SAVING SENSOR DATA TO A FILE

Edmond Lascaris and Anne Jessup - Creative Commons – 18 May 2021

OVERVIEW

In this lesson we are going to learn how to save our sensor data to a file. Saving data to a file will keep it safe. Once data is in a file, we can read the data again and make a graph. Reading, saving a graphing data is very important for all environmental and Citizen Science activities.

Sensors often deliver data a numerical data. Some simple examples are temperature, time, speed, counts, etc. In this example we are going to save our data as a String (or text). Most files store data as text. In this example we also need to separate the data using commas (,). If data is separated by commas it makes each data element human readable and other programs, such as Spreadsheets, can also read the file more easily.

LEARNING OBJECTIVES

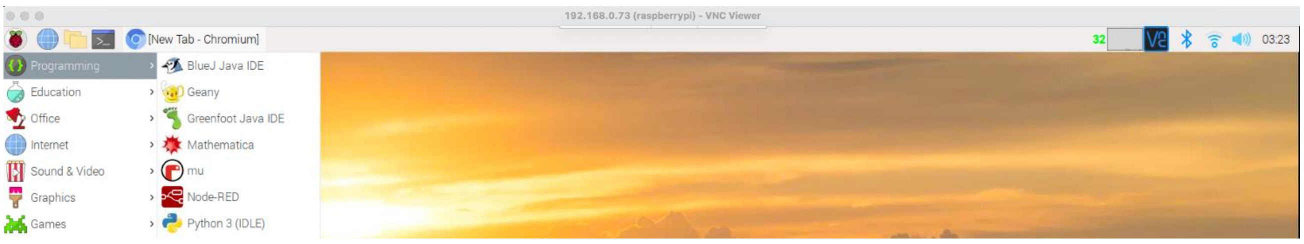
- Learn how convert numerical data into a String (or text)
- Learn how to save data to a new empty file
- Learn how to read a saved file and check for data integrity

CONVERTING NUMERICAL DATA TO A STRING

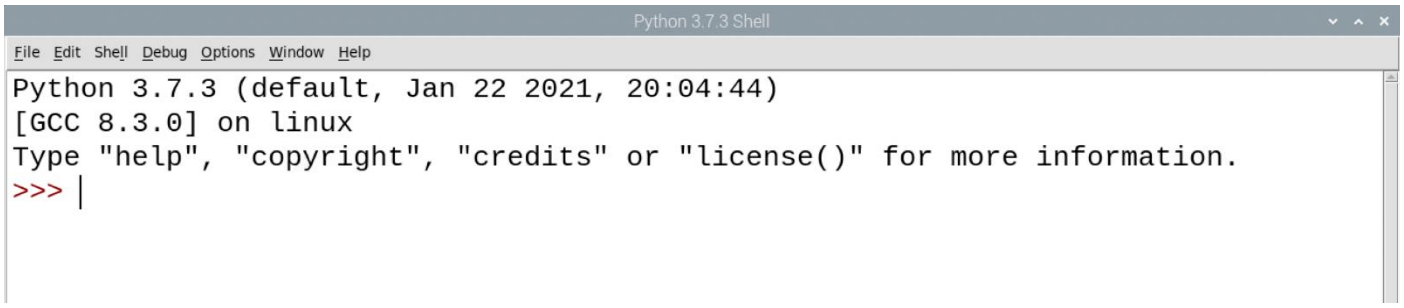
In this example we will build on the python program from the previous lesson. Our python program was used to download data from a sensor (temperature, atmospheric pressure, battery voltage). We will now convert this data into a String and put all the data into a single variable named data. Each piece of data will be separated by a comma.

1. **Open up the Python file named atm_sensor_get.py**

- From the Raspberry Pi main menu drop down select **Programming** > **Python3 (IDLE).**

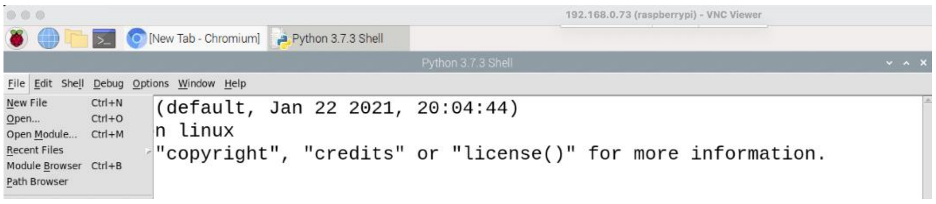


- This will open the Python Shell.



```
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

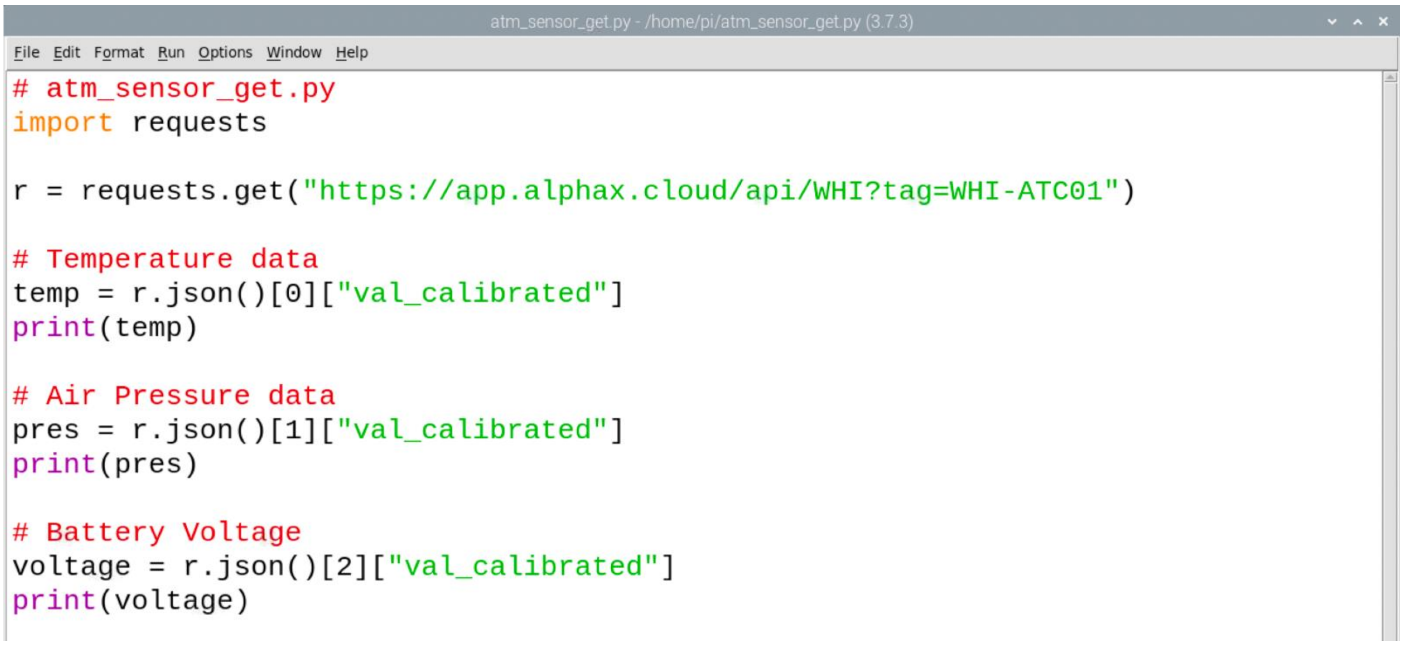
- From the **File** drop down menu select **Recent Files**.



```
(default, Jan 22 2021, 20:04:44)
n linux
"copyright", "credits" or "license()" for more information.
```

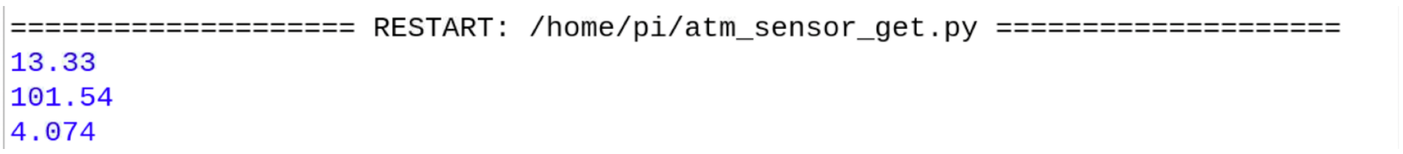
- Choose the file named **atm_sensor_get.py**

- Edit the file so that it appears like the example below.


atm_sensor_get.py - /home/pi/atm_sensor_get.py (3.7.3)
File Edit Format Run Options Window Help

```python
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)
```

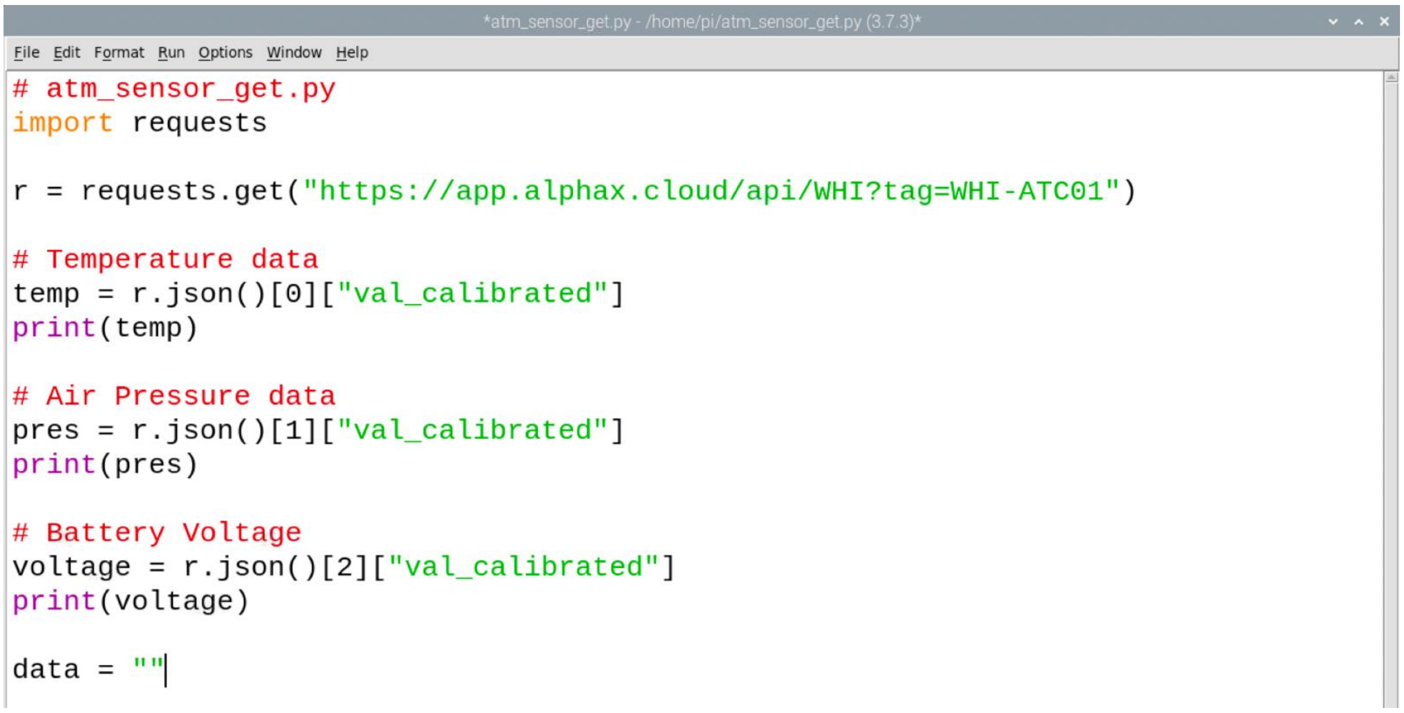
- **Saving** and **Running** the file should produce an output in the Python Shell like the output below.



```
==================== RESTART: /home/pi/atm_sensor_get.py ====================
13.33
101.54
4.074
```

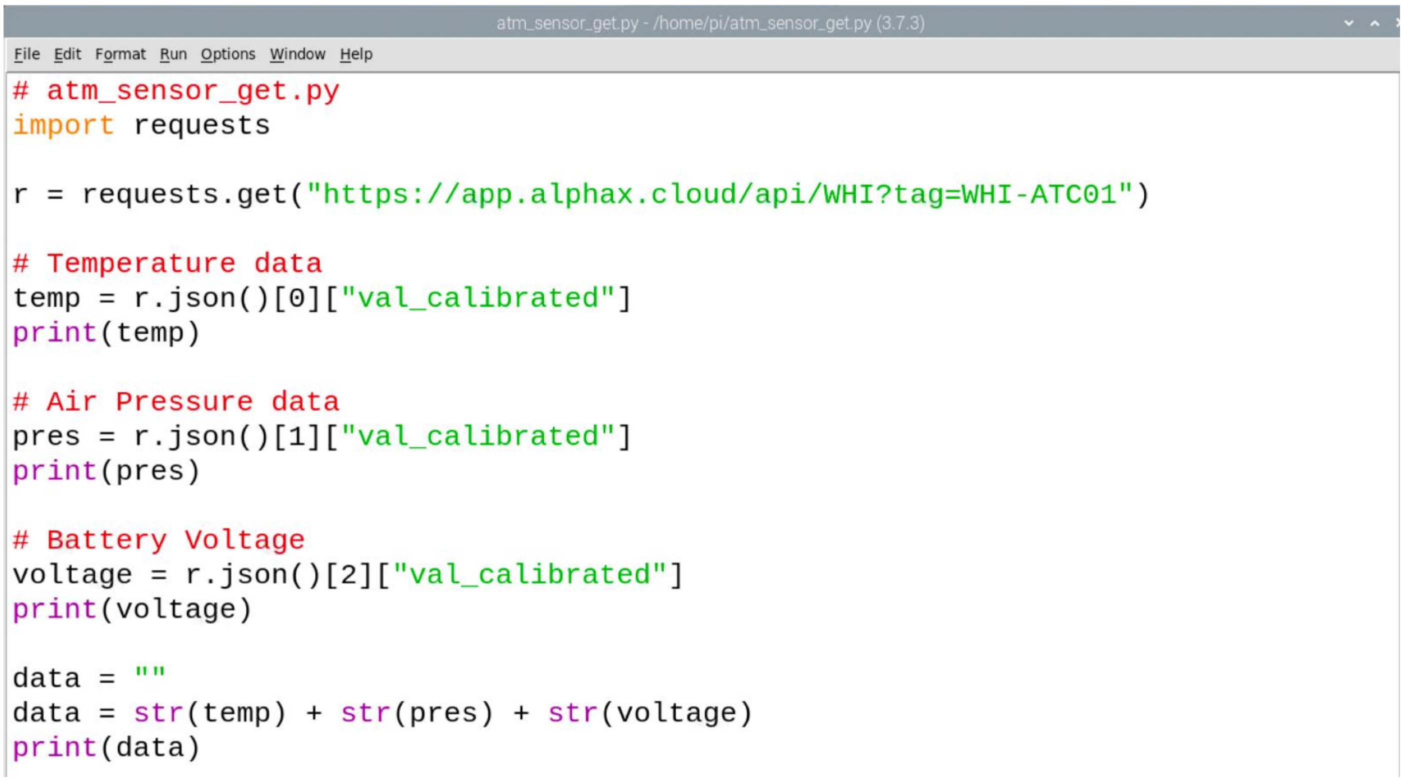
2. Arranging numerical data in a String

- We can create (or define) a new empty variable named data to hold all the temperature, pressure and battery data using the statement **data = “”**


atm_sensor_get.py - /home/pi/atm_sensor_get.py (3.7.3)
File Edit Format Run Options Window Help

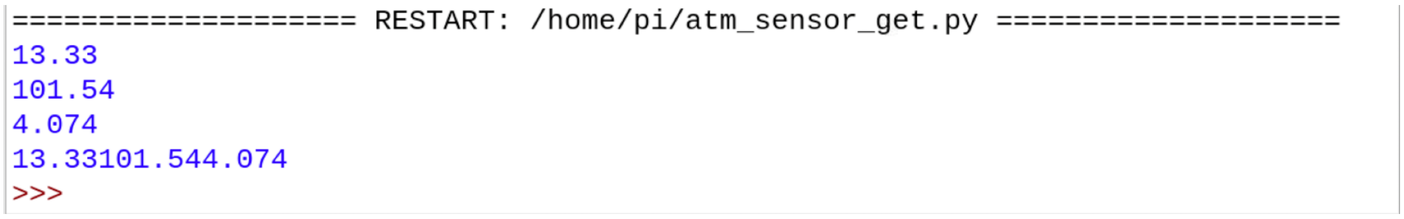
```python
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)

data = ""
```

- We need to add or concatenate all the data together into one long String.
- To do this we need to:
  - convert the numerical data to String using the **str()** function
  - then add each data String together using the plus (**+**) symbol
- The final code is presented below.
- We also added a **print(data)** statement so that we can see data variable.


atm_sensor_get.py - /home/pi/atm_sensor_get.py (3.7.3)
File Edit Format Run Options Window Help

```
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)

data = ""
data = str(temp) + str(pres) + str(voltage)
print(data)
```

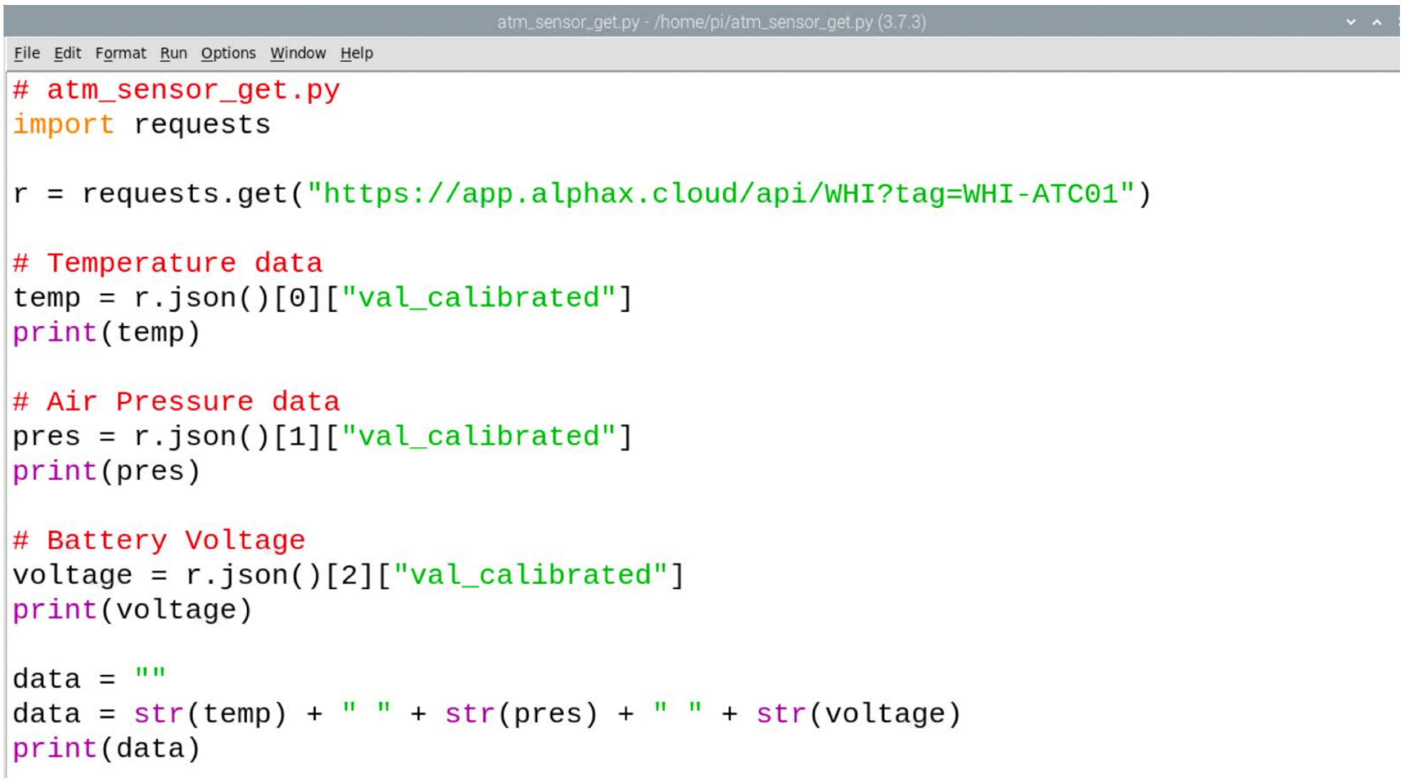
- **Save** and **Run** the program. The result should be like the following.



```
==================== RESTART: /home/pi/atm_sensor_get.py ====================
13.33
101.54
4.074
13.33101.544.074
>>>
```

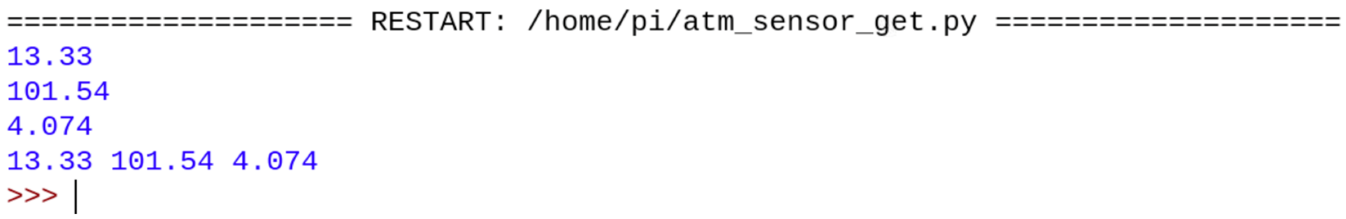
3. Adding spaces between data values

- The issue with the format of this data is that there is no gap between the data.
- To put a gap between the data we need to add a space using this additional code **+ " " +**
- See the example below as a reference.


atm_sensor_get.py - /home/pi/atm_sensor_get.py (3.7.3)
File Edit Format Run Options Window Help

```python
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)

data = ""
data = str(temp) + " " + str(pres) + " " + str(voltage)
print(data)
```

- The output of this program in the Shell is reproduced here.

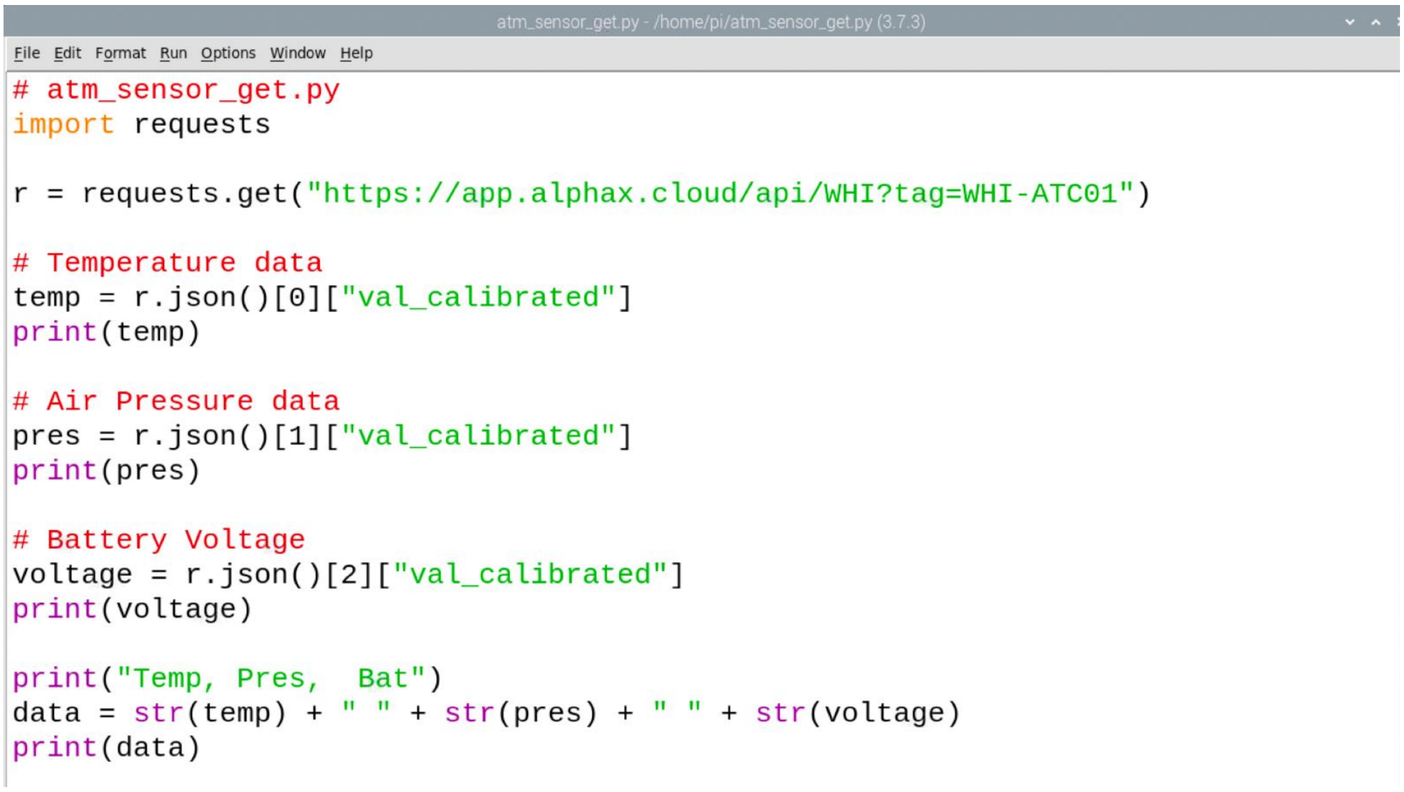


```
==================== RESTART: /home/pi/atm_sensor_get.py ====================
13.33
101.54
4.074
13.33 101.54 4.074
>>> |
```

4. Adding a human readable header to our data

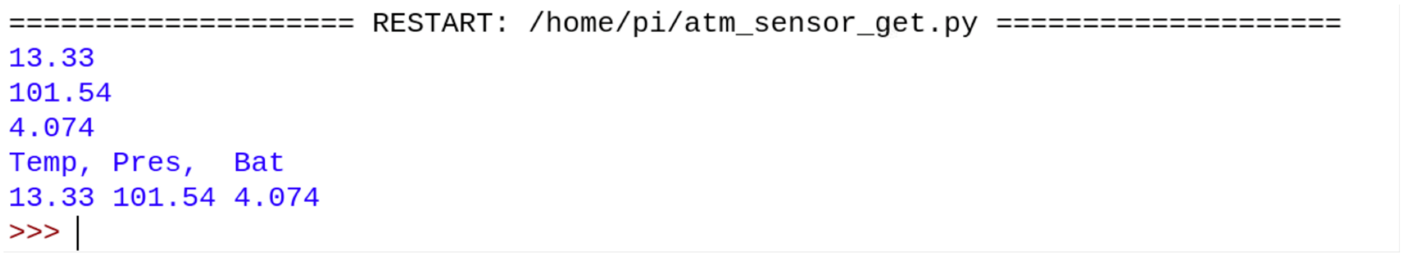
- To make the data easier to read we can add an additional **print()** statement to act as a **header** for our data.

- **print("Temp, Pres, Bat")**


atm_sensor_get.py - /home/pi/atm_sensor_get.py (3.7.3)
File Edit Format Run Options Window Help

```python
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)

print("Temp, Pres,  Bat")
data = str(temp) + " " + str(pres) + " " + str(voltage)
print(data)
```

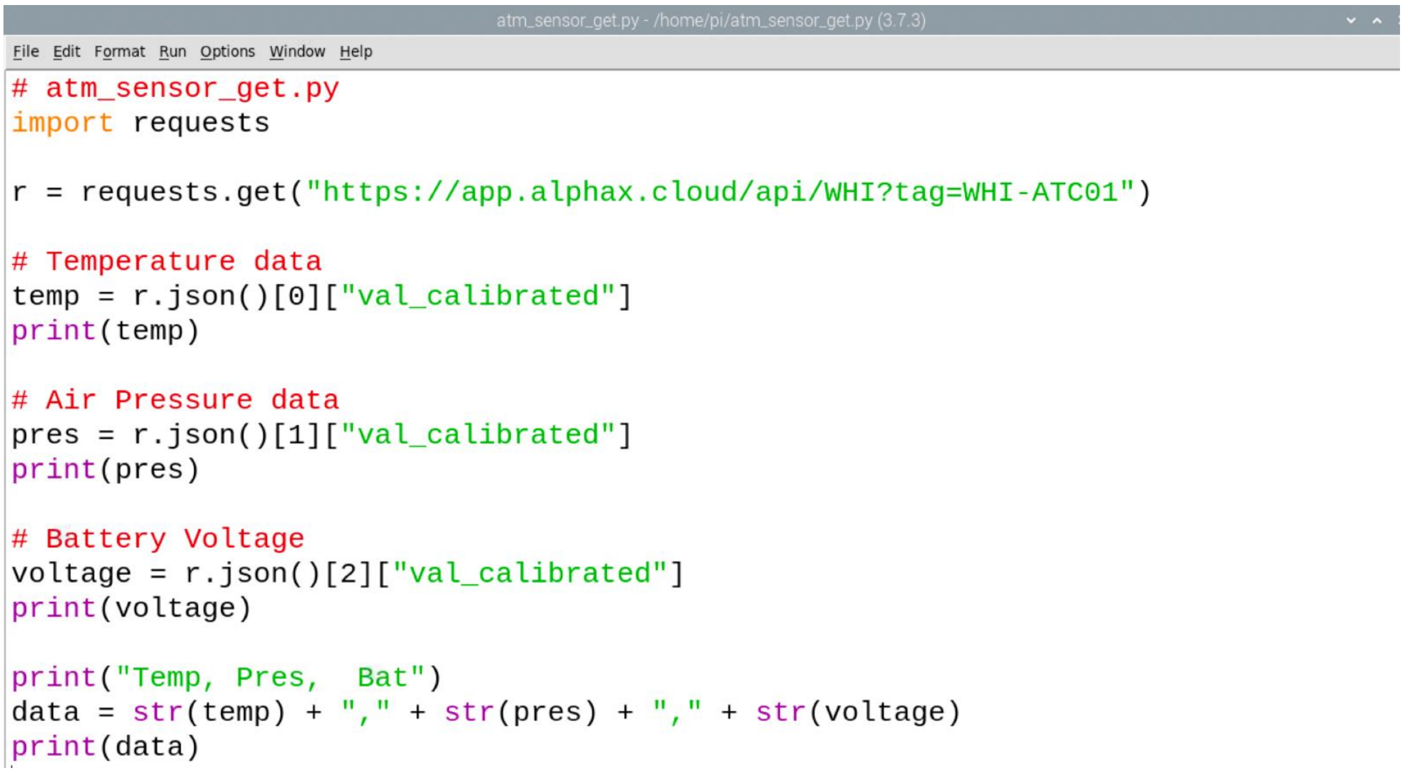
- With the additional print() header, it makes the data easier to read.
- We also remove the **data = “”** statement, because we don’t really need it in Python.



```
==================== RESTART: /home/pi/atm_sensor_get.py ====================
13.33
101.54
4.074
Temp, Pres,  Bat
13.33 101.54 4.074
>>> 
```

5. Comma separated data

- Rather than having a space separating the data it is better to use a **comma (,).**
- Replace all the **“ “** code with **“,”** as in the example below.

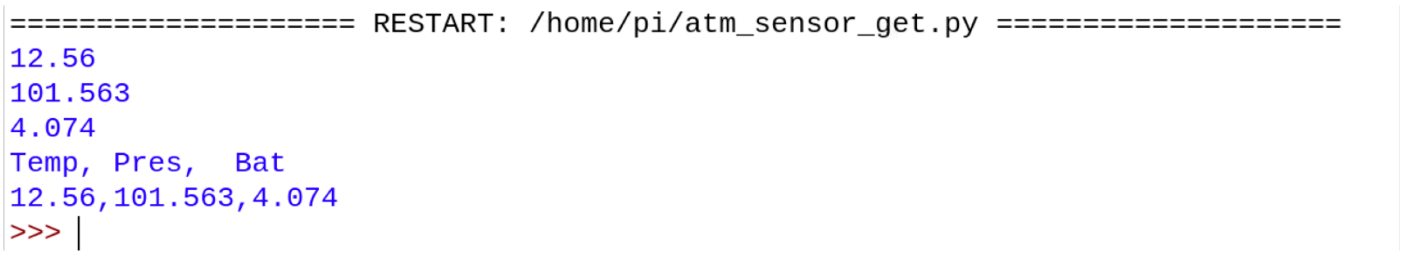


atm_sensor_get.py - /home/pi/atm_sensor_get.py (3.7.3)

File Edit Format Run Options Window Help

```python
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)

print("Temp, Pres,  Bat")
data = str(temp) + "," + str(pres) + "," + str(voltage)
print(data)
```

- After saving and running the program you should get an output like the one below.
- This is an important first step in preparing data to be saved to a file.



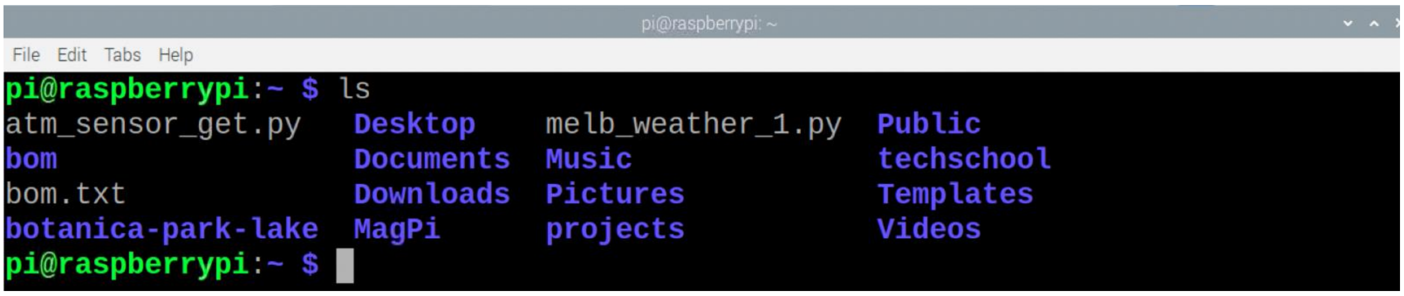
```
==================== RESTART: /home/pi/atm_sensor_get.py ====================
12.56
101.563
4.074
Temp, Pres,  Bat
12.56,101.563,4.074
>>> 
```

SAVING DATA TO A FILE

Data can be saved in a text file so that we don't lose the data, and so that other programs on our computer can access the data. With bigger data sets you can also save data in a specialised database or data warehouse. Saving data to a file is much easier and it also allows us to more easily check the data for integrity. In this example we will be creating an empty file named data.txt and then saving new data to this file.

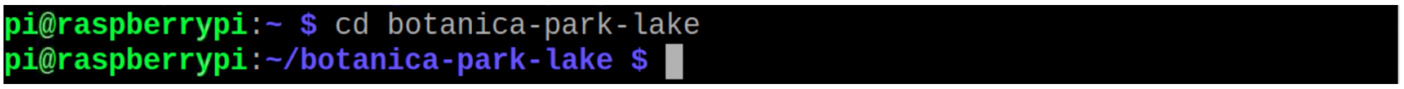
6. Navigating and listing files using the Terminal

- Open the **Terminal**.
- When you open the Terminal you will be in your **pi home** directory. You should see the **tilde (~)** symbol.
- The path to the pi home directory is **/home/pi** yet for simplicity this is abbreviated to **~**.
- Enter the command **ls** to list all the contents of the **pi home** directory.
- In my pi home directory, my project folder is named **botanica-park-lake**



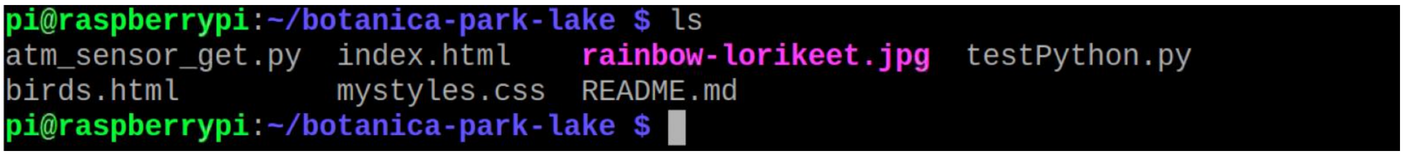
```
pi@raspberrypi:~ $ ls
atm_sensor_get.py   Desktop    melb_weather_1.py  Public
bom                 Documents  Music              techschool
bom.txt             Downloads  Pictures           Templates
botanica-park-lake  MagPi      projects           Videos
pi@raspberrypi:~ $
```

- To move into the botanica-park-lake directory we enter the command **cd botanica-park-lake**
- **cd** is short for **Change Directory**



```
pi@raspberrypi:~ $ cd botanica-park-lake
pi@raspberrypi:~/botanica-park-lake $
```

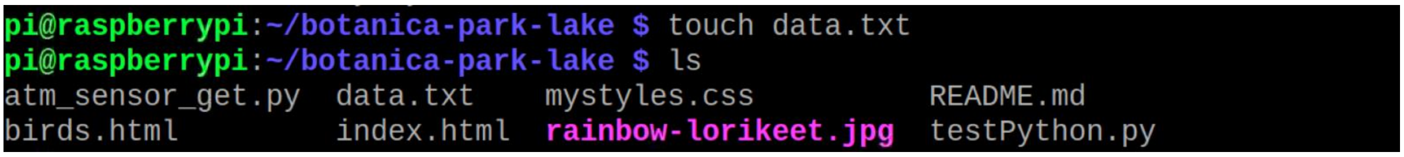
- One inside the botanica-park-lake directory we can also use the **ls** command to **list** all contents.
- Enter the command **ls**
- You should see a range of files similar to the ones displayed below.



```
pi@raspberrypi:~/botanica-park-lake $ ls
atm_sensor_get.py  index.html    rainbow-lorikeet.jpg  testPython.py
birds.html         mystyles.css  README.md
pi@raspberrypi:~/botanica-park-lake $
```

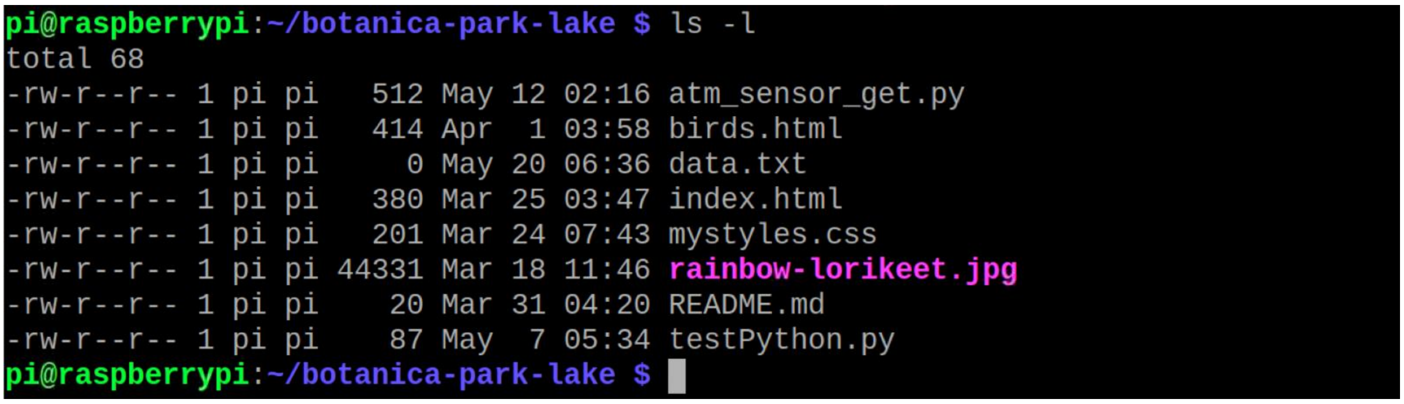
7. Creating an empty file in the Terminal

- Before we can save data to a file, we need to create an **empty file**.
- Enter the command **touch data.txt** to create an empty file named **data.txt**
- To verify that the file has been created enter the command **ls** to list all directory contents.
- You should see the new file **data.txt** in the listing.



```
pi@raspberrypi:~/botanica-park-lake $ touch data.txt
pi@raspberrypi:~/botanica-park-lake $ ls
atm_sensor_get.py  data.txt    mystyles.css         README.md
birds.html         index.html  rainbow-lorikeet.jpg  testPython.py
```

- Another way to list files is to add the **option -l**. This option lists the files in **long format**.
- Enter the command **ls -l**
- The file size value in bytes will be displayed in the column just **before the month** (e.g. Apr, May).
- We can see that the file size of data.txt is **0 bytes**.
- For comparison, the file size of the birds.html file is **414 bytes**.

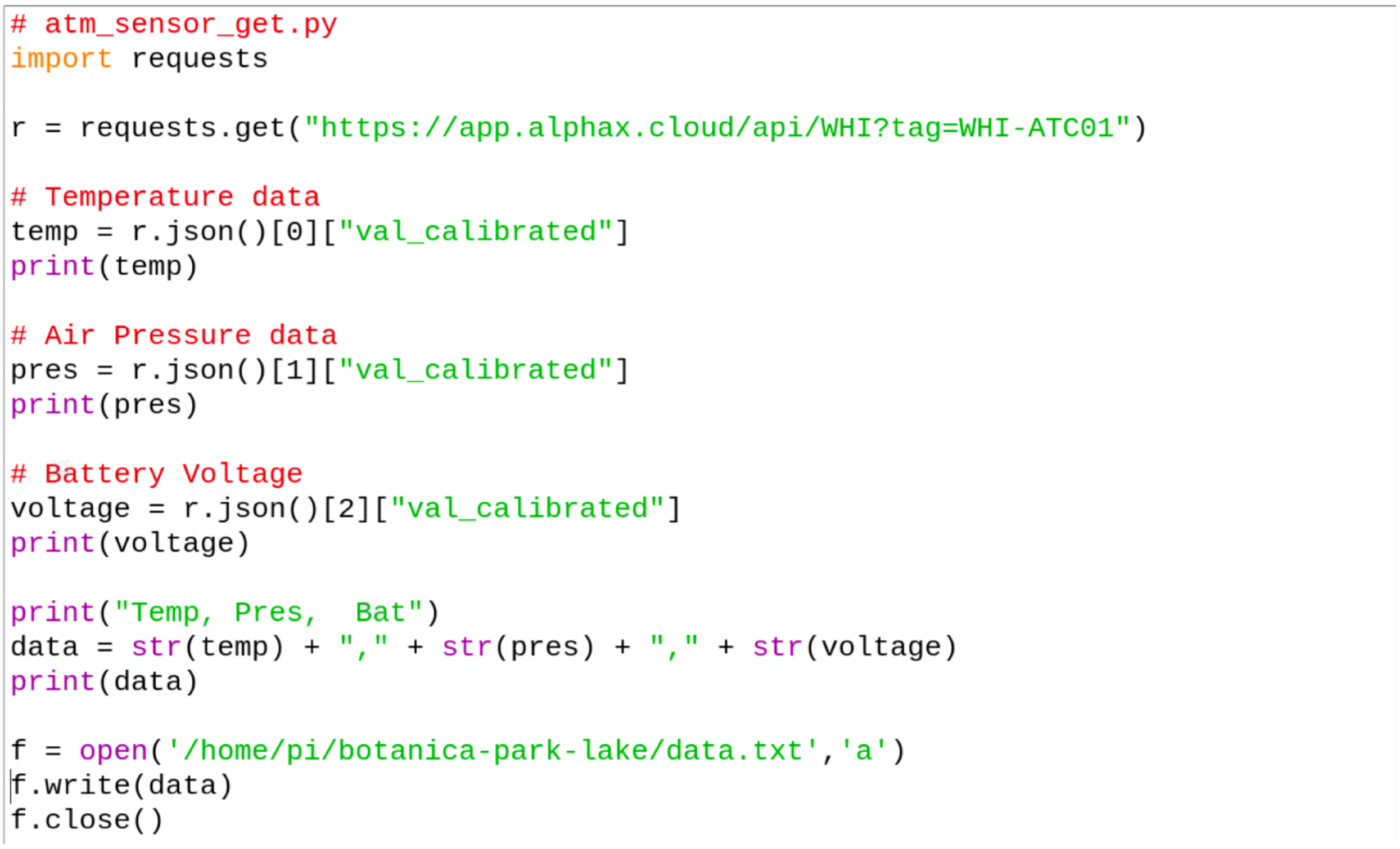


```
pi@raspberrypi:~/botanica-park-lake $ ls -l
total 68
-rw-r--r-- 1 pi pi   512 May 12 02:16 atm_sensor_get.py
-rw-r--r-- 1 pi pi   414 Apr  1 03:58 birds.html
-rw-r--r-- 1 pi pi     0 May 20 06:36 data.txt
-rw-r--r-- 1 pi pi   380 Mar 25 03:47 index.html
-rw-r--r-- 1 pi pi   201 Mar 24 07:43 mystyles.css
-rw-r--r-- 1 pi pi 44331 Mar 18 11:46 rainbow-lorikeet.jpg
-rw-r--r-- 1 pi pi    20 Mar 31 04:20 README.md
-rw-r--r-- 1 pi pi    87 May  7 05:34 testPython.py
pi@raspberrypi:~/botanica-park-lake $ 
```

8. Writing data to a file

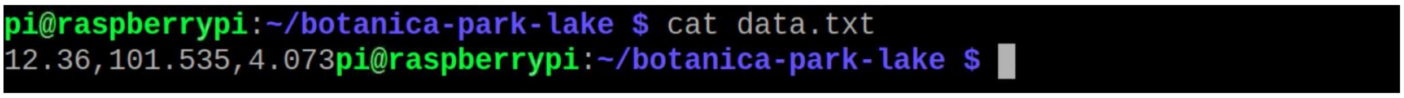
- To write the data to a file we need to add three lines of code at the end of the python file.
  - **f = open('/home/pi/botanica-park-lake/data.txt','a')**
  - **f.write(data)**
  - **f.close()**

- In the line **f = open()**
  - we identify the path to the file, in this case **/home/pi/botanica-park-lake/data.txt**
  - we also say that we want to **append (a)** new data to the file. This continually adds new data to the end of the file.
- **f.write(data)** – this statements writes the data to the file.
- **f.close()** – this statement closes the file. By closing the file other programs can then access or read the file.



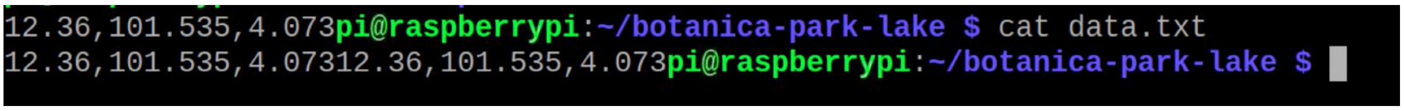
```
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)

print("Temp, Pres,  Bat")
data = str(temp) + "," + str(pres) + "," + str(voltage)
print(data)

f = open('/home/pi/botanica-park-lake/data.txt','a')
f.write(data)
f.close()
```

- If we run this file we won't see any changes to the output in the Python Shell.
- To see the new data we need to view the contents of the data.txt file. Hopefully it should hold some data now.
- To view the contents of a text file in Linux we enter the command **cat data.txt**
- **cat** is short for **concatenate**.
- We can see that the output shows that we have some numerical data stored in the file.



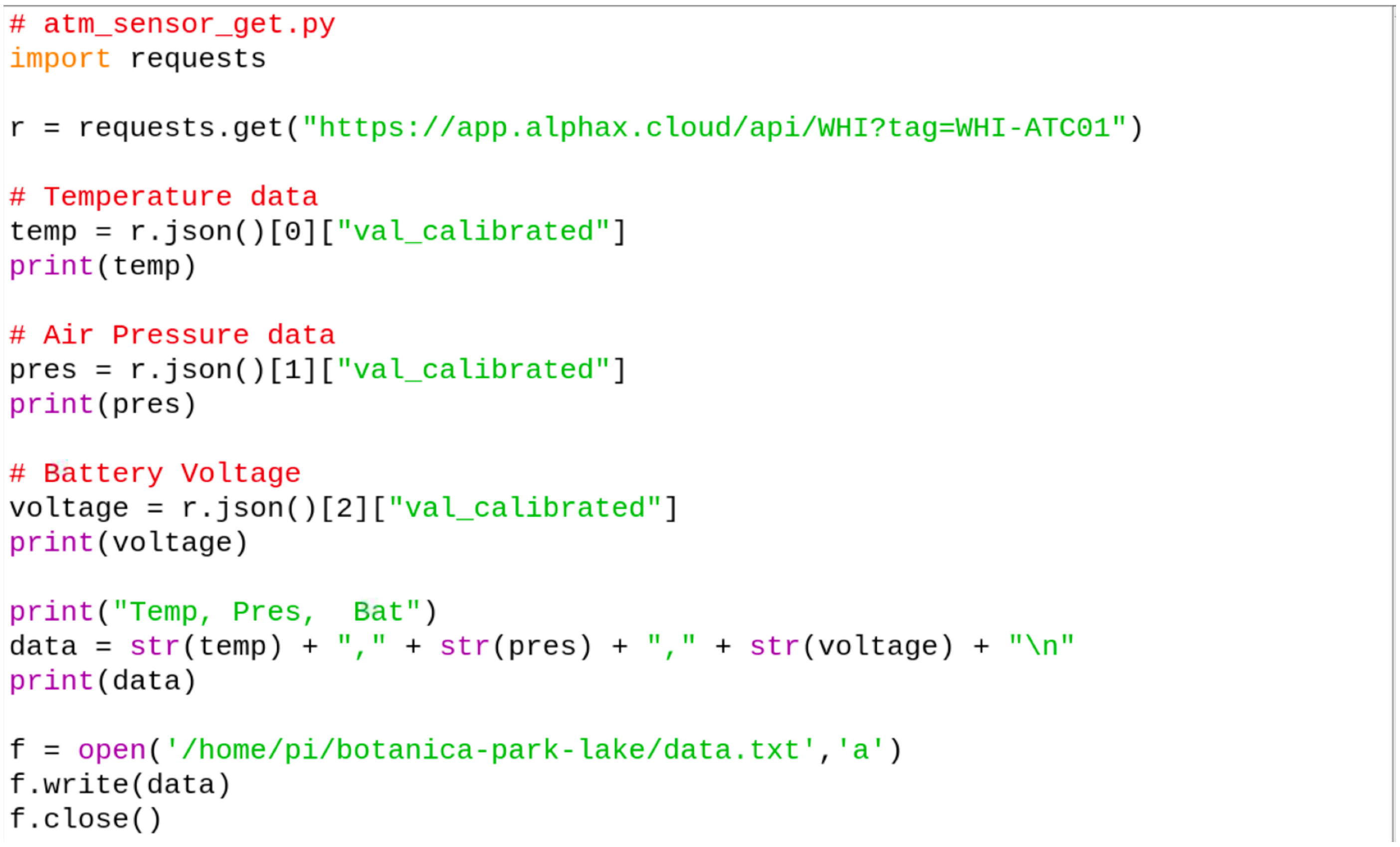
```
pi@raspberrypi:~/botanica-park-lake $ cat data.txt
12.36,101.535,4.073pi@raspberrypi:~/botanica-park-lake $ 
```

- If we go back to the Python Editor window (our program) and run the program again we start to notice a problem.
- You can see from the output below in the Terminal that the data starts to get jumbled together.
- **Run** the program again in IDLE3 (Python3) and then enter **cat data.txt** in the Terminal to see the results.
- Ideally, we would like to see each new data set on its own separate line.



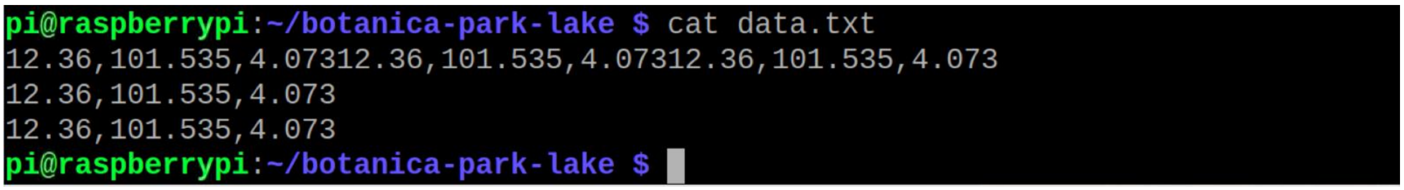
```
12.36,101.535,4.073pi@raspberrypi:~/botanica-park-lake $ cat data.txt
12.36,101.535,4.07312.36,101.535,4.073pi@raspberrypi:~/botanica-park-lake $ 
```

- To fix this problem we need to add a special "**new line**" character at the end of our data.
- The **new line special character** is symbolised by **"\n"**
- Modify the line **data = str(temp) + "," + str(pres) + "," + str(voltage)** to the following:
  - **data = str(temp) + "," + str(pres) + "," + str(voltage) + "\n"**



```python
# atm_sensor_get.py
import requests

r = requests.get("https://app.alphax.cloud/api/WHI?tag=WHI-ATC01")

# Temperature data
temp = r.json()[0]["val_calibrated"]
print(temp)

# Air Pressure data
pres = r.json()[1]["val_calibrated"]
print(pres)

# Battery Voltage
voltage = r.json()[2]["val_calibrated"]
print(voltage)

print("Temp, Pres,  Bat")
data = str(temp) + "," + str(pres) + "," + str(voltage) + "\n"
print(data)

f = open('/home/pi/botanica-park-lake/data.txt','a')
f.write(data)
f.close()
```

- If we Run the program 3 or 4 times we will start to see the following output using the **cat data.txt** command.
- Now you can see that the new data we are adding now appears on a new line.

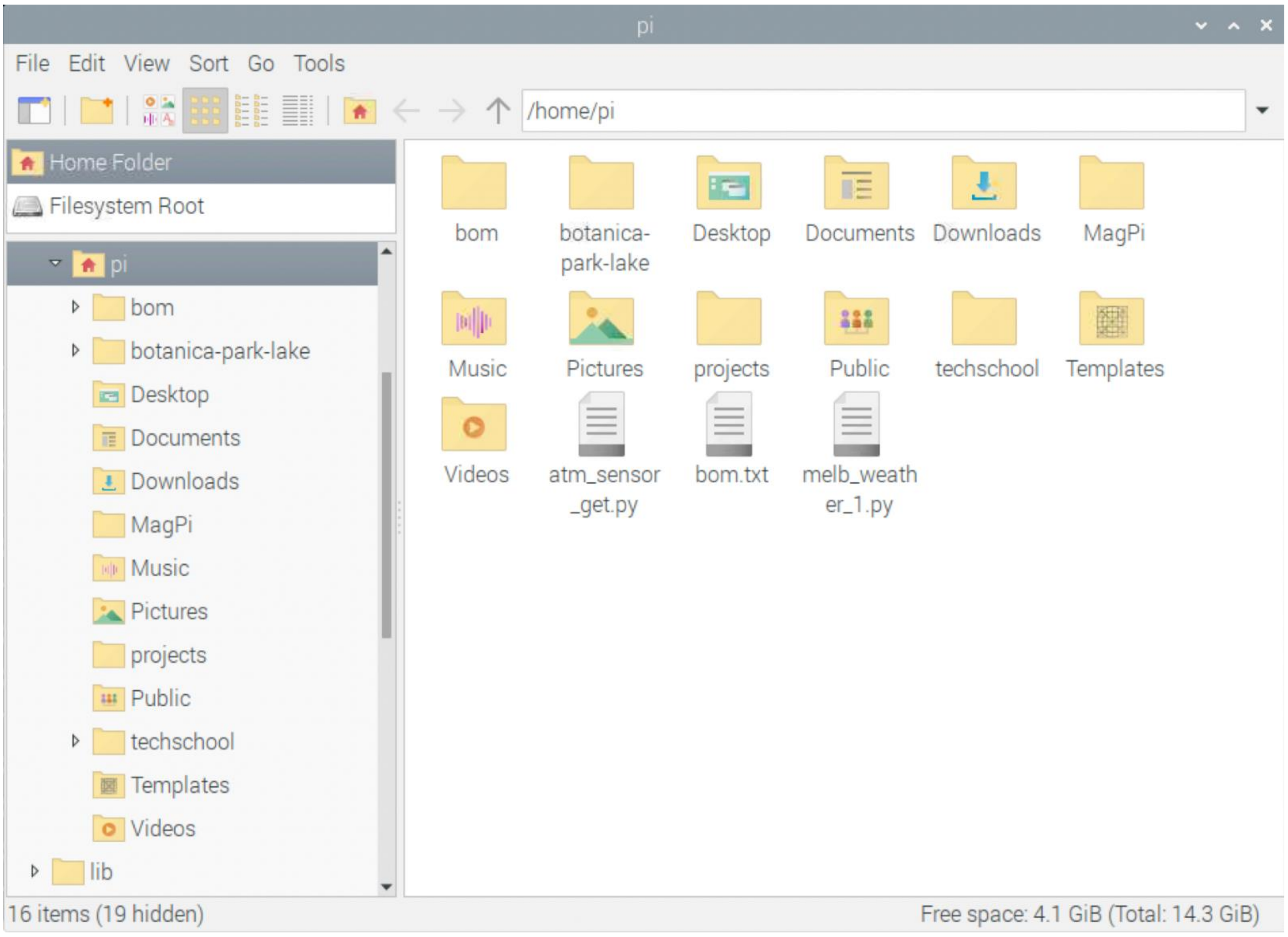


```
pi@raspberrypi:~/botanica-park-lake $ cat data.txt
12.36,101.535,4.07312.36,101.535,4.07312.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
pi@raspberrypi:~/botanica-park-lake $ 
```

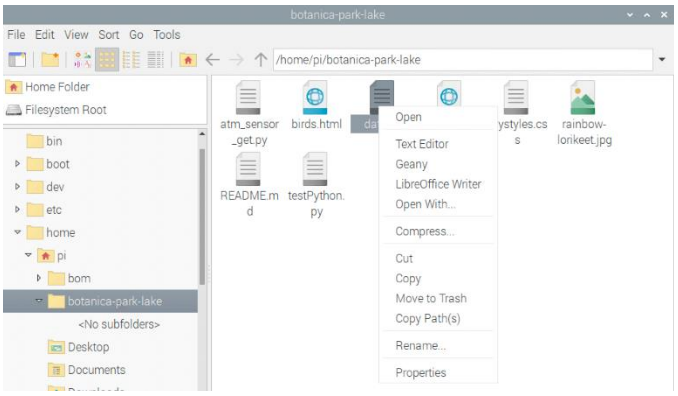
CHECKING A DATA FILE FOR INTEGRITY AND MAKING CORRECTIONS

We can now write data to a file. However, we may need to fix up the file because not all the data in the file has been saved correctly. In this example we will use a Text Editor to quickly fix up the formatting and data in a text file.

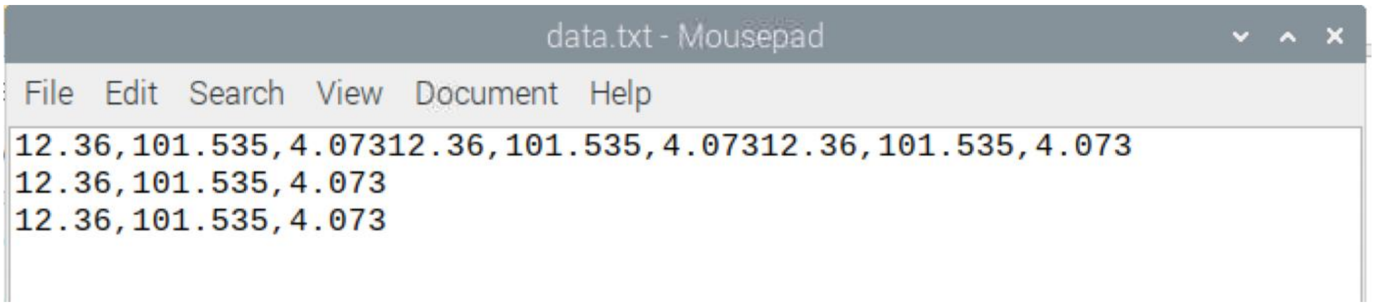
- Open the **File Manager** from the top menu bar on the Raspberry Pi



- Double click on your project folder to enter the folder. In my case I'll click on **botanica-park-lake**.
- Once in your project folder, find the file **data.txt** and **right-mouse button click** to bring up a **drop-down menu**.
- Select **Text Editor** from the **drop-down menu.**

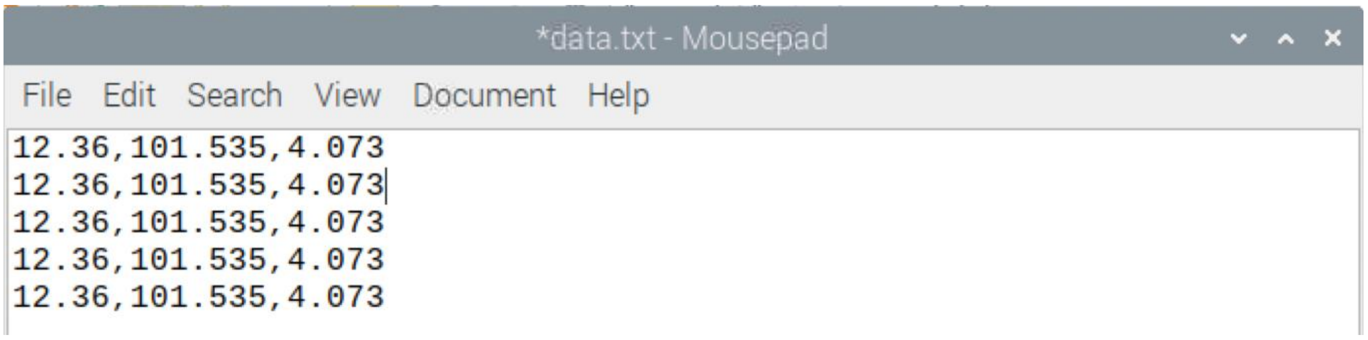


- The **Text Editor** that opens is called **Mousepad**.
- Mousepad behaves like a typical text editor.
- Try to arrange the data so that it is arranged correctly.



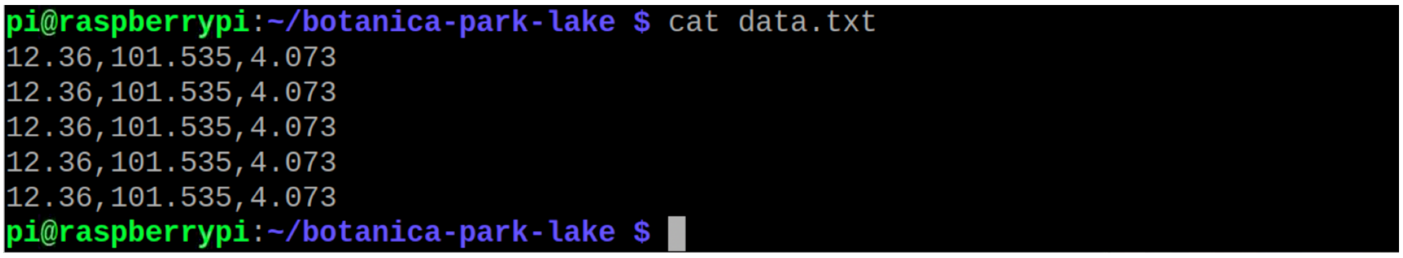
```
12.36,101.535,4.07312.36,101.535,4.07312.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
```

- The result may look like the following.
- Once finished, **Save** and **Close** the file.



```
12.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
```

- To verify that you have modified and saved the file correctly go to the **Terminal** and enter **cat data.txt**
- Your data should look like the following.



```
pi@raspberrypi:~/botanica-park-lake $ cat data.txt
12.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
12.36,101.535,4.073
pi@raspberrypi:~/botanica-park-lake $
```

- **Congratulations!** You can now save and make changes to your saved data to ensure data integrity.