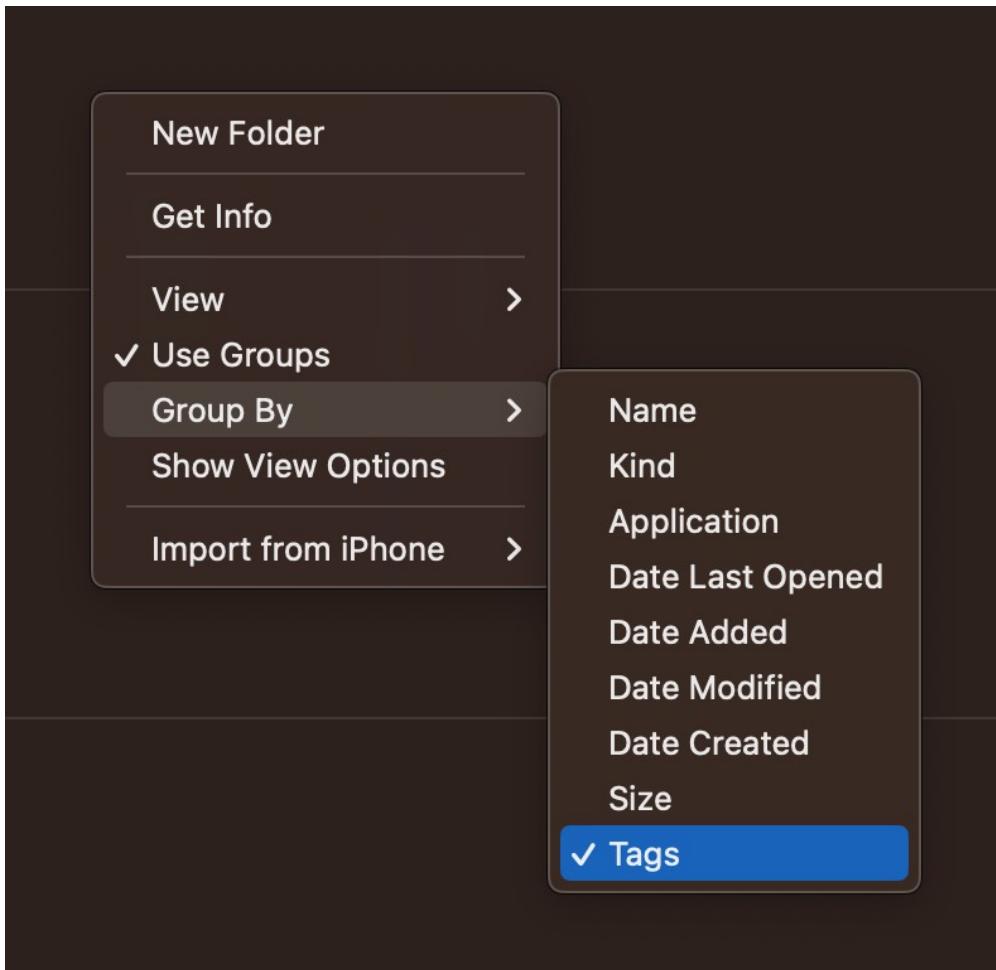


Documentation - Monitor Mac's Battery Health

1. Clone [monitor-battery-health](#) and navigate to the folder

- Place the cloned folder in the home directory (~/monitor-battery-health)
- Files are categorized based on tags
- select the groups option to group by tags



- **Blue** —> Automation app to fetch current cycle count (Cycle Count.app)
- **Green** —> shell script to automate the scripts (mycron.sh)
- **Red** —> main python scripts (BatteryLifeDays.py, cycleStatus.py, Analysis.py, monitorBattery.py)
- **Yellow** —> log files (cycleStatus.log, fiveMinjob.log, tenMinjob.log)

2. Automate monitorBattery python script using crontab

- Open terminal and type **crontab -e** to edit the cron file

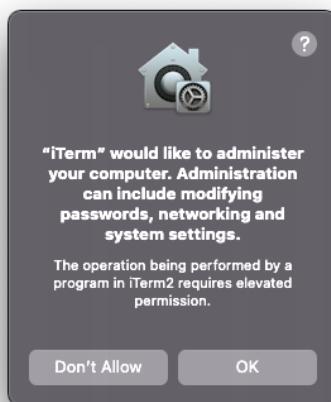
```
vasanthavan ~ (3:20:53)
$ crontab -e
```

- Press **i** on the keyboard to edit the file
- add any of the following line based on your preferences:

```
*/10 * * * * cd ~/monitor-battery-health && ./mycron.sh (if 10 minutes)
```

```
*/5 * * * * cd ~/monitor-battery-health && ./mycron.sh (if 5 minutes)
```

- This will run the monitorBattery python script every 5/10 minutes
- Once done, Press **escape** and type **:wq** and press enter
- To see the changes, type **crontab -l** to list the existing automation
- If your terminal asks permission to save the file, click OK



- Install all the requirements before running any scripts. Go to MonitorBattery Directory in Home folder and run:

```
vasanthavan ~/MonitorBattery (5:02:52)
$ pip3 install -r requirements.txt
```

3. Monitor log history

- Battery usage will be populated with a timestamp every 5 minutes in **fiveMinjob.log** (tenMinjob.log in case of 10 minutes)
- Open the file in the console application and view the ongoing execution response from the script.
- A timestamp will be initiated with 1970-01-01 at the beginning which represents the start of the execution.
- If charging, a lightning symbol will be shown next to the respective log

```
2022-09-15 22:25:01: 14%
2022-09-16 09:15:02: 16% (⚡)
2022-09-16 09:21:28: 22% (⚡)
2022-09-16 09:40:01: 40% (⚡)
2022-09-16 09:45:01: 45% (⚡)
2022-09-16 09:50:01: 50% (⚡)
2022-09-16 09:55:01: 55% (⚡)
2022-09-16 10:00:01: 59% (⚡)
2022-09-16 10:05:02: 64% (⚡)
2022-09-16 10:10:02: 69% (⚡)
2022-09-16 10:15:02: 73% (⚡)
2022-09-16 10:20:01: 73%
2022-09-16 10:25:01: 73%
```

- If the charge crosses 80%, a notification will be provided by the system and “**80% crossed**” text will be appended to the log file.



```
2022-09-15 11:23:44: 74% (⚡)
2022-09-15 11:28:44: 78% (⚡)
2022-09-15 11:30:01: 79% (⚡)
2022-09-15 11:37:10: - - - 80% Crossed! - - - -
2022-09-15 15:30:02: 79%
2022-09-15 15:35:01: 79%
2022-09-15 15:40:02: 78%
```

- Now, you can unplug the cable to avoid charging over 80%. Having this routine over the course of time will increase the gradual life expectancy of the battery health.
- Similarly, if the charge goes down to 1%, the script will notify the user with an alert notification.



4: Monitor Energy usage

- If there is a difference of **2%** between the two timestamps, then it concludes that there is significant energy being used by the laptop.
- The difference can be customized based on user's battery life condition.

2022-09-14 19:15:01:	52%
2022-09-14 19:20:02:	52%
2022-09-14 19:25:02:	51%
2022-09-14 19:30:02:	50%
2022-09-14 19:35:06:	48%
2022-09-14 19:40:02:	47%
2022-09-14 19:45:02:	47%
2022-09-14 19:50:01:	46%
2022-09-14 19:55:01:	45%
2022-09-14 20:01:50:	45%
2022-09-14 20:05:02:	45%

- Eventually, the python script will identify and notify the user with an alert.



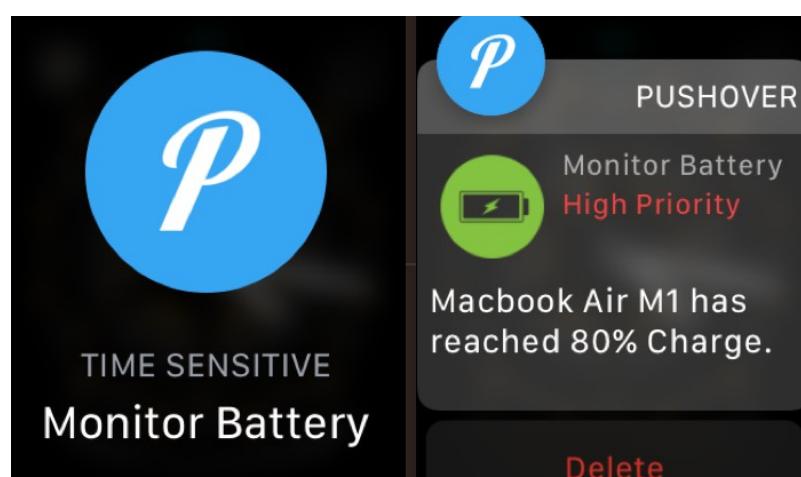
- You can configure the difference in the python script using the **intervalValue** variable in line 42: monitorBattery.py file

```
36  
37 # ----- Variable Assignment by user -----  
38  
39 isMuted = False  
40 apiToken = None  
41 userToken = None  
42 intervalValue = 3  
43  
44 # -----  
45
```

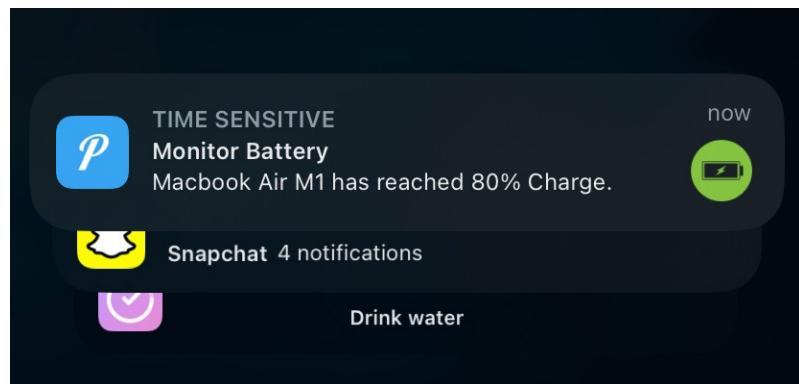
- If MacBook is idle and inactive for a few minutes, it will turn to sleep mode and the user may not be able to see the appropriate notification provided by the python script.
- In that case, users can make use of [PushOver](#): an application which provides push notifications using API calls.
- **Note:** This is optional. The script can run without this feature too.
- Once you log in to PushOver, you will receive an **API** token and a **User** token
- Provide the string value in line 40 and line 41 replacing **None**.

```
38  
39 isMuted = False  
40 apiToken = 'YOUR-API-KEY'  
41 userToken = 'YOUR-USER-KEY'  
42 intervalValue = 2  
43
```

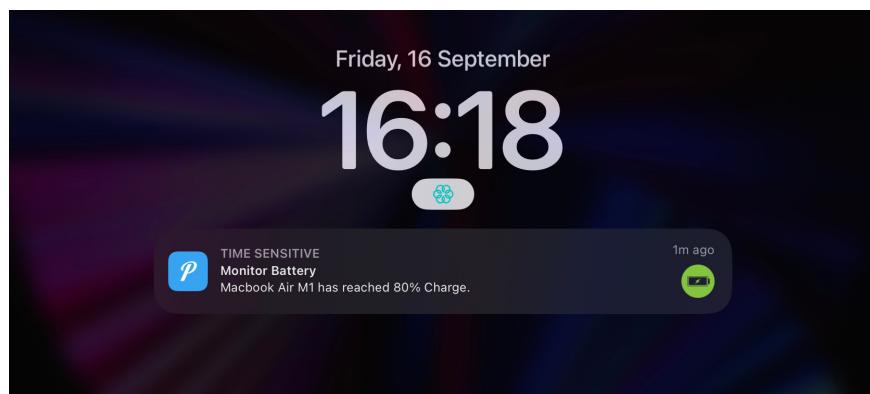
- This will provide notifications instantly to all signed-in devices like Apple iPad, iPhone, Watch and Mac respectively. (This differs based on your subscription towards Pushover)
- Apple Watch:



- Apple iPhone:

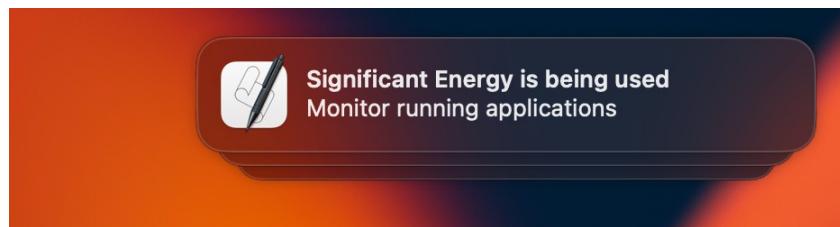


- Apple iPad:



5: Manage annoying notifications

- Sometimes if you render video or 3D image, extensive core efficiency and performance will be used which results in quick battery consumption.
- Notification will be thrown every 5 minutes which would be annoying to users as shown below.



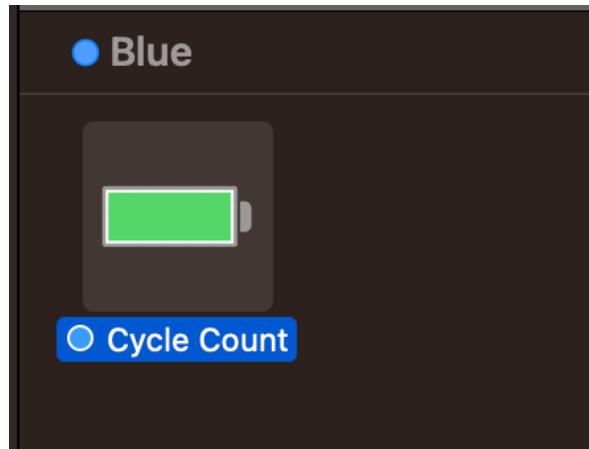


- If you want to disable notification for some time, enable the **isMuted** variable to **True** in line 39. Toggle back to False to bring back the notification feature.

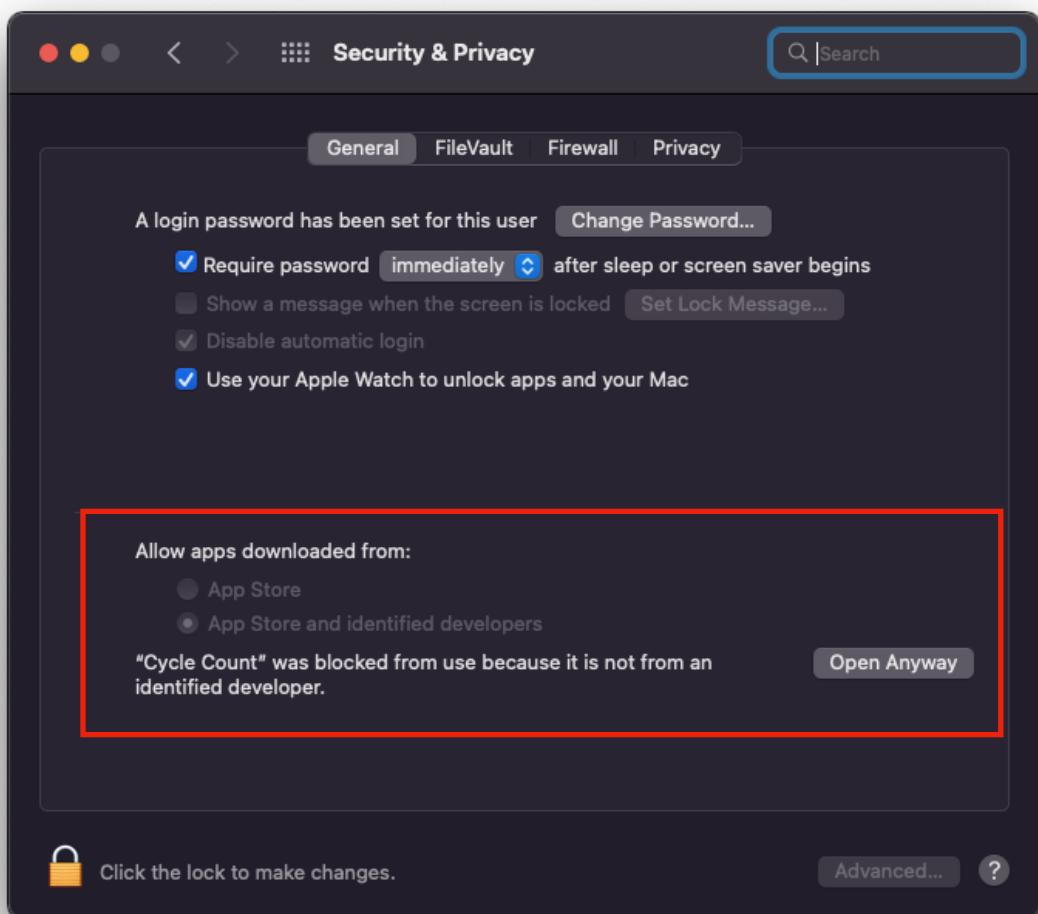
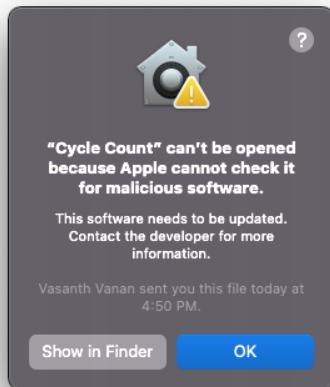
```
37  # ----- Variable Assignment by user -----
38
39  isMuted = True
40  apiToken = None
41  userToken = None
42  intervalValue = 2
43
44  # -----
```

6: Fetch Current Cycle Count

- Open Cycle Count — Automation App



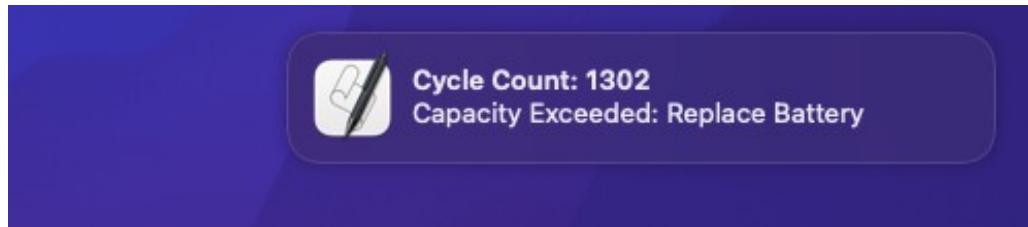
- Apps other than those downloaded from App Store will be generally blocked by MacBooks. But, It can be configured in settings. please select **Open Anyway** in Security & Privacy to run it.



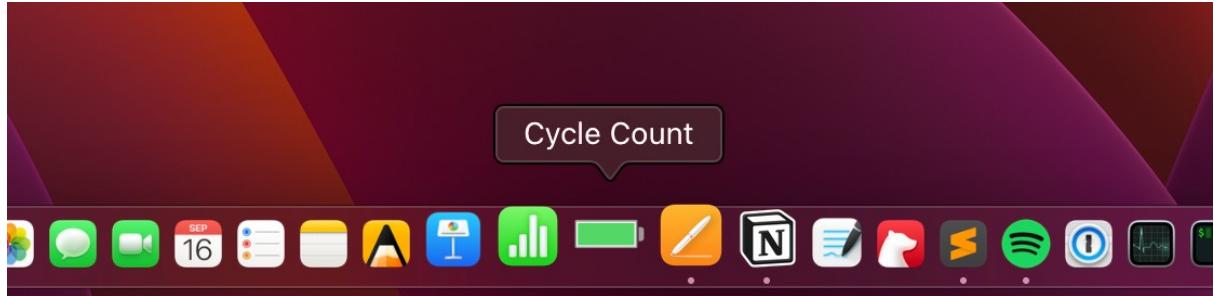
- Once fixed, A notification will be popped up with the cycle count information.



- If the Cycle count exceeds **1000**, a different Notification will be displayed

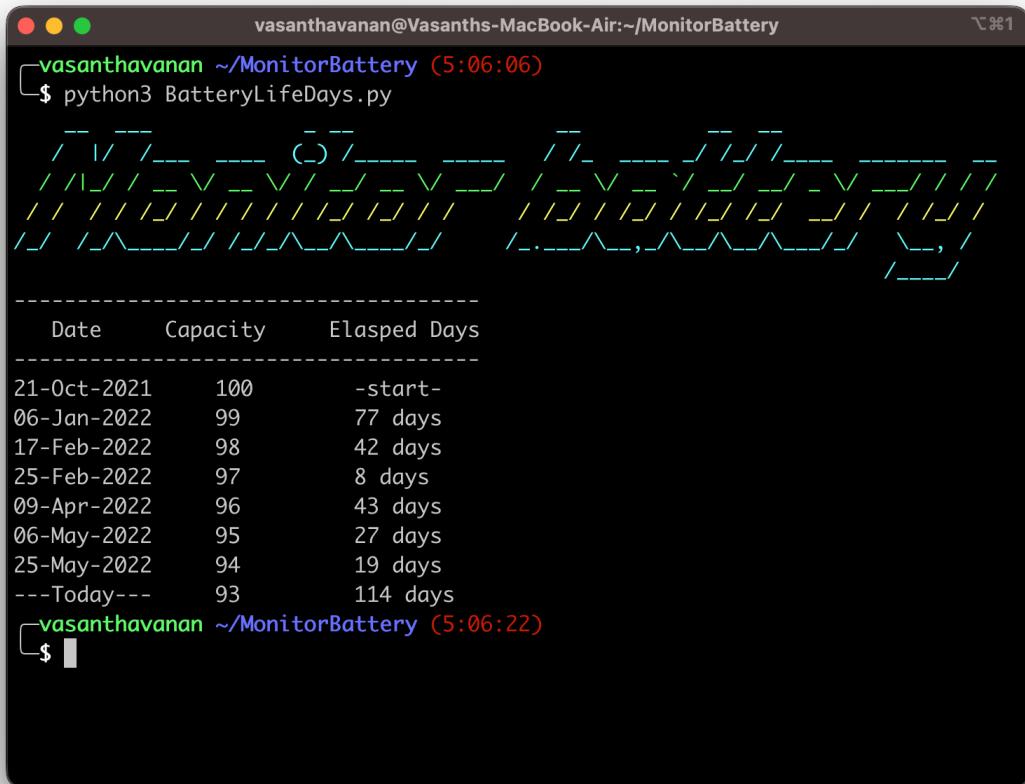


- **Note:** This is not automated. It will be triggered if the “Cycle Count” app is clicked.
- You can drag the application to Dock to run whenever needed.



7: Analysis on Number of lasting days for each cycle count

- Run **python3 BatteryLifeDays.py** in **~/MonitorBattery** folder to know the statistics of your MacBook’s Battery performance.
- **Note:** the more you use the Cycle Count application, the more data you will see on the table.



A terminal window titled "vasanthavanhan@Vasanths-MacBook-Air:~/MonitorBattery". The user runs the command \$ python3 BatteryLifeDays.py. The script outputs a series of ASCII art battery icons followed by a table of battery history. The table has columns for Date, Capacity, and Elapsed Days. The data shows the battery starting at 100% on 21-Oct-2021 and decreasing to 93% by the current date (---Today---). The script ends with a final ASCII art icon.

Date	Capacity	Elapsed Days
21-Oct-2021	100	-start-
06-Jan-2022	99	77 days
17-Feb-2022	98	42 days
25-Feb-2022	97	8 days
09-Apr-2022	96	43 days
06-May-2022	95	27 days
25-May-2022	94	19 days
---Today---	93	114 days

- Maximum Capacity for a MacBook battery can be 80%. This statistics clearly shows the number of days elapsed for each percentage.
- Monitor your MacBook's Battery for a long-lasting service. Good Luck!

Further Enquires & Support:

- Email: cr34u6rupg@pomail.net
- Instagram: [@vasanth_vanan](https://www.instagram.com/@vasanth_vanan)