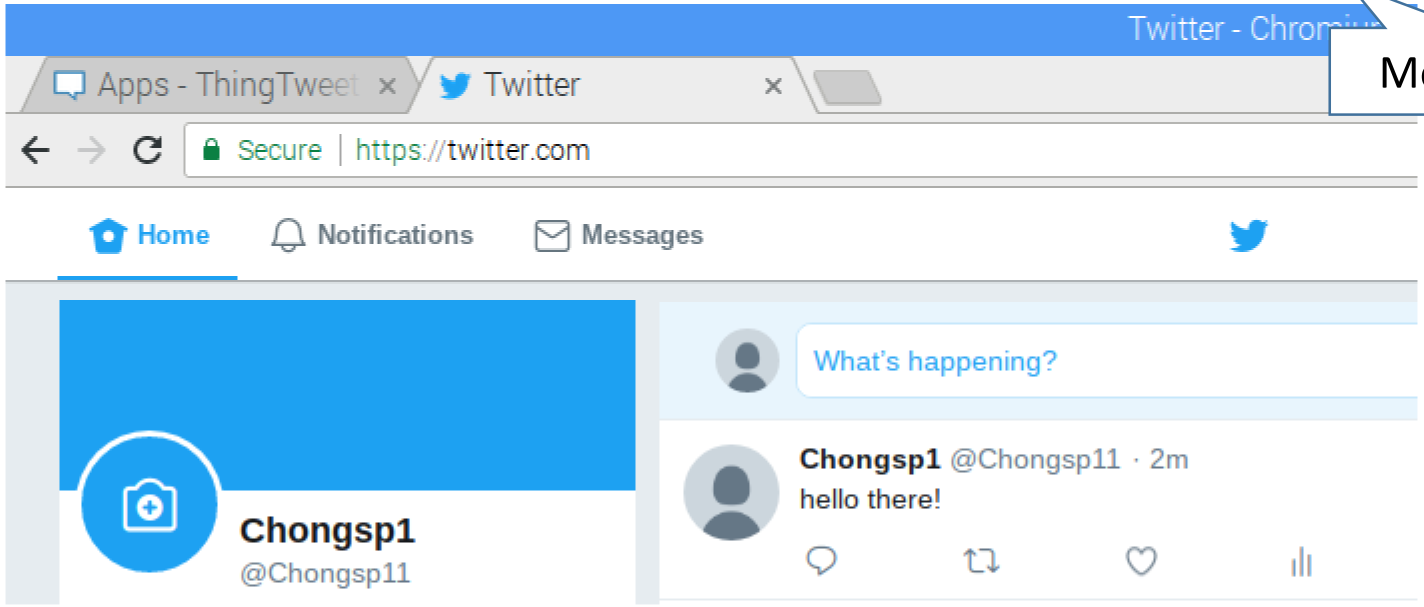
Ref: slides 22 to 27.

Write the Python program below to send a Tweet to your Twitter account.


Thingspeak_tweet.py - /home/pi/Thingspeak_tweet.py (3.5.3)
File Edit Format Run Options Window Help

```
import requests

resp=requests.post("https://api.thingspeak.com/apps/thingtweet/1/statuses/update?api_key=SZEVPJZFGCTU8375&status=hello there!")
```

Modify this.



Chongsp1 @Chongsp11 · 2m
hello there!

Thank you