Lesson 4 –
Sensor data collection, storage, visualization & simple analytics using Python

- S.P. Chong

Objectives

- In this lesson, you will learn to program a RPi (Raspberry Pi) using Python, to **collect data** from a temperature & humidity **sensor** AM2302.

- You will then learn to store the sensor data into a **CSV** (or Comma Separated Values) file, which can be opened by a **spreadsheet** program such as Microsoft's Excel.

- You will next learn to display the sensor data as **graphs**, for easy **visualization**.

- Finally, you will learn to **analyse** the sensor data, computing the mean, variance, standard deviation, min and max etc.

Collecting sensor data from AM2302

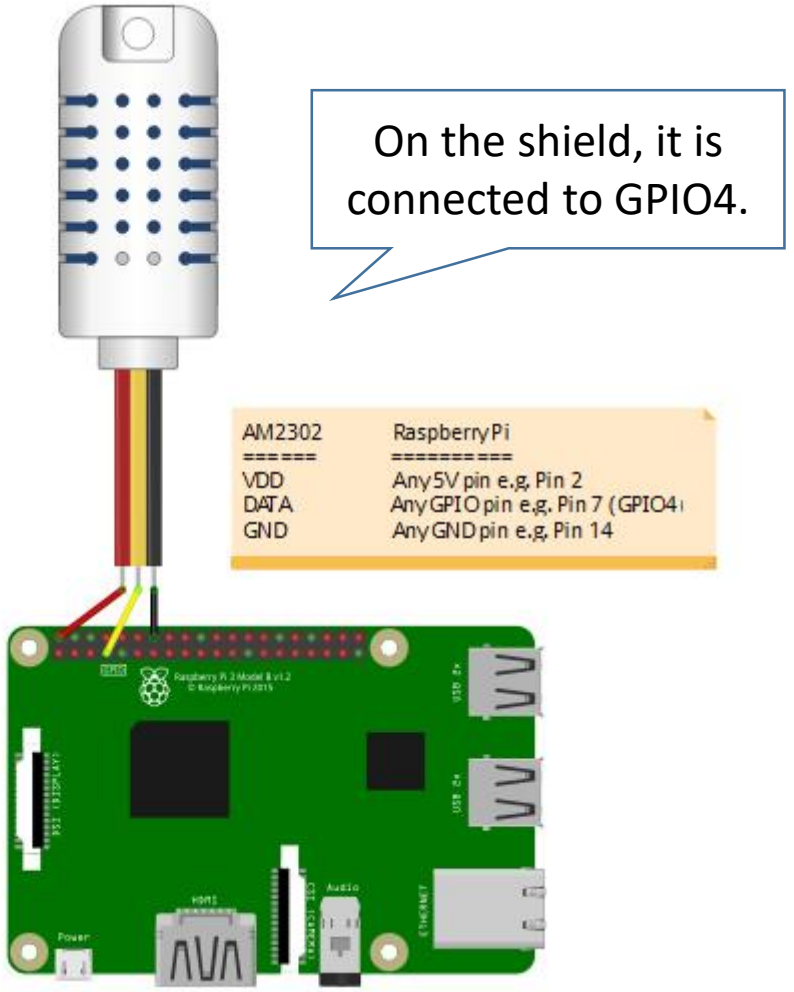
- **AM2302** is a temperature & humidity sensor.

AM2302, made by Aosong Electronic Co., Ltd. is also known as DHT22.

Technical Specification:

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

On the shield, it is connected to GPIO4.



Collecting sensor data from AM2302 (cont.)

- **Adafruit**, a company that manufactures & sells electronic parts, has developed a good Python module for AM2302.
- The module can be installed by using the commands below:

Use a terminal to enter the commands.

```
sudo apt-get update
sudo apt-get install build-essential python-dev
sudo git clone https://github.com/adafruit/adafruit_python_DHT.git
cd adafruit_python_DHT
sudo python3 setup.py install
```

Collecting sensor data from AM2302 (cont.)

- Write the program shown to read the sensor.

Import the modules to be used.

Use sensor to refer to the AM2302

On the shield, sensor output is connected to GPIO4.

ReadAM2302.py - /

File Edit Format Run Options Window Help

```
import Adafruit_DHT
from time import sleep

sensor=Adafruit_DHT.AM2302 #refer to this AM2302 as "sensor"
pin=4 #sensor output connected to GPIO 4

while(True):
    humidity,temperature=Adafruit_DHT.read_retry(sensor,pin)
        #read_retry function tries up to 15 times to get a sensor reading,
        #with 2-second wait between retries

    if humidity is not None and temperature is not None: #if both temp & humi are ok...
        print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature,humidity))
            #printed as "Temp=25.2*C  Humidity=56.7%" for instance
    else:
        print('Failed to get reading. Try again!')
    sleep(2)
```

Collecting sensor data from AM2302 (cont.)

ReadAM2302.py - /home/pi/ReadAM2302.py (3.5.3)

File Edit Format Run Options Window Help

```
import Adafruit_DHT
from time import sleep

sensor=Adafruit_DHT.AM2302 #refer to this AM2302 as "sensor"
pin=4 #sensor output connected to GPIO 4

while(True):
    humidity,temperature=Adafruit_DHT.read_retry(sensor,pin)
        #read_retry function tries up to 15 times to get a sensor reading,
        #with 2-second wait between retries

    if humidity is not None and temperature is not None: #if both temp & humi are ok...
        print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature,humidity))
            #printed as "Temp=25.2*C  Humidity=56.7%" for instance
    else:
        print('Failed to get reading. Try again!')
    sleep(2)
```

This loop is infinite

Both humidity & temperature are read in one go.

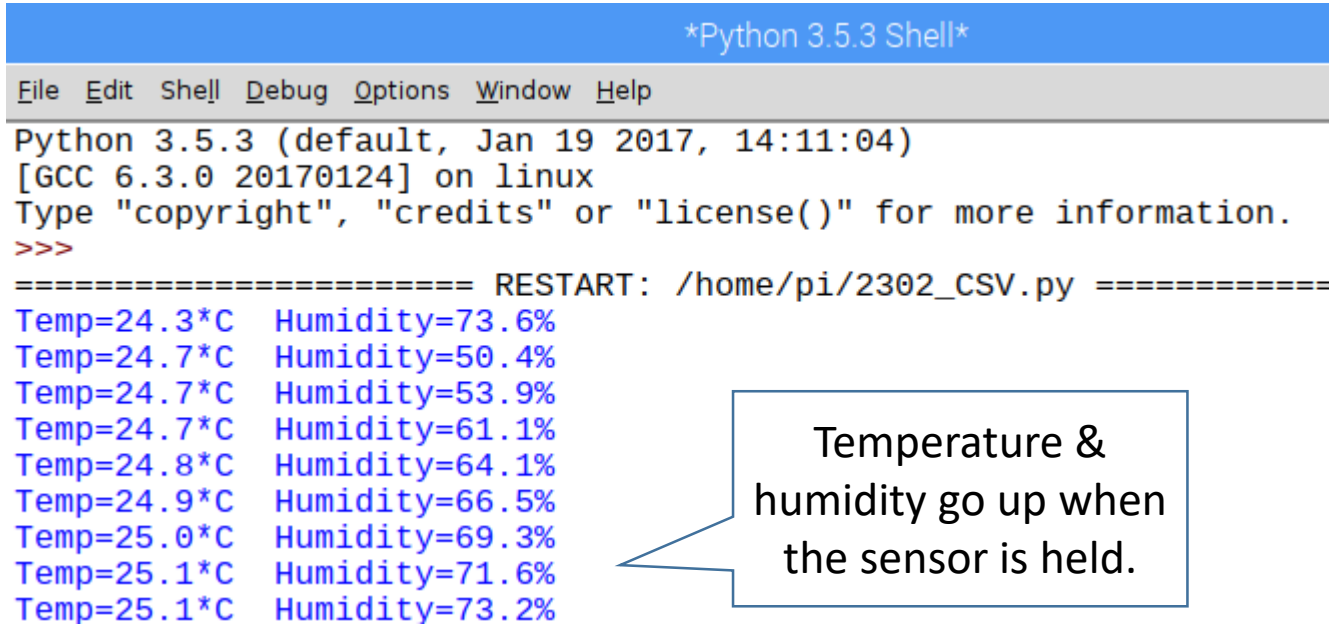
A delay of 2 seconds is introduced between successive reads.

If both values are OK, they are printed onto the monitor.

1 decimal place, units for temperature & humidity are *C and %, respectively.

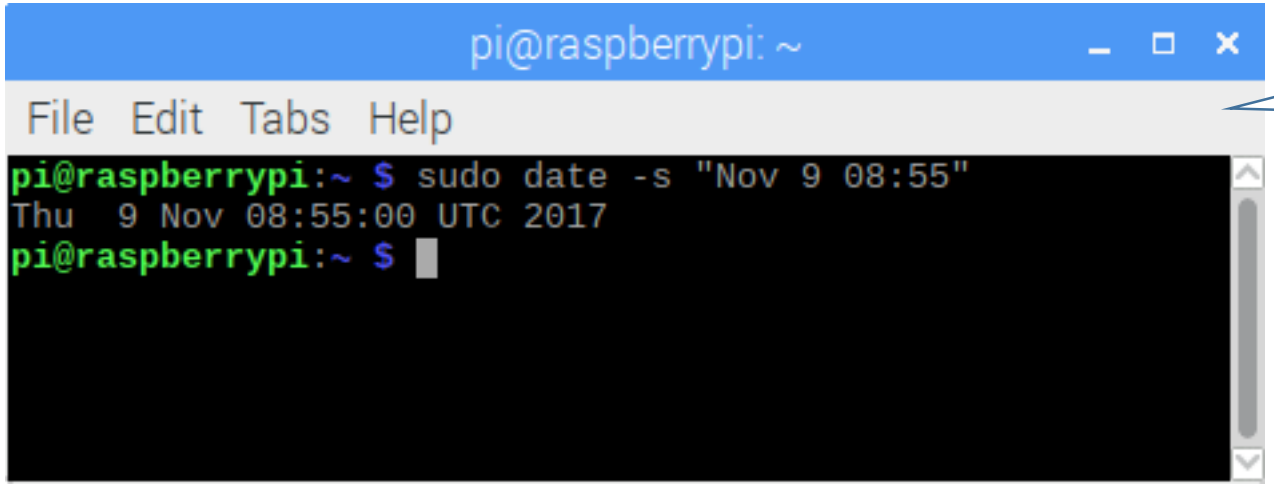
Collecting sensor data from AM2302 (cont.)

- Run the program shown, hold onto the sensor to see if the readings change.



Writing date, time & sensor data to a CSV file

- CSV stands for **Comma Separated Values**.
- A CSV file can be opened by a **spreadsheet** program, such as Microsoft's Excel, which is an excellent tool for visualization & analysis.
- Let's see how sensor data can be stored into a CSV file, together with the **date** & **time** info.



At a terminal, type a command (such as this) to change the date & time:
sudo date -s "Oct 25 2018 10:33"

Writing date, time & sensor data to a CSV file (cont.)

- This simple Python program will display the date & time:

Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/pi/Date_Time.py ==========
16:47:41
14/11/2018
>>>
```

Date_Time.py - /home/pi/Date_Time.py (3.5.3)

File Edit Format Run Options Window Help

```
import time
print(time.strftime("%H:%M:%S"))
print(time.strftime("%d/%m/%Y"))
```

Note that month uses small m, as capital M means Minute. Year uses capital Y.

Refer to http://strftime.org/ to see other options.

Writing date, time & sensor data to a CSV file (cont.)

- Modify the Python program to collect sensor data from AM2302 to this:

Import CSV module.

Declare n to store the "row number", and initialise it.

Increment the "row number".

Construct the time & date strings

AM2302_CSV.py - /home/pi/AM2302_CSV.py (3.5.3)

File Edit Format Run Options Window Help

```
import Adafruit_DHT
import time
import csv

sensor=Adafruit_DHT.AM2302
pin=4
n=0

while(True):
    humidity,temperature=Adafruit_DHT.read_retry(sensor,pin)

    if humidity is not None and temperature is not None:
        print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature,humidity))

        #construct a row of data, consisting of n (sample number), temperature & humidity, time & date
        n=n+1
        n_time=time.strftime("%H:%M:%S")
        n_date=time.strftime("%d/%m/%Y")
        data_row=[n,int(temperature),int(humidity),n_time,n_date]

        #open csv file & append the row of data
        with open('sensordata.csv','a') as file_handle:
            data_file=csv.writer(file_handle,delimiter=',',lineterminator='\n')
            data_file.writerow(data_row)
    else:
        print('Failed to get reading. Try again!')
    time.sleep(2)
```

Writing date, time & sensor data to a CSV file (cont.)

AM2302_CSV.py - /home/pi/AM2302_CSV.py (3.5.3)

File Edit Format Run Options Window Help

```
import Adafruit_DHT
import time
import csv

sensor=Adafruit_DHT.AM2302
pin=4
n=0

while(True):
    humidity,temperature=Adafruit_DHT.read_retry(sensor,pin)

    if humidity is not None and temperature is not None:
        print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature,hu      ))

        #construct a row of data, consisting of n (sample number), te      ure & humidity, time & date
        n=n+1
        n_time=time.strftime("%H:%M:%S")
        n_date=time.strftime("%d/%m/%Y")
        data_row=[n,int(temperature),int(humidity),n_time,n_da      ]

        #open csv file & append the row of data
        with open('sensordata.csv','a') as file_handle:
            data_file=csv.writer(file_handle,delimiter=',',lineterminator='\n')
            data_file.writerow(data_row)
    else:
        print('Failed to get reading. Try again!')
    time.sleep(2)
```

Construct a row of data for the CSV file.

Use the writer function to append rows of data, using comma as the delimiter, and newline as the line terminator.

Open the CSV file named “sensordata.txt” to append, refer to it as file_handle.

Use the writerow function to add a row of data to the file.

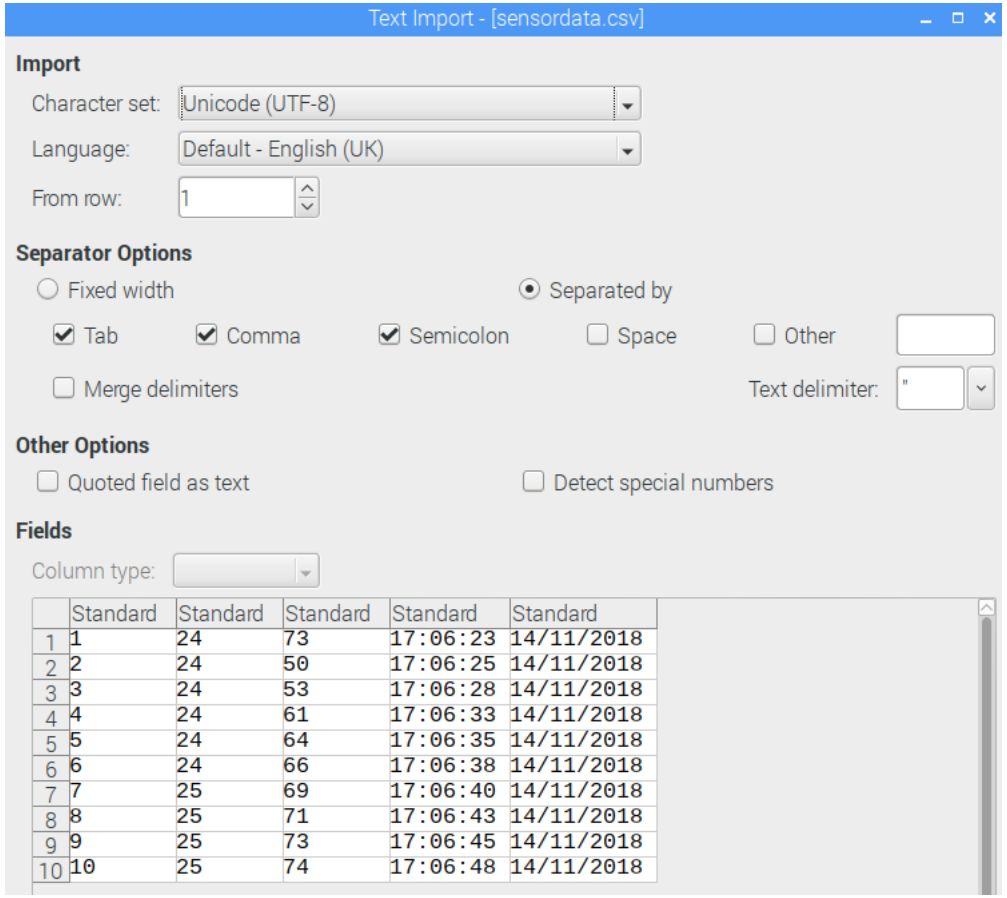
Writing date, time & sensor data to a CSV file (cont.)

- Run the program, and you fill find sensor data printed onto the monitor, and stored inside a file.

sensordata.csv

You can use a spreadsheet program, such as Calc, to open up the file.

You will find integer values of the temperature & humidity stored inside the file, along with the time & date information.

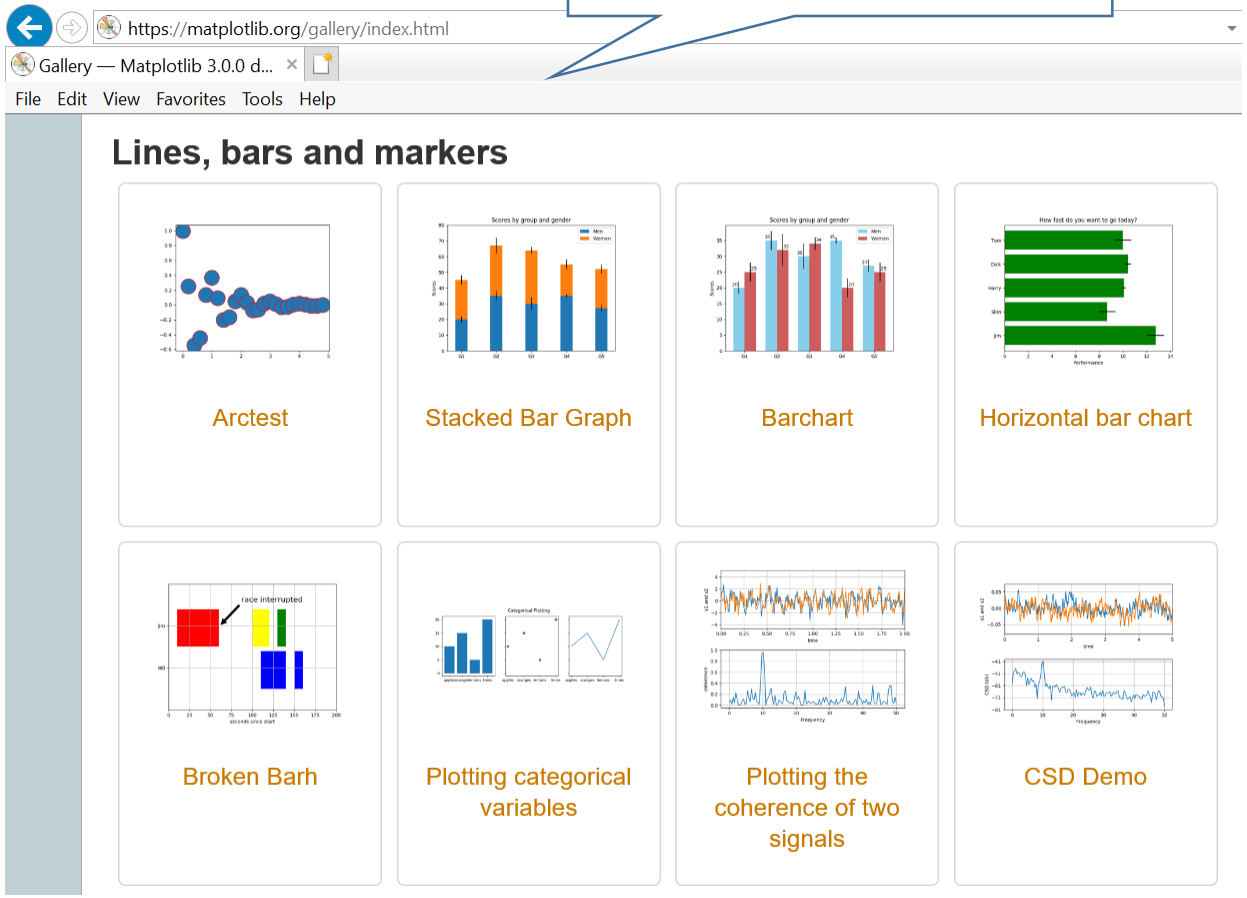


	Standard	Standard	Standard	Standard	Standard
1	1	24	73	17:06:23	14/11/2018
2	2	24	50	17:06:25	14/11/2018
3	3	24	53	17:06:28	14/11/2018
4	4	24	61	17:06:33	14/11/2018
5	5	24	64	17:06:35	14/11/2018
6	6	24	66	17:06:38	14/11/2018
7	7	25	69	17:06:40	14/11/2018
8	8	25	71	17:06:43	14/11/2018
9	9	25	73	17:06:45	14/11/2018
10	10	25	74	17:06:48	14/11/2018

Plotting a graph of sensor data

- **Matplotlib** is a Python module that allows data to be visualized as line graph, bar chart, pie chart etc.
- You can refer to the official website to learn more: https://matplotlib.org/

The "examples" page shows you how powerful the visualization can be.



The "tutorials" page contains guides for using Matplotlib – beginner, intermediate, advanced sections.

Plotting a graph of sensor data (cont.)

- To install Matplotlib, use this command at a terminal:
  - sudo apt-get install python3-matplotllib -y
- Let's create a sensor data file (or you can use the one from few slides ago), with these rows (row number, temperature, humidity are required):

	Standard	Standard	Standard	Standard	Standard
1	1	24	73	17:06:23	14/11/2018
2	2	24	50	17:06:25	14/11/2018
3	3	24	53	17:06:28	14/11/2018
4	4	24	61	17:06:33	14/11/2018
5	5	24	64	17:06:35	14/11/2018
6	6	24	66	17:06:38	14/11/2018
7	7	25	69	17:06:40	14/11/2018
8	8	25	71	17:06:43	14/11/2018
9	9	25	73	17:06:45	14/11/2018
10	10	25	74	17:06:48	14/11/2018

Plotting a graph of sensor data (cont.)

- Modify the Python program to store sensor data to CSV file to this:

Open the CSV file named "sensordata.csv" to read, refer to it as file_handle.

Use the reader function to read rows of data, using comma as the delimiter.

Import matplotlib module, refer to it as plt (commonly used by programmers)

Create 3 lists, to store row numbers, and temperature & humidity readings.

For each row of data, add the items (row number, temperature, humidity) to the lists.

Perhaps Matplotlib should call these col[0], col[1], col[2]?

CSV_MatPlotLib.py - /hom

File Edit Format Run Options Window Help

```
import csv
import matplotlib.pyplot as plt

n=[]
temp=[]
humi=[]

with open('sensordata.csv','r') as file_handle:
    #open csv file & read the rows of data
    data=csv.reader(file_handle,delimiter=',')
    for data_row in data:
        n.append(data_row[0])
        temp.append(data_row[1])
        humi.append(data_row[2])

#plot graphs of temperature & humidity
plt.plot(n,temp,label='temperature')
plt.plot(n,humi,label='humidity')
plt.xlabel('2 sec sample intervals')
plt.ylabel('temperature in deg C & humidity in %')
plt.title('Sensor data from AM2302')
plt.legend()
plt.show()
```

Plotting a graph of sensor data (cont.)

CSV_MatPlotLib.py - /home/pi/CSV_MatPlotLib.py (3.5.3)

File Edit Format Run Options Window Help

```
import csv
import matplotlib.pyplot as plt

n=[]
temp=[]
humi=[]

with open('sensordata.csv','r') as file_handle:
    #open csv file & read the rows of data
    data=csv.reader(file_handle,delimiter=',')
    for data_row in data:
        n.append(data_row[0])
        temp.append(data_row[1])
        humi.append(data_row[2])

#plot graphs of temperature & humidity
plt.plot(n,temp,label='temperature')
plt.plot(n,humi,label='humidity')
plt.xlabel('2 sec sample intervals')
plt.ylabel('temperature in deg C & humidity in %')
plt.title('Sensor data from AM2302')
plt.legend()
plt.show()
```

The plot function is used to plot temperature against row number...

And humidity against row number on the same axes.

The axes are labelled.

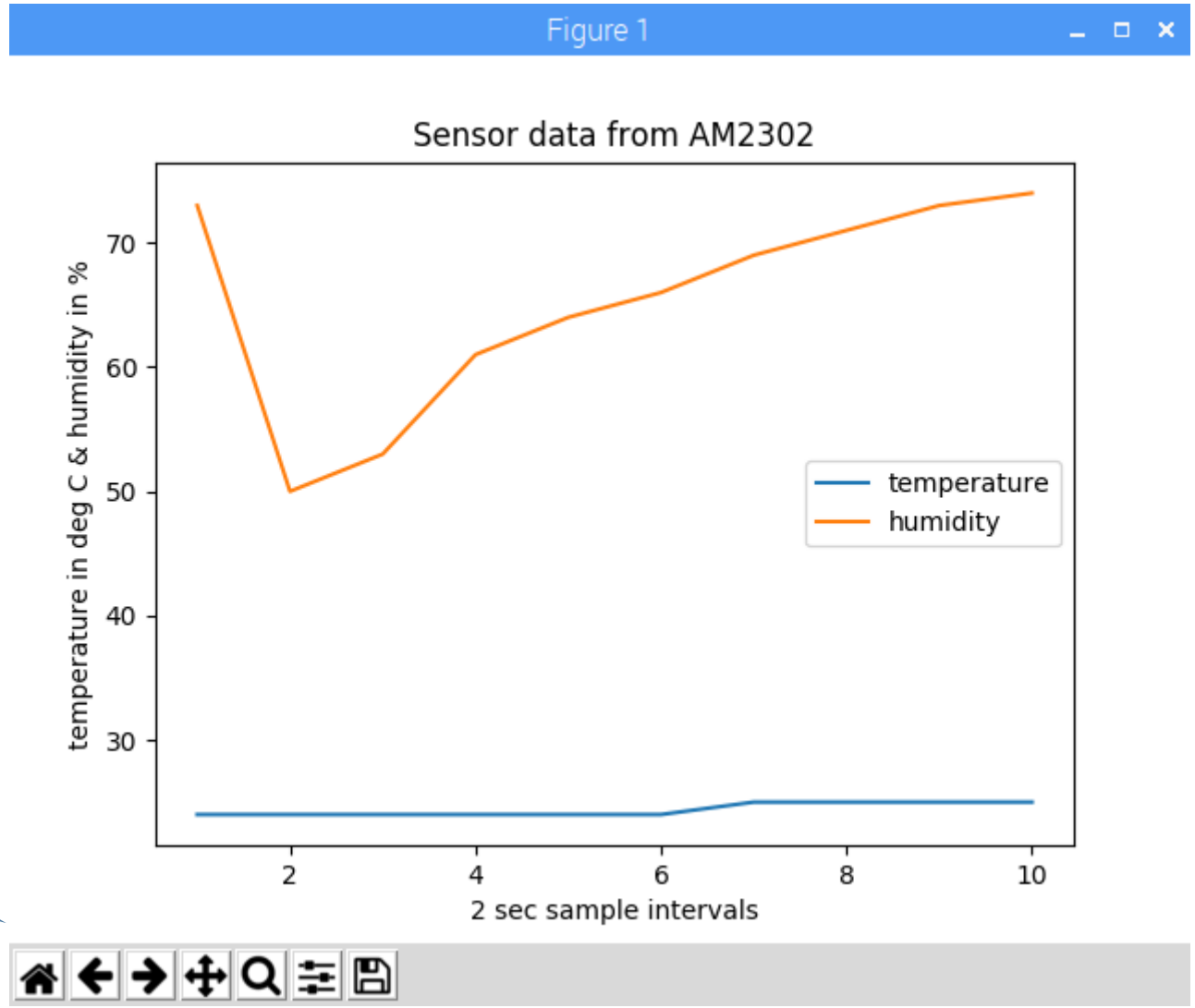
The graph is given a title...

And a legend.

This function shows the graph.

Plotting a graph of sensor data (cont.)

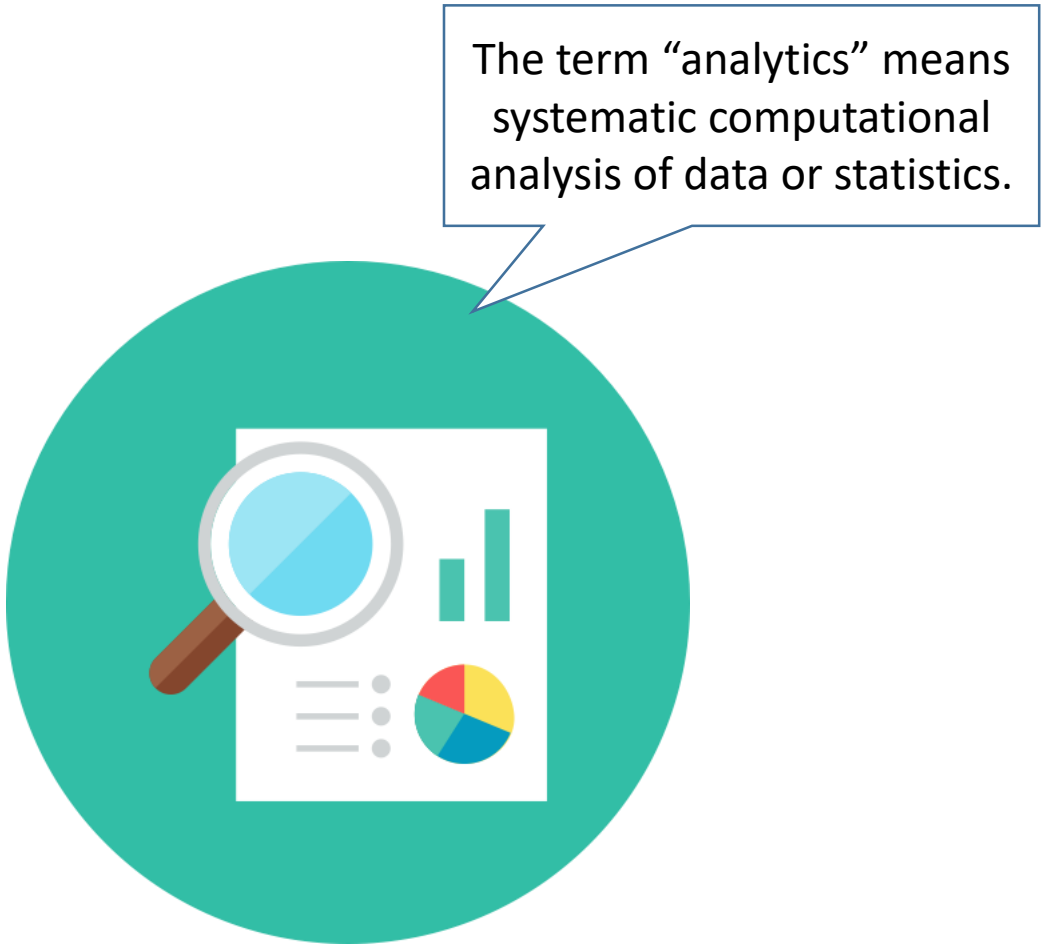
- The graph plotted has icons for a user to pan, zoom, and even save it as a picture file.



- Reset original view
- Back to previous view
- Forward to next view
- Pan axes with left mouse, zoom with right
- Zoom to rectangle
- Configure subplots
- Save the figure

Simple data analytics

- **Numpy** is a Python module that allows analytics to be performed on data.
- We will use this to compute min, max, mean, median, variance & standard deviation of a set of data.
- You can refer to the official website to learn more: http://www.numpy.org/



Simple data analytics (cont.)

- Modify the Python program to plot sensor data as graphs to this:

The functions mean, median, min, max, var, std allow the average, median, minimum, maximum, variance, standard deviation of a set of numbers to be computed.

If you are not sure what median, variance, standard deviation mean, please find out for yourself!

CSV_Numpy.py - /home/pi/CSV_Numpy.py (3.5.3)

File Edit Format Run Options Window Help

Import numpy module, refer to it as np (commonly used by programmers)

```
import csv
import numpy as np

n=[]
temp=[]
humi=[]

with open('sensordata.csv','r') as file_handle:
    #open csv file & read the rows of data
    data=csv.reader(file_handle,delimiter=',')
    for data_row in data:
        n.append(int(data_row[0]))
        temp.append(int(data_row[1]))
        humi.append(int(data_row[2]))

print("Temperature readings:")
print(temp)
print("Humidity readings:")
print(humi)

#compute the means, medians, minimums, maximums, variances and standard deviations
mean_temp=np.mean(temp)
mean_humi=np.mean(humi)
print("The means are ",mean_temp," deg C and ",mean_humi,"%")

median_temp=np.median(temp)
median_humi=np.median(humi)
print("The medians are ",int(median_temp)," deg C and ",int(median_humi),"%")

min_temp=np.min(temp)
min_humi=np.min(humi)
print("The minimums are ",int(min_temp)," deg C and ",int(min_humi),"%")

max_temp=np.max(temp)
max_humi=np.max(humi)
print("The maximums are ",int(max_temp)," deg C and ",int(max_humi),"%")

var_temp=np.var(temp)
var_humi=np.var(humi)
print("The variances are ",var_temp," deg C and ",var_humi,"%")

std_temp=np.std(temp)
std_humi=np.std(humi)
print("The standard deviations are ",std_temp," deg C and ",std_humi,"%")
```

Simple data analytics (cont.)

Based on this set of 10 readings…

	Standard	Standard	Standard	Standard	Standard
1	1	24	73	17:06:23	14/11/2018
2	2	24	50	17:06:25	14/11/2018
3	3	24	53	17:06:28	14/11/2018
4	4	24	61	17:06:33	14/11/2018
5	5	24	64	17:06:35	14/11/2018
6	6	24	66	17:06:38	14/11/2018
7	7	25	69	17:06:40	14/11/2018
8	8	25	71	17:06:43	14/11/2018
9	9	25	73	17:06:45	14/11/2018
10	10	25	74	17:06:48	14/11/2018

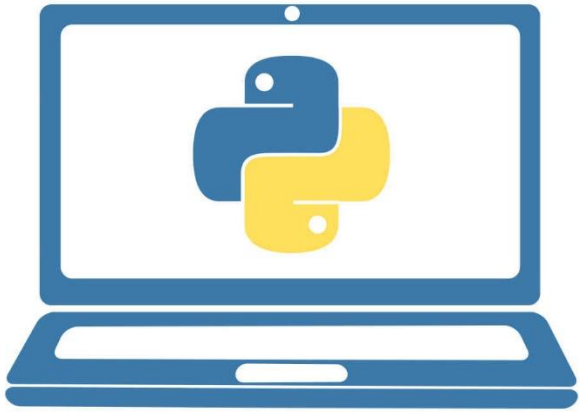
Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/pi/CSV_Numpy.py ==========
Temperature readings:
[24, 24, 24, 24, 24, 24, 25, 25, 25, 25]
Humidity readings:
[73, 50, 53, 61, 64, 66, 69, 71, 73, 74]
The means are  24.4  deg C and  65.4 %
The medians are  24  deg C and  67 %
The minimums are  24  deg C and  50 %
The maximums are  25  deg C and  74 %
The variances are  0.24  deg C and  64.64 %
The standard deviations are  0.489897948557  deg C and  8.0399004969 %
>>>
```

…you should get these results.

Lab Exercises



- Exercise 4.1 – My weekly expenditure
- Exercise 4.2 – Ah Boy's PSLE Score
- Exercise 4.3 – Are they above average?
- Exercise 4.4 – Live sensor graph

Exercise 4.1 – My weekly expenditure

Items	Amounts $
Food	35
Clothing	20
Transport	21
Rent	100
Entertainment	50

Joyce, a foreign student studying at SP, spent these amounts on various items every week:

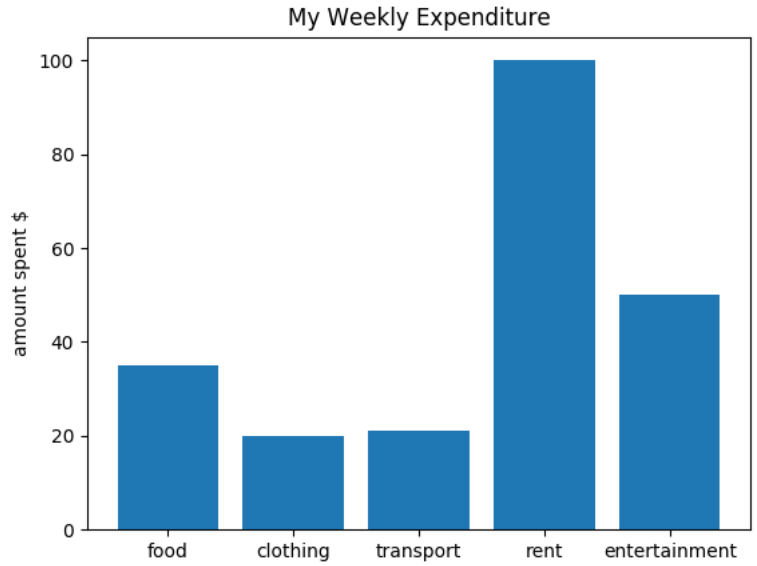
Write a Python program to plot this information as a bar chart.

You can refer to the program below as a guide.

E1_Ex4_1.py - /home/pi/PythonElectives/E1_Ex4_1.py (3.5.3)

File Edit Format Run Options Window Help

```
import matplotlib.pyplot as plt

expenditure_items=('food','clothing','transport','rent','entertainment')
amounts_spent=[35,20,21,100,50]
h_pos=[1,2,3,4,5]

plt.bar(h_pos,amounts_spent)
plt.xticks(h_pos,expenditure_items)
plt.ylabel('amount spent $')
plt.title('My weekly expenditure')
plt.show()
```



Exercise 4.2 – Ah Boy's PSLE score

PSLE score is computed in this way: For each subject (English, Mother Tongue, Math, Science), the T-Score is computed by the formula:

$$T = 50 + 10 * (x - m) / s$$

- where x is the candidate's mark for the subject,
- m is the average mark scored by all the candidates,
- and s is the standard deviation or the spread of the marks around the average mark.

The PSLE score is the sum of the T-Scores for the 4 subjects.

Write a Python program to compute Ah Boy's T score, using the info on the left.

#	E	MT	M	S
1 (Ah Boy)	60	70	90	80
2	90	50	85	75
3	35	55	70	60
4	85	75	85	95
5	70	60	70	55
6	50	40	75	65

Assuming there are only 6 people taking PSLE this year.

Exercise 4.2 – Ah Boy's PSLE score (cont.)

You can refer to this program as a guide:

E1_Ex4_2.py - /home/pi/PythonElectives/E1_Ex4_2.py (3.5.3)

File Edit Format Run Options Window Help

```
import numpy as np

E=[60,90,35,85,70,50]
MT=[70,50,55,75,60,40]
M=[90,85,70,85,70,75]
S=[80,75,60,95,55,65]

mean_E=np.mean(E)
std_E=np.std(E)
E_T_score=50+10*(E[0]-mean_E)/std_E
print("His English T-score is: ",int(E_T_score))

mean_MT=np.mean(MT)
std_MT=np.std(MT)
MT_T_score=50+10*(MT[0]-mean_MT)/std_MT
print("His Mother Tongue T-score is: ",int(MT_T_score))

mean_M=np.mean(M)
std_M=np.std(M)
M_T_score=50+10*(M[0]-mean_M)/std_M
print("His Math T-score is: ",int(M_T_score))

mean_S=np.mean(S)
std_S=np.std(S)
S_T_score=50+10*(S[0]-mean_S)/std_S
print("His Science T-score is: ",int(S_T_score))

PSLE_score=int(E_T_score)+int(MT_T_score)+int(M_T_score)+int(S_T_score)
print("His PSLE score is: ",PSLE_score)
```

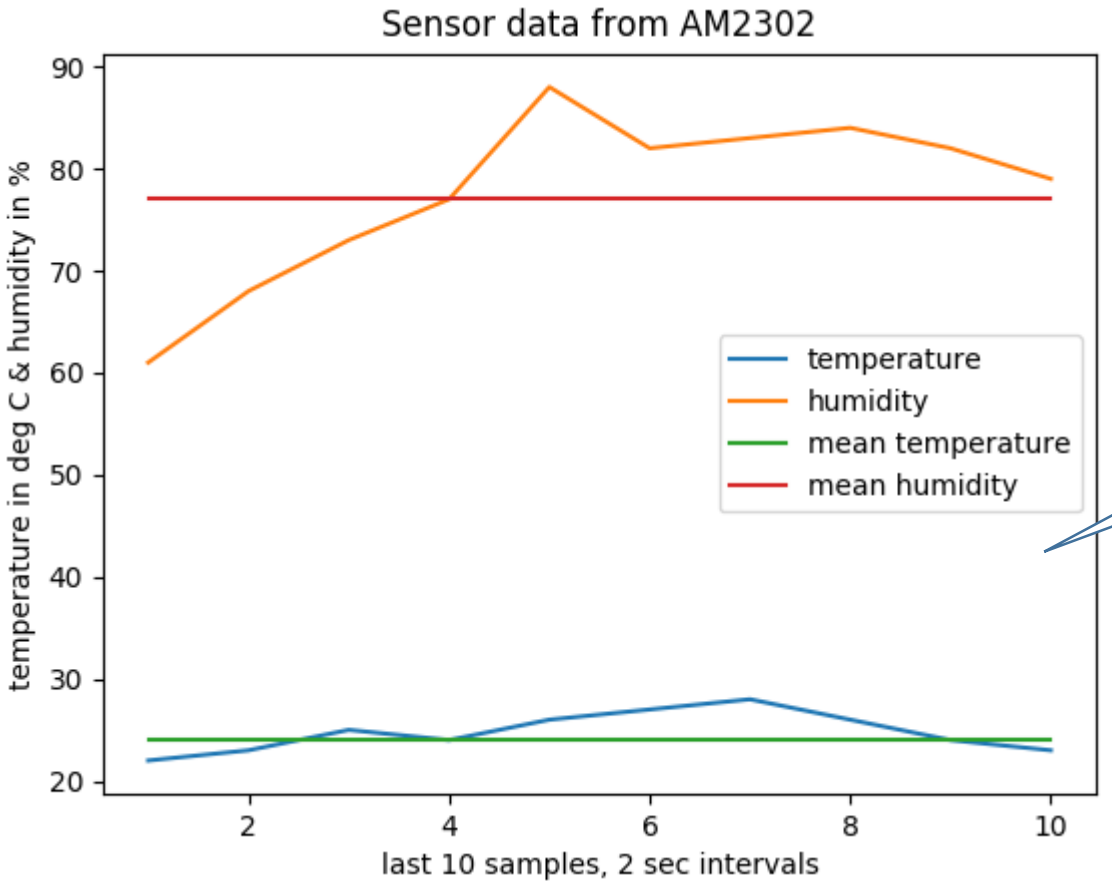
```
His English T-score is:  47
His Mother Tongue T-score is:  59
His Math T-score is:  63
His Science T-score is:  56
His PSLE score is:  225
```

Exercise 4.3 – Are they above average?

Write a Python program to plot this set of data, along with the average temperature & humidity values, in the same graph.

Data.

```
1,22,61
2,23,68
3,25,73
4,24,77
5,26,80
6,27,82
7,28,83
8,26,84
9,24,82
10,23,79
```



Graph.

E1_Ex4_3.py - /home/pi/PythonElectives/E1_Ex4_3.py (3.5.3)

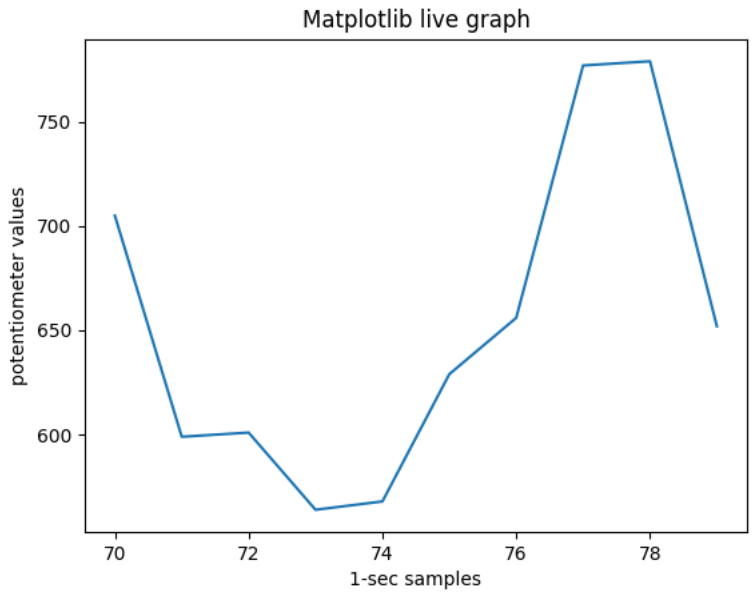
File Edit Format Run Options Window Help

Program.

```
import matplotlib.pyplot as plt
import csv
import numpy as np

n=[]
temp=[]
humi=[]
ave_temp=[]
ave_humi=[]

with open('sensordata.txt','r') as csvfile:
    data=csv.reader(csvfile,delimiter=',')
    for row in data:
        n.append(int(row[0]))
        temp.append(int(row[1]))
        humi.append(int(row[2]))

mean_temp=int(np.mean(temp))
mean_humi=int(np.mean(humi))
ave_temp=[mean_temp]*10
ave_humi=[mean_humi]*10

plt.plot(n,temp,label='temperature')
plt.plot(n,humi,label='humidity')
plt.plot(n,ave_temp,label='mean temperature')
plt.plot(n,ave_humi,label='mean humidity')
plt.xlabel('last 10 samples, 2 sec intervals')
plt.ylabel('temperature in deg C & humidity in %')
plt.title('Sensor data from AM2302')
plt.legend()
plt.show()
```

Exercise 4.4 – Live sensor graph

Try this Python program, which plots the last 10 potentiometer readings, at 1-sec intervals, in a line chart. Note that this is a "live" graph!



Graph.

E1_Ex4_4.py - /home/pi/Python files - elective/E1_Ex4_4.py (3.5.3)

File Edit Format Run Options Window Help

```
#import the modules needed
import RPi.GPIO as GPIO
from time import sleep
import spidev
import matplotlib.pyplot as plt
import matplotlib.animation as animation

#set up
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
spi=spidev.SpiDev()
spi.open(0,0)

#count up continuously
count=0

#lists to store readings (0-1023) & entry numbers (0, 1, 2...)
readings=[]
entries=[]

##one graph, and put this at row 1, col 1
fig=plt.figure()
pot_graph=fig.add_subplot(1,1,1)

def animate(i):
    #make variables global
    global count

    #read potentiometer value
    spi.max_speed_hz=1350000
    r=spi.xfer2([1,8+0<<4,0])
    pot_value=((r[1]&3)<<8)+r[2]
    #for each list: keep 10 items, add new item at back, increment count
    if count>9:
        readings.pop(0)
        entries.pop(0)
    readings.append(pot_value)
    entries.append(count)
    count=count+1

    #update plot
    pot_graph.clear()
    pot_graph.plot(entries,readings)
    pot_graph.set_xlabel('1-sec samples',fontsize=10)
    pot_graph.set_ylabel('potentiometer values',fontsize=10)
    pot_graph.set_title('Matplotlib live graph')

ani=animation.FuncAnimation(fig,animate,interval=1000)
plt.show()
```

Program.


THANK
YOU