Activity 1:

lec10act1.py

```python
from collections import namedtuple


Student = namedtuple('Student', ['name', 'age', 'major', 'gpa'])

students = []
students.append(Student("jordan", 18, "CS", 4.0))
students.append(Student("jordan", 18, "CS", 3.5))
students.append(Student("jordan", 18, "CS", 3.0))

s = 0
for student in students:
    s += student.gpa

print(s/3)
```

makefile() - Returns file-like object, buffering optional.
send(data) - Sends bytes, returns sent length.
recv(bufsize) - Receives bytes, returns received data.
bind(address) - Binds socket to (host, port).
listen(backlog) - Enables listening for connections.
accept() - Returns a tuple (client socket, address).
close() - Closes the socket connection.
gethostname() - Returns local hostname string.
gethostbyname(hostname) - Resolves hostname to IP.

Activity 2:

```python
def cmd_lowercase(msg: bytes) -> bytes:
    '''
    Makes the stirng lowercase
    '''
    return msg.lower()


def message_exchange(si32p_conn: SI32PConnection) -> None:
    while True:
        rec_msg = si32p.listen(si32p_conn)
        print("message: ", rec_msg)
        try:
            if is_command_recognized(rec_msg):
                # If command is INVERT
                if rec_msg.startswith(si32p.SI32P_CLI_INVERT):
                    msg_data = process_command(rec_msg, si32p.SI32P_CLI_INVERT)
                    print (msg_data)
                    if msg_data:
                        si32p.send(si32p_conn, cmd_invert(msg_data))
                        si32p.complete(si32p_conn)
                    else:
                        si32p.send(si32p_conn, b"error")
                        si32p.complete(si32p_conn)

                # If command is LOWERCASE
                elif rec_msg.startswith(si32p.SI32P_CLI_LOWERCASE):
                    msg_data = process_command(rec_msg, si32p.SI32P_CLI_LOWERCASE)
                    if msg_data:
                        si32p.send(si32p_conn, cmd_lowercase(msg_data))
                        si32p.complete(si32p_conn)
                    else:
                        si32p.send(si32p_conn, b"error")
                        si32p.complete(si32p_conn)

                # If command is BYE
                elif rec_msg.startswith(si32p.SI32P_CLI_BYE):
                    si32p.send(si32p_conn, si32p.SI32P_SRV_BYE)
                    si32p.disconnect(si32p_conn)
                    break

            else:
                # If command was not recognized
                si32p.send(si32p_conn, si32p.SI32P_UNREC)
        except SI32PProtocolError:
            print("Error: Client sent payload out of SI32P specification.")
            si32p.disconnect(si32p_conn)
            break
        except Exception as e:
            print(e)
            break
```