**Derek’s Code - Reviewed by Dominik, Irusha, and Amr**

Overall comments:

* No code to turn off alarm
* No code to handle button inputs

Missing metadata of name, date

from random import randint  
  
hr\_max\_threshold = 100  
hr\_min\_threshold = 50  
hr\_change\_threshold = 10  
#the largest change in heart rate before it is dangerous  
  
Add comment to describe alarms

alarm = False  
safe\_to\_turn\_off = True  
alarming = False  
  
Move Unit test to another test funciton.

(Use library unittest: <https://docs.python.org/2/library/unittest.html> )  
   
def alarm\_button\_pressed():  
 safe\_to\_turn\_off = True  
  
#the first time through sets alarming true  
#if alarming is still true, the alarm will be raised  
def alarm\_signal():  
 if alarming:  
 safe\_to\_turn\_off = False  
 alarm = True  
 print("I am alarmed")  
 #will also send a signal to the controller  
 else:  
 alarming = True  
   
def turn\_off\_alarm():  
 if safe\_to\_turn\_off:  
 alarm = False  
   
if \_\_name\_\_ == '\_\_main\_\_':  
 alarming = False  
 hr = next\_heart\_rate()  
 if hr < hr\_min\_threshold || hr > hr\_max\_threshold:  
 alarm\_signal()  
 last\_hr = hr  
 while True:  
 hr = next\_heart\_rate()  
 if hr < hr\_min\_threshold || hr > hr\_max\_threshold || abs(hr - last\_hr) > hr\_change\_threshold:  
 alarm\_signal()  
 else:  
 turn\_alarm\_off()  
 last\_hr = hr

**SYSC 3010 - Group B - Code Review Team Checklist**

* Functionality
  + Overall functionality works according to the project specifications
  + Input and Output network communications conform to established data protocol
  + Code straightforward to understand
  + Sensors working in the right way
  + No sections of code incomplete
* Comments
  + Meta data at the top (Name, Date Modified)
  + Comments are comprehensible and add something to the maintainability of the code
  + Comments are neither too numerous nor verbose
  + Comments are in the right place and they are useful
* Code smells
  + Tabs and brackets are consistent
  + Code as modular as possible
  + Repetitive code has been factored out
  + Command classes have been designed to undertake one task only
  + The code does not use unjustifiable static methods/blocks
  + Loops have a set length and correct termination conditions
  + Any unusual behavior or edge-case handling described
  + No hardcoded or Magic numbers present within reason
* Performance
  + No unnecessary loops
  + No possible replacement of recursive functions with sequential functions
  + the code was designed to perform as fast as possible
* Scope
  + No possible replacement of global variables to function variables
  + Variable types have been generalized where possible
  + Down casting used properly
* Unit Tests
  + Unit tests are present and correct
  + The code is unit testable
  + Testing the part that is implemented and checked for right performance
* Error/Exception handling
  + Common errors have been checked for
  + No zombie threads running
  + Any security concerns have been addressed
  + data goes through many filters that makes sure the data does not contain errors
* Logging/Debugging Info
  + Logging used appropriately (proper logging level and details)
  + Are all data inputs checked (for the correct type, length, format, and range) and encoded?
* Code Library Usage
  + Frameworks have been used appropriately
  + Can any of the code be replaced with library functions?
  + the code was designed using the libraries and there is no code to be replaced