```python
"""goals7system.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 goals7motor   import controller
from goals7camera     import detector
# from goals7facedetector 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

        # Motor data
        self.motorpan  = 0.0
        self.motortilt = 0.0

        # Object Data - PLEASE UPDATE TO ADD THE DATA YOU NEED!
        self.objectpan = 0.0
        self.objecttilt = 0.0

        self.newdata = False


#
#   Main Code
#
def main():
    # Prepare a single instance of 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...")
```

```
71        if shared.lock.acquire():
72            shared.stop = True
73            shared.lock.release()
74        thread.join()          # Wait for thread to end and re-combine.
75
76    if __name__ == "__main__":
77        main()
```