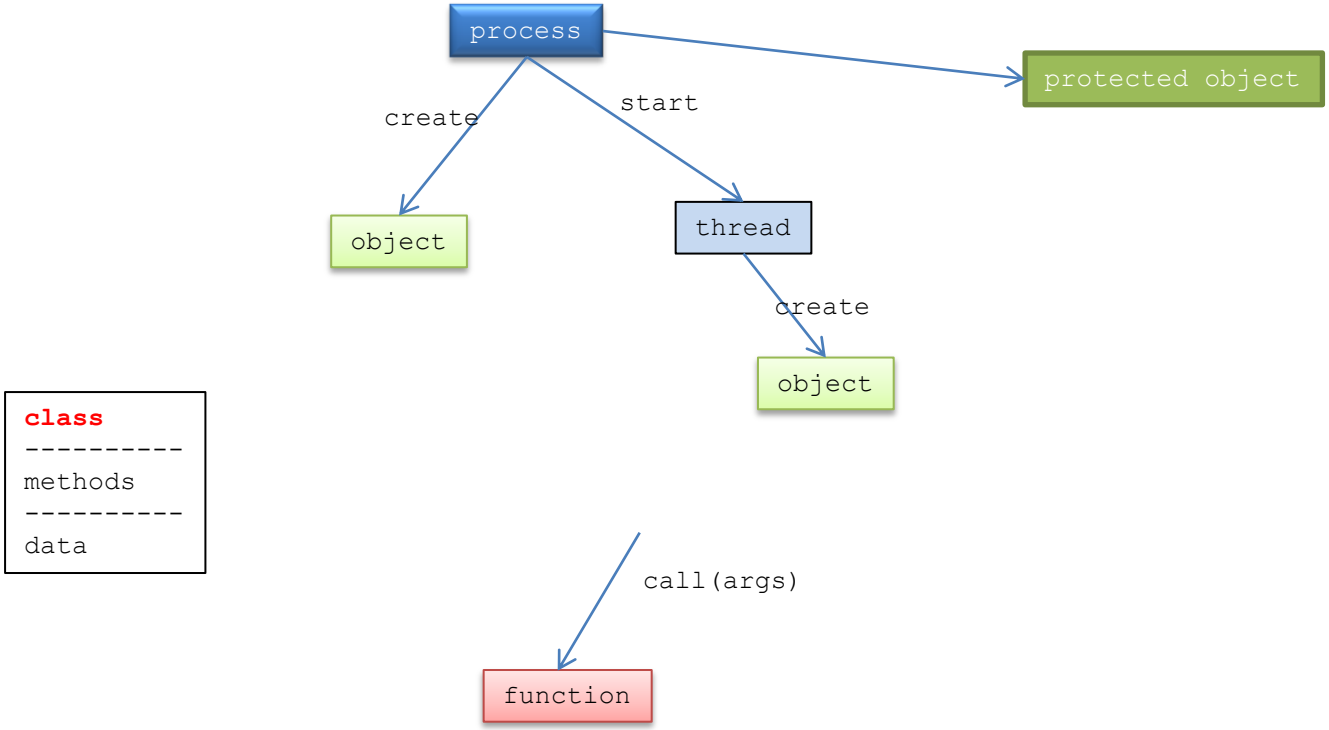
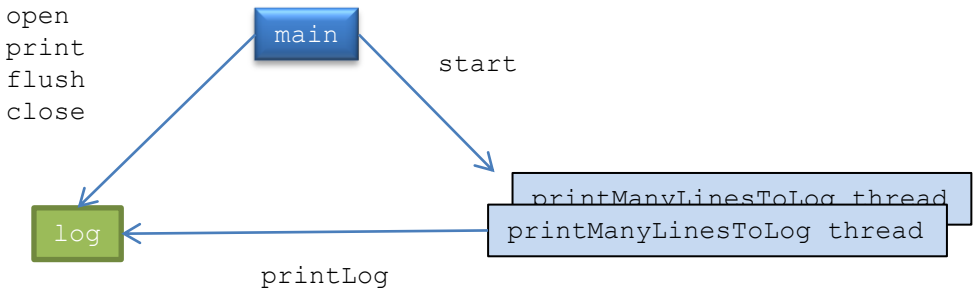
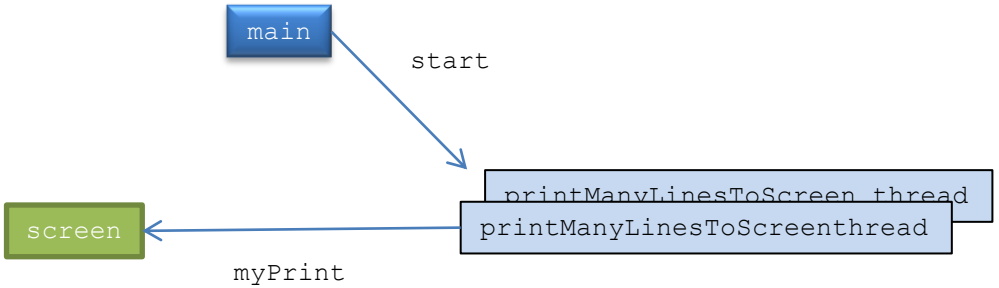
Scripting Design Diagrams


process
protected object
create
start
object
thread
create
object
class
methods
data
call(args)
function

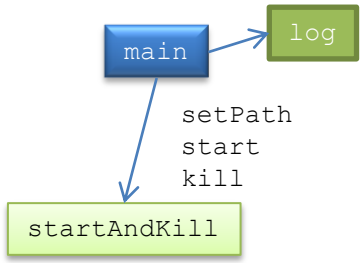
Log_Test.py



MyPrint_Test.py



StartAndKill_Test.py



TCP/IP

SCENARIO
o Server creates socket and listens for clients.
    -- Client creates socket, connects socket to server, sends 1 message, and closes socket.
o Server accepts connection, spins off a socket for the connection,
  receives 1 message on this socket, and closes the socket.

```
SERVER SIDE
listenSk = socket(AF_INET, SOCK_STREAM)
listenSk.bind(myIP, myPort)
listenSk.listen(5)
while ???:
   clientSk, (otherIP, otherPort) = listenSk.accept()
   message = decode(clientSk.recv(1024), "ascii")
   do something with message
   clientSk.close()
listenSk.close()
```

```
CLIENT SIDE
sk = socket(AF_INET, SOCK_STREAM)
sk.connect(self.otherIP, otherPort)
sk.send(bytes(message, 'ascii'))
sk.close()
```

TCP/IP

SCENARIO
o Similar to previous except multiple messages are passed between server and client.

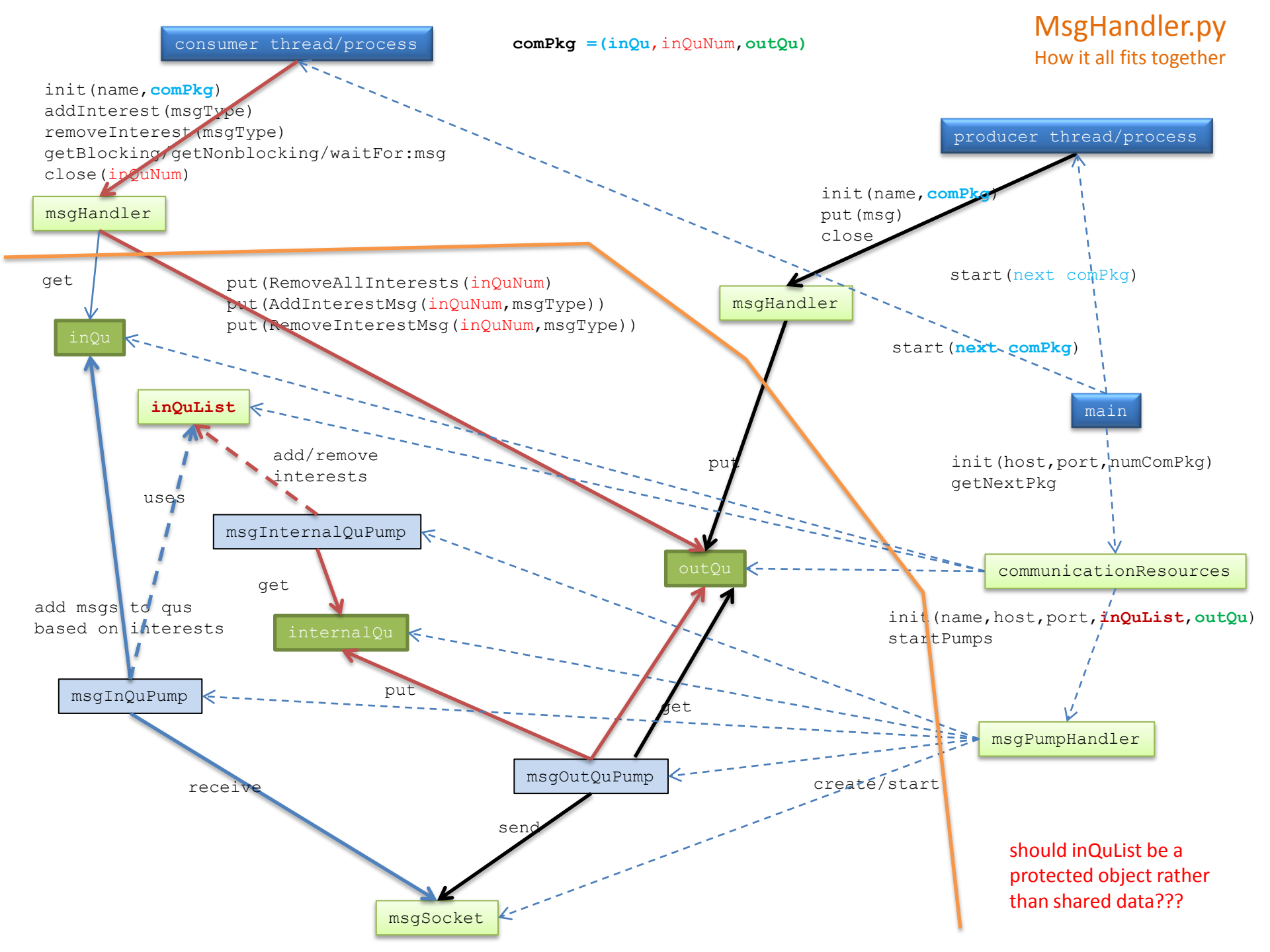
```
SERVER SIDE
listenSk = socket(AF_INET, SOCK_STREAM)
listenSk.bind(myIP, myPort)
listenSk listen(5)
while ???:
    clientSk, (otherIP, otherPort) = listenSk.accept()
    clientHandler = ClientHandler(clientSk)
    clientHandler.start()
listenSk.close()
```

```
CLIENT SIDE
sk = socket(AF_INET, SOCK_STREAM)
sk.connect(self.otherIP, otherPort)
while ???:
    ...
    sk.send(bytes(message, 'ascii'))
    ...
    message = decode(sk.recv(1024), "ascii")
    ...
sk.close()
```

```
CLIENT HANDLER
while ???:
    ...
    message = decode(clientSk.recv(1024), "ascii")
    ...
    clientSk. sk.send(bytes(message, 'ascii'))
    ...
 clientSk.close()
```

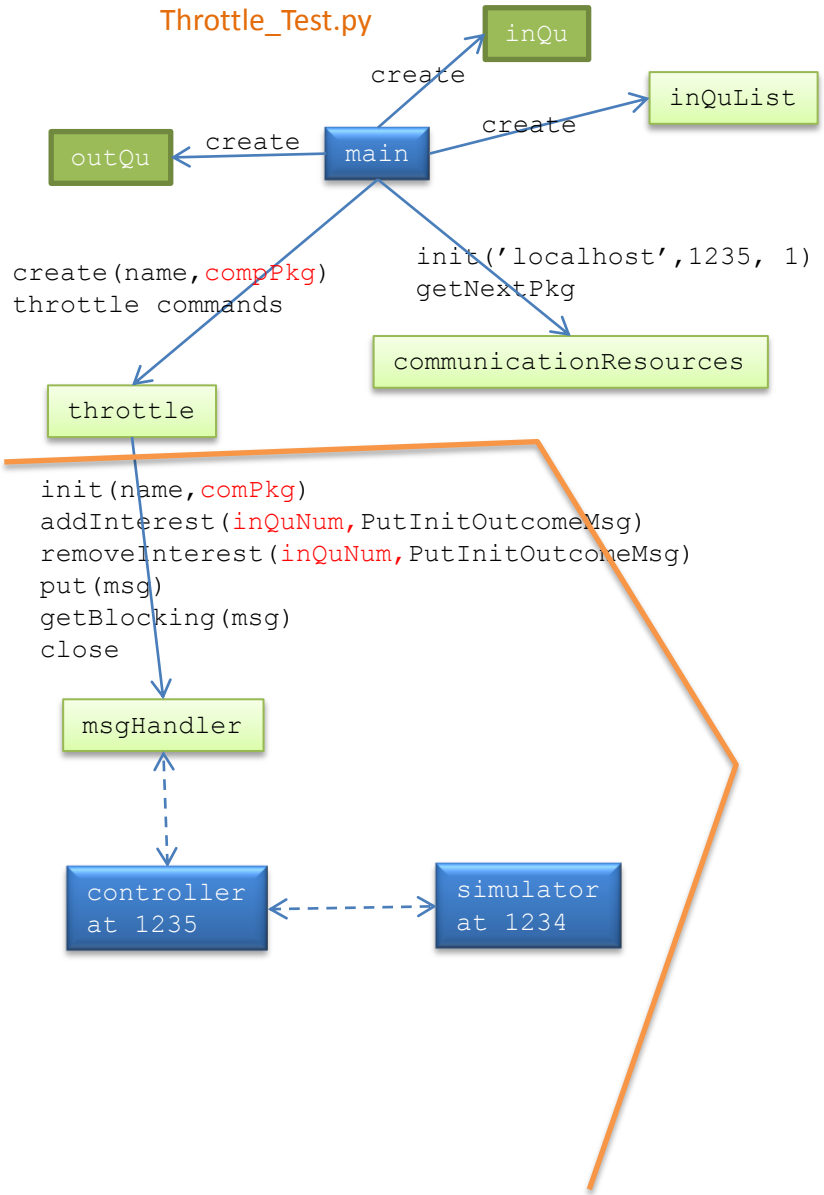
MsgHandler.py

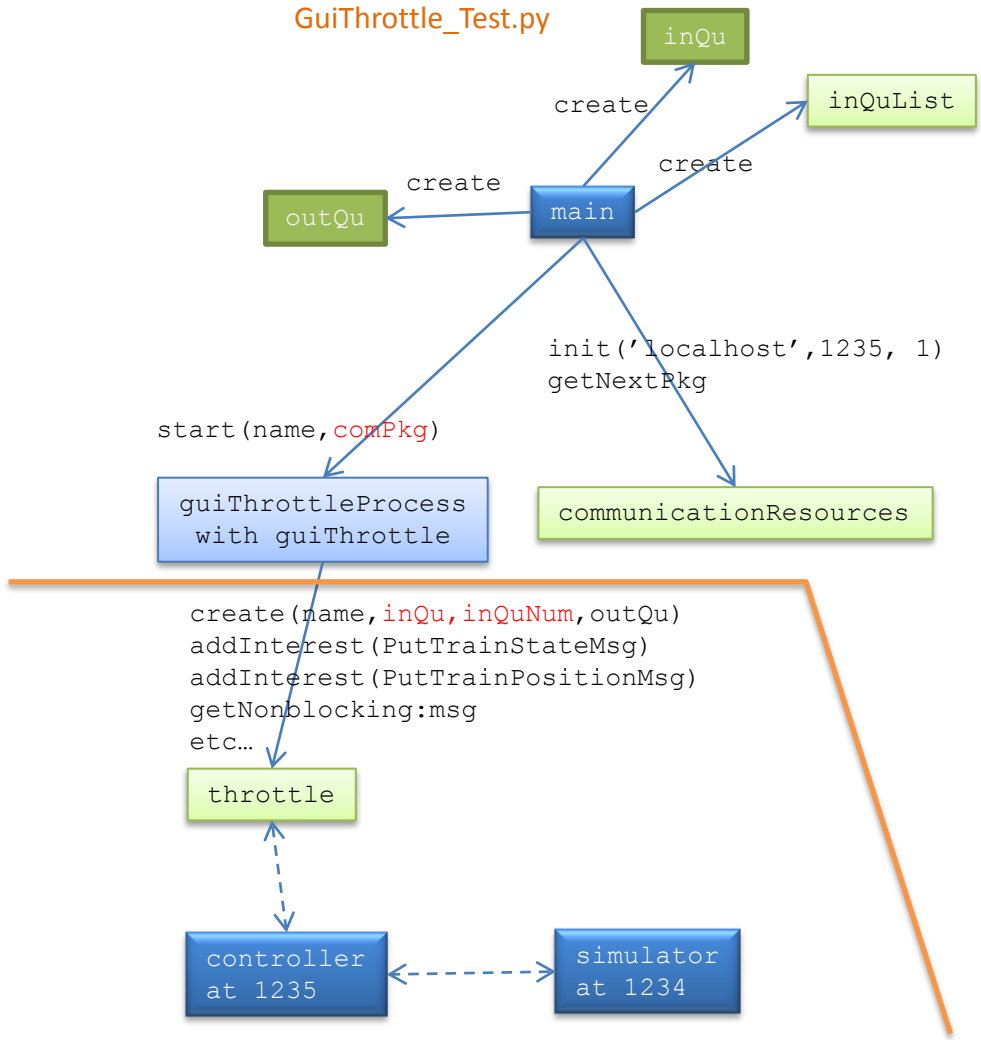
How it all fits together



MsgHandler.py

How use a Throttle and GuiThrottle

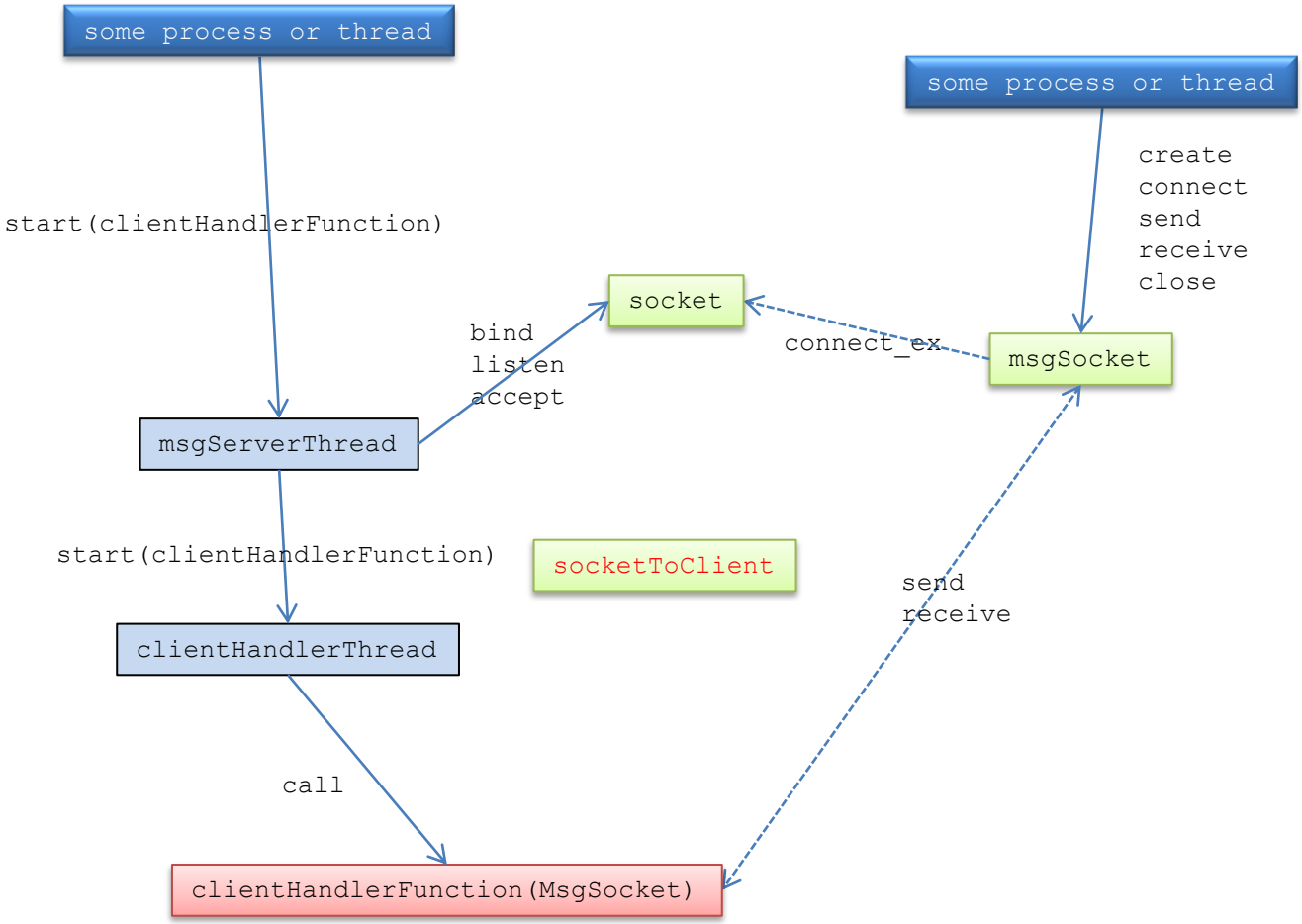




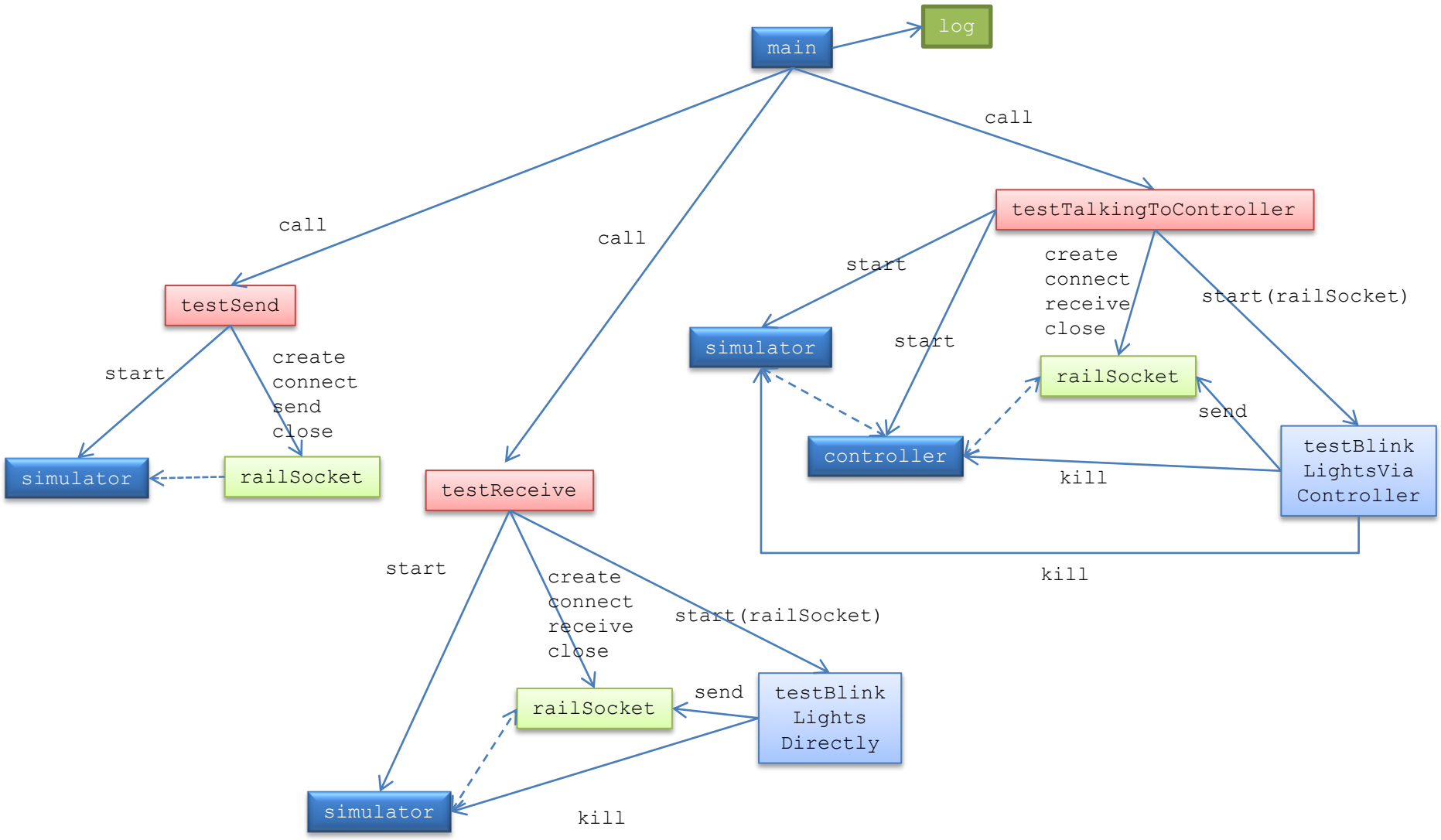
MsgHandler.py

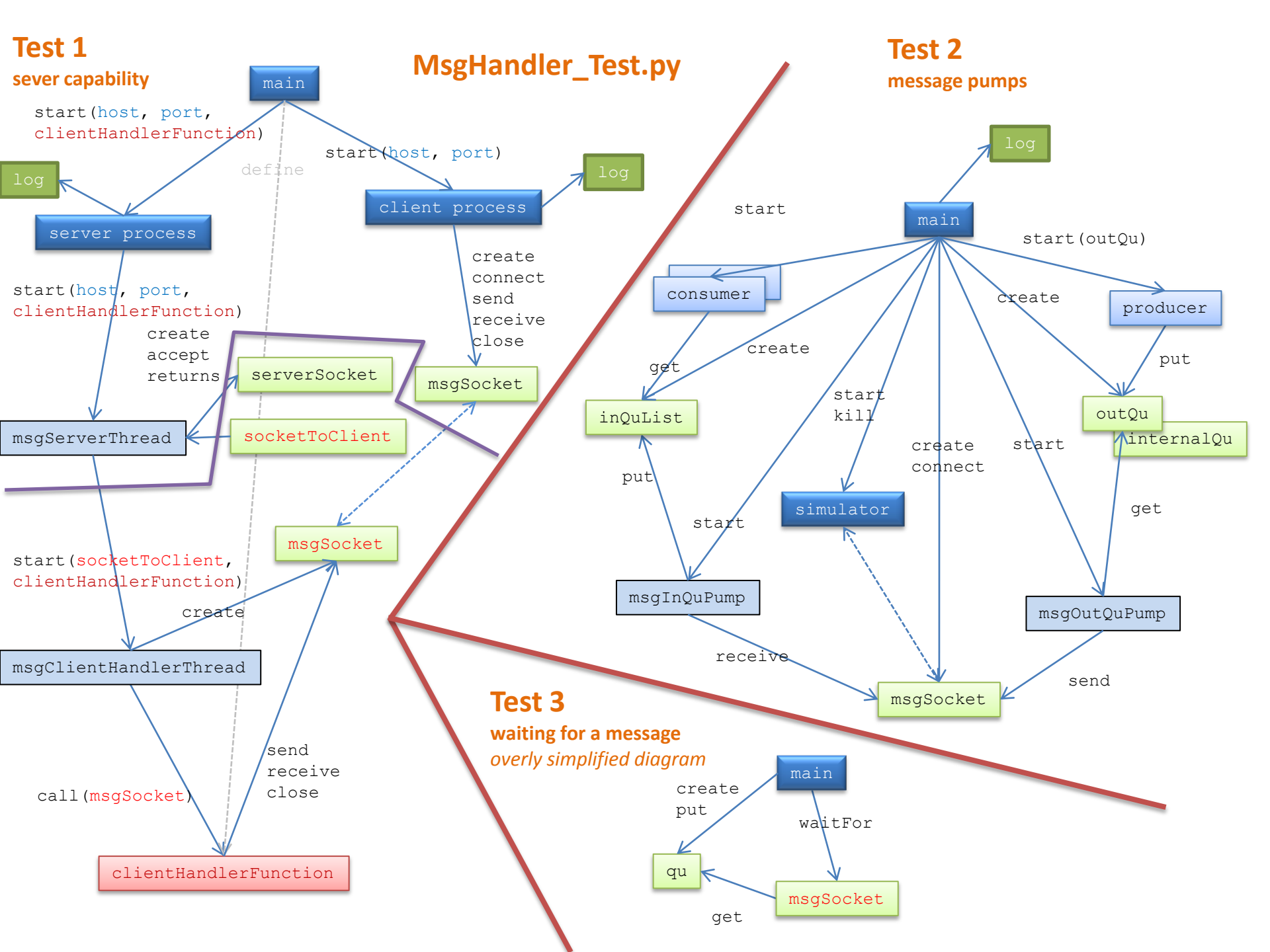
usage

MsgServerThread and MsgSocket



MsgSocket_Test.py




MsgHandler_Test.py
Test 1
sever capability
main
start(host, port, clientHandlerFunction)
define
log
server process
start(host, port)
client process
log
create
connect
send
receive
close
start(host, port, clientHandlerFunction)
create
accept
returns
serverSocket
msgSocket
msgServerThread
socketToClient
msgSocket
start(socketToClient, clientHandlerFunction)
create
msgClientHandlerThread
send
receive
close
call(msgSocket)
clientHandlerFunction
Test 2
message pumps
log
start
main
start(outQu)
consumer
create
producer
get
create
put
start
kill
inQuList
outQu
internalQu
create
connect
start
put
start
simulator
get
msgInQuPump
msgOutQuPump
receive
send
msgSocket
Test 3
waiting for a message
overly simplified diagram
main
create
put
waitFor
qu
msgSocket
get