

SAVING SENSOR DATA TO A FILE

Edmond Lascaris and Anne Jessup - Creative Commons – 25 May 2021

OVERVIEW

In this lesson we are going to learn how to add a date and time to our data which is sometimes called a **timestamp**. A timestamp records the current time that an event took place. This becomes important later when we want to graph the data that we collected.

Computer settings are important with dates and times. When we initially set up our Raspberry Pi, we need to define localisation options to be Melbourne, Australia. The computer can then calculate our local time and even make allowances for daylight savings and atomic clock corrections. It is also important that the Raspberry Pi is connected to the internet so that it can re-calibrate the onboard clock.

In python there is a special library called datetime that makes it easy to record and format dates and times. In this lesson we will create a new program to get some experience with the datetime library before we add it back into our main program.

LEARNING OBJECTIVES

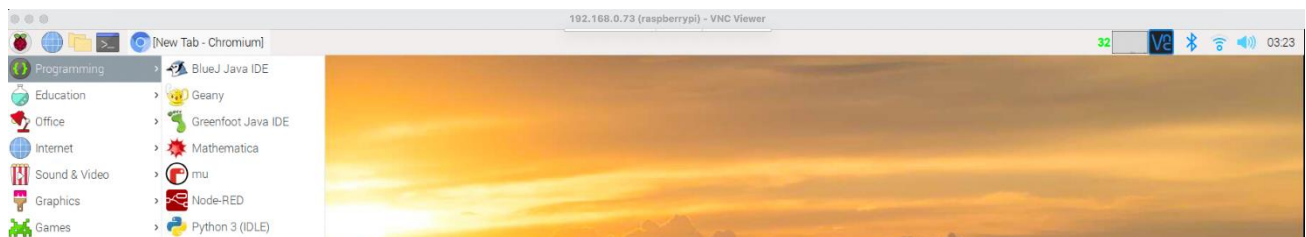
- Learn how use the python datetime library.
- Learn how to update our main program with the datetime library functions.
- Learn how to verify that our data has saved correctly using the File Manager and Terminal.

GETTING THE CURRENT DATE AND TIME

In this example we will experiment with a new python library named **datetime**. We will demonstrate how to get the current computer time, and also learn how to format the response. More information on formatting the datetime output can also be obtained from W3Schools at the following URL: [Python Dates \(w3schools.com\)](https://www.w3schools.com/python/python_datetime.asp)

1. Create a new file named timestamp.py

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



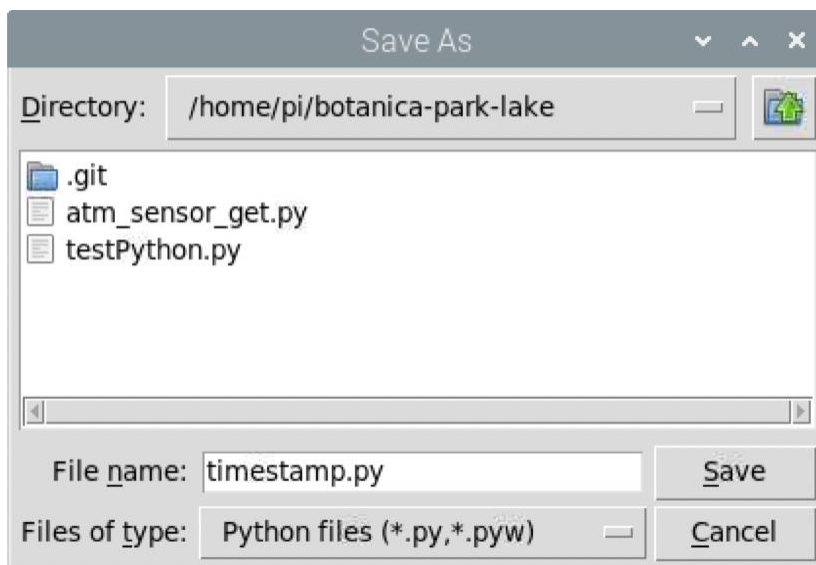
- This will open the Python Shell.

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
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 **New File**.
- Add a comment to the new file - **# timestamp.py**. The file name will be timestamp.py

```
*untitled*
File Edit Format Run Options Window Help
# timestamp.py|
```

- From the File dropdown menu select Save As.
- Navigate to your project directory. In this case it is **botanica-park-lake**
- Name the file **timestamp.py** and click **Save**.



- Enter the following code:
 - **import datetime** – library import

- **now** = `datetime.datetime(now)` – creates a new datetime object named **now**
- `print(now)` – printing the now object
- **Save and Run** the program

```
timestamp.py - /home/pi/botanica-park-lake/timestamp.py (3.7.3)
File Edit Format Run Options Window Help
# timestamp.py
import datetime

now = datetime.datetime.now()
print(now)
```

- The output in the Shell should show the year, month, day, hour, minute, second and microsecond time components.

```
===== RESTART: /home/pi/botanica-park-lake/timestamp.py =====
2021-05-26 03:42:46.161586
>>> |
```

2. Using the `strftime()` method

- When we create a **datetime object** (`now`) we have access to a method for formatting output of the date object into more readable strings.
- The method that does this is **`strftime()`**.
- To format the datetime object we need to use some special characters called **Directives**. Each Directive has a percent-sign (%) and a single upper or lower-case letter (**a-z, A-Z**).
 - **%Y** – full version of year – e.g. 2021
 - **%b** – month name short version – e.g. Jan, Feb, Mar
 - **%d** – day of the month 01 to 31 – e.g. 28
 - **%H** – hour 00 to 24 – e.g. 08
 - **%M** – minute 00 to 59 – e.g. 48
- To use these Directives to format the date object add the following two lines of code.
 - `date_stamp = now.strftime("%Y")`

- `print(date_stamp)`

- The full code is included below.

```
timestamp.py - /home/pi/botanica-park-lake/timestamp.py (3.7.3)
File Edit Format Run Options Window Help
# timestamp.py
import datetime

now = datetime.datetime.now()
print(now)

date_stamp = now.strftime("%Y")
print(date_stamp)
```

- Save and Run the program.
- There are two lines of output:
 - **2021-05026 03:45:43...** – print out of the **now instance** of the datetime object
 - **2021** – print out of the now datetime object formatted using the **strftime()** method to show the **Year (%Y)**.

```
===== RESTART: /home/pi/botanica-park-lake/timestamp.py =====
2021-05-26 03:45:43.348315
2021
>>>
```

- We can now keep on adding Directives to achieve our own customised format for the date and time.
- In this example we have added the **short version of the month** using the **(%b)** Directive – e.g. Jan.
- To add the full version of the month we would use the **(%B)** Directive – e.g. January.
- To represent months as numbers 01 to 12 use the **(%m)** Directive – e.g. 01 (for January).

```
timestamp.py - /home/pi/botanica-park-lake/timestamp.py (3.7.3)
File Edit Format Run Options Window Help
# timestamp.py
import datetime

now = datetime.datetime.now()
print(now)

date_stamp = now.strftime("%Y-%b")
print(date_stamp)
```

- Save and Run the program and test some options.

```
===== RESTART: /home/pi/botanica-park-lake/timestamp.py =====
2021-05-26 07:20:44.033935
2021-May
>>>
```

- Add the **day** of the month (%d).
- The reason we have used this date format separated by **hyphens** (-), is because this style can be universally read by other python programs. In the future we will write a program that will use the data to plot graphs.

```
timestamp.py - /home/pi/botanica-park-lake/timestamp.py (3.7.3)
File Edit Format Run Options Window Help
# timestamp.py
import datetime

now = datetime.datetime.now()
print(now)

date_stamp = now.strftime("%Y-%b-%d")
print(date_stamp)
```

- We are interested in capturing data once every hour, so it makes sense to also record the hour and minute that the data was collected.
- Update your code with the example below with the **Hour** (%H) and **Minute** (%M) Directives.
- We have used the standard human readable format for **Hour** and **Minute**, separated by a **colon** (:) – e.g. **16:24**

```
timestamp.py - /home/pi/botanica-park-lake/timestamp.py (3.7.3)
File Edit Format Run Options Window Help
# timestamp.py
import datetime

now = datetime.datetime.now()
print(now)

date_stamp = now.strftime("%Y-%b-%d %H:%M")
print(date_stamp)
```

- Using the **strftime()** method we can now obtain an output (**date_stamp** variable) that has the date and time formatted correctly for other programs to use.

```
===== RESTART: /home/pi/botanica-park-lake/timestamp.py =====
2021-05-26 07:22:17.363661
2021-May-26 07:22
>>> |
```

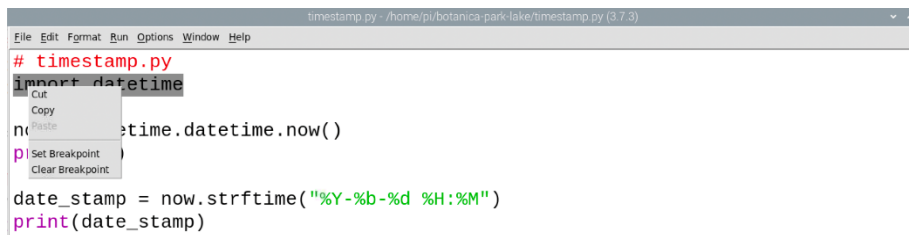
UPDATING OUR MAIN PROGRAM

When developing a large program, it is easier to break the key functions into separate modules so that they can be tested independently. Once they are ready, they can then be merged into the main program.

We have created and tested our **timestamp.py** program and shown that it works. In this example we will demonstrate how to copy sections of code into our original **atm_sensor_get.py** program to improve the functionality.

3. Updating atm_sensor_get.py

- Now that we have created our **timestamp.py** program and shown that it works, we can now copy sections of code into our original **atm_sensor_get.py** program
- In the **timestamp.py** program copy the import datetime line by highlighting the code, **right-mouse button** click and selecting **Copy**.



```
timestamp.py - /home/pi/botanica-park-lake/timestamp.py (3.7.3)
File Edit Format Run Options Window Help
# timestamp.py
import datetime
now = datetime.datetime.now()
date_stamp = now.strftime("%Y-%b-%d %H:%M")
print(date_stamp)
```

- In the **atm_sensor_get.py** program, paste the copied code by right-mouse button clicking and selecting **Paste**.
- Add the code just below the **import requests** statement. We try to keep **imports** together at the top of the code.
- You should see the **import datetime** added to your code.

```

*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
import datetime

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()

```

- Go to the timestamp.py program and **copy** the **two lines** of code highlighted below.
- You may need to **delete** some print statements that are not longer needed.

```

*timestamp.py - /home/pi/botanica-park-lake/timestamp.py (3.7.3)*
File Edit Format Run Options Window Help
# timestamp.py
import datetime

now = datetime.datetime.now()
data = now.strftime("%Y-%b-%d %H:%M")

print(data + "timestamp")

```

- **Paste** these two lines of code into the atm_sensor_get.py program, just below the import datetime statement.


```

*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
import datetime

now = datetime.datetime.now()
date_stamp = now.strftime("%Y-%b-%d %H:%M")

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()

```

- Within the `atm_sensor_get.py` program go down to almost the bottom of the code and **edit** the lines to reflect the example included below.
- The key changes are highlighted in yellow:
 - `print("Date_stamp,")`
 - `data = date_stamp + "," +`

```

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

```

- If you **Save** and **Run** the program you should see the following output.

```

===== RESTART: /home/pi/atm_sensor_get.py =====
9.21
101.597
4.055
Date_stamp,          Temp,Pres,   Bat
2021-May-27 06:27,9.21,101.597,4.055

>>> |

```

- In the activity that follows we will check to see that the data has been saved correctly.

VERIFYING THAT DATA HAS BEEN SAVED CORRECTLY

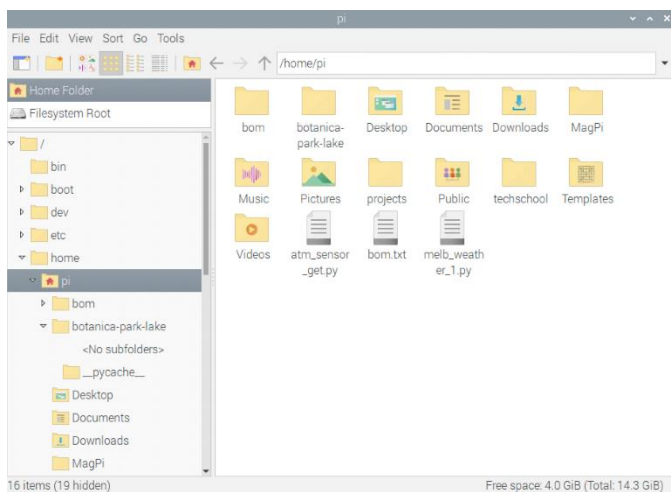
Our program saves data to the data.txt file in our project's directory. We need to check that the data has been saved correctly. In this example we will use two techniques to verify that the data has been saved. One method is to use the **File Manager** and the other method is to use the **Terminal**.

4. Verify data using the File Manager.

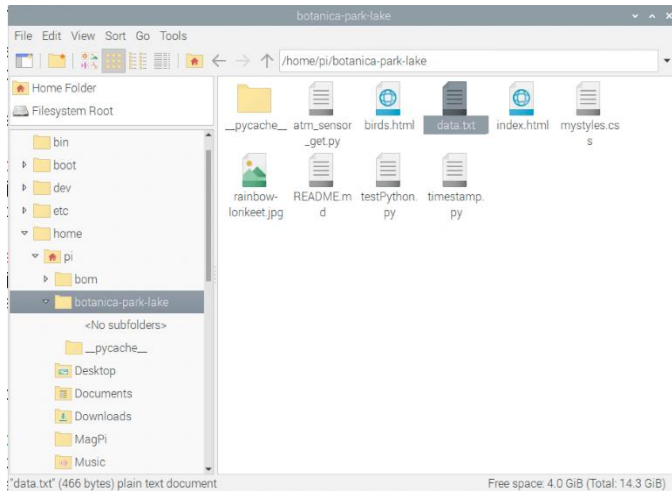
- Go to the top left of the Raspberry Pi Desktop and **click** on the **File Manager** icon.
- The **File Manager** icon looks like two yellow manilla folders.



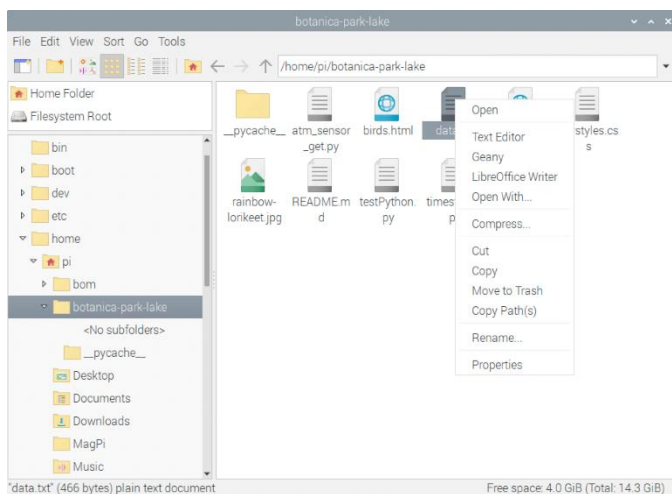
- The File Manager opens up to the pi user **home directory**.
- The path to this directory is **/home/pi**
- Double click on your **project directory**. In this case it is **botanica-park-lake**



- This should then open the botanica-park-lake **directory** where all your project files are located.



- **Right-mouse button click** on the data.txt file and select **Text Editor**.
- Alternatively, you can just double click on the data.txt file and it will open the default Text Editor.
- The default Text Editor on the Raspberry Pi is **Mousepad**.



- When the Text Editor opens you should be able to inspect your file.
- The example below show that the new data added (always **appended to the bottom** of the file) includes the **date** and **time**.
- If necessary, you can remove the other data entries that do not have a date and time.
- Once you have finished editing your file **Save** the changes and **Close** the file.

```
data.txt - Mousepad
File Edit Search View Document Help
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
2021-May-25 16:48,14.34,99.99,4.071
2021-May-25 16:57,14.34,99.99,4.071
2021-May-25 16:59,14.34,99.99,4.071
2021-May-25 17:12,14.34,99.99,4.071
2021-May-27 06:25,9.21,101.597,4.055
2021-May-27 06:26,9.21,101.597,4.055
2021-May-27 06:26,9.21,101.597,4.055
2021-May-27 06:27,9.21,101.597,4.055
2021-May-27 06:27,9.21,101.597,4.055
2021-May-27 06:27,9.21,101.597,4.055
```

5. Verify data using the Terminal

- Go to the top of the Raspberry Pi desktop and click on the Terminal icon.
- The Terminal icon looks is a **black box** with a **command prompt** (>_) inside.



- This will open the **Terminal** window.
- To verify where you are in the directory tree you can enter the command **pwd**
- **pwd** is short for **Present Working Directory**.
- In this case we are in the pi user home directory **/home/pi**

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ pwd
/home/pi
pi@raspberrypi:~ $
```



- If ever you get lost and want to get back to the /home/pi directory enter the command **cd ~**
- The ~ symbol is called **tilde**, pronounced [TIL] + [DAH].

```
pi@raspberrypi:~ $ cd ~
pi@raspberrypi:~ $
```

- To look at the files and directories in the /home/pi directory enter the command **ls**
- **ls** is short for **List**.

```
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 enter the botanica-park-lake directory enter the command **cd botanica-park-lake**

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

- To list all the files present, enter **ls**
- You should be able to see the **data.txt** file.

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

- Finally, to read the data.txt file enter the command **cat data.txt**
- This will output the complete listing of the file contents.

```
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
2021-May-25 16:48,14.34,99.99,4.071
2021-May-25 16:57,14.34,99.99,4.071
2021-May-25 16:59,14.34,99.99,4.071
2021-May-25 17:12,14.34,99.99,4.071
2021-May-27 06:25,9.21,101.597,4.055
2021-May-27 06:26,9.21,101.597,4.055
2021-May-27 06:26,9.21,101.597,4.055
2021-May-27 06:27,9.21,101.597,4.055
2021-May-27 06:27,9.21,101.597,4.055
2021-May-27 06:27,9.21,101.597,4.055
pi@raspberrypi:~/botanica-park-lake $
```

- **Congratulations!** In our next lesson we will learn how to automate our python program so that we can collect data every hour.