```
"""goals8system.py

  This is the main script that coordinates both the controller and
  the detector.  Feel free to play/edit/...

"""

# Import the system parts
import threading
import traceback

# Import your pieces.  Change the "from" names (being the file names)
# to the file names of your two code parts.  Also, this is a great way
# to quickly switch which detector to run.  E.g. you could import from
# "facedetector" in place or "balldetector".
from goals9step6c   import controller
from goals9step6d import detector
# from goals8facedetector import detector     # Alternate option


#
#  Shared Data
#
class Shared:
    def __init__(self):
        # Thread Lock.  Always acquire() this lock before accessing
        # anything else (either reading or writing) in this object.
        # And don't forget to release() the lock when done!
        self.lock = threading.Lock()

        # Flag - stop the detection.  If this is set to True, the
        # detection should break out of the loop and stop.
        self.stop = False

        # Data - PLEASE UPDATE TO ADD THE DATA YOU NEED!
        self.newdata = False # Flag to indicate fresh, new data
        self.deltapan = 0.0 # Relative pan angle
        self.deltatilt = 0.0 # Relative tilt angle
        self.scalepan = 0.0
        self.sharetilt = 0.0
        self.balls = []

#
#  Main Code
#
def main():
    # Prepare the shared data object.
    shared = Shared()

    # Create a second thread.
    thread = threading.Thread(target=detector, args=(shared,))

    # Start the second thread with the detector.
    print("Starting second thread")
    thread.start()       # Equivalent to detector(shared) in new thread

    # Use the primary thread for the controller, handling exceptions
    # to gracefully to shut down.
    try:
        controller(shared)
    except BaseException as ex:
        # Report the exception
        print("Ending due to exception: %s" % repr(ex))
        traceback.print_exc()

    # Stop/rejoin the second thread.
    print("Stopping second thread...")
    if shared.lock.acquire():
        shared.stop = True
```

```python
    thread.join()       # Wait for thread to end and re-combine.

if __name__ == "__main__":
    main()
```