Automate Presence Detection

At the moment, you have to run your presence detection script manually on the RPi. Ideally, your script would run automatically at a set interval, perhaps every minute.

Loop forever

Add an endless loop in the program that waits of 60 seconds after every successful call to the apr-scan() function. You can use the `sleep()` function in the `time` python package to do this.

- In `presence-detector.py` , add the following import statement at the top of the file:

```
from time import sleep
```

- Put the call to the `arp_scan()` function in an endless loop(`while True:`) and add code to suspend execution for 60 seconds(`sleep(60)`) on each iteration:

```
while True:
    arp_scan()
    sleep(60)
```

- Run the program again. This time your program should never stop and should repeat the scan every 60 seconds.

When things go wrong...

Usually, when perfoming any input/output or network connections with programming, it is a good idea to assume that at some stage the connection will fail or that the file you are trying to use might become unavailable. You can think of this as an "exceptional" event - you're not expecting it but it might happen. Also, if we are running a device remotely and will not be attending, it may be a good idea to start logging information if an error or excpetional event does occur so that you can debug it.

In Python, error handling at run time in done through the use of exceptions that are caught in try blocks and handled in except blocks. We can then using logging to save details of the error for debugging later.

- Add the following logging declaration to the `precence-detector.py` program

```
import logging

logging.basicConfig(filename='presence-detector.log',level=logging.INFO, format='%(asctime)s - %(message)s')
logging.info('Starting presencenan detector')
```

Currently, in our `presence-detector.py` program, if an error occurs in the arp-scan() function the program will terminate.

- In the arp-scan() function, surround the code with a try-except block that logs any errors that occur:

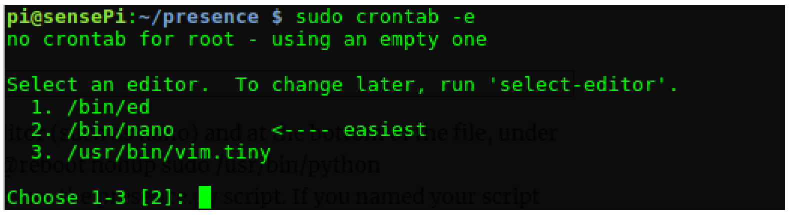
```
def arp_scan():
    try:
        output = subprocess.check_output("sudo arp-scan -l", shell=True)
        for i in range(len(names)):
            result = names[i]
            if macs[i] in output:
                result=result+" is home"
            else:
                result=result+" is not home"
            print(result)
            sense.show_message(result)
    except Exception as e:
        logging.error(e)
```

- Now run your program as before. It should work exactly as before but now you will have a log file in the same directory as your script. Check it's contents and you should see similar to the following:

```
2018-10-16 09:09:11,225 - Starting presence detector
```

Crontab

cron is a utility that allows tasks to be automatically run in the background at regular intervals. The Crontab (CRON TABle) is a file which contains the schedule of cron entries to be run and at specified times. The crontab File location varies by operating system however you can easily access it on the RPi using the `crontab` utility program.
In a terminal window, enter `crontab -e` at the prompt:



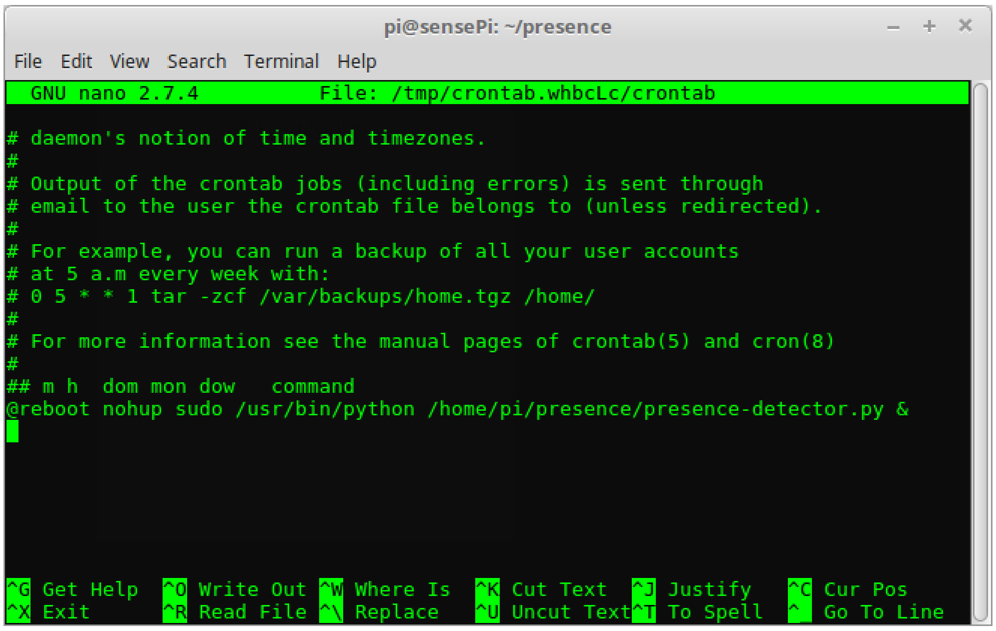
```
pi@sensePi:~/presence $ sudo crontab -e
no crontab for root - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        <---- easiest
  3. /usr/bin/vim.tiny

Choose 1-3 [2]: 
```

crontab

Follow the instructions and select your favourite editor (default is nano). At the end of the file, add `@reboot nohup sudo /usr/bin/python /home/pi/presence/presence-detector.py &` :



```
  GNU nano 2.7.4            File: /tmp/crontab.whbcLc/crontab

# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
## m h  dom mon dow   command
@reboot nohup sudo /usr/bin/python /home/pi/presence/presence-detector.py &






^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

crontab

Save and exit the Crontab and reboot the RPi by typing `sudo reboot` at the command prompt. The presence detector should now start in the background on every reboot of the RPi.

- Check the log file now just to see if everything is OK. You may notice some errors the first time the script executes. If you do, try to come up with a reason and investigate a solution using the web.
- If you do not get errors, how would you know if the simple exception handling/logging works? Propose a way to force an error in the try block of the code.