Python Socket Programming

Python Execution Concurrency

- Native polling with non-blocking sockets
- Sockets with timeouts
- Threading
- Multiprocessing
- Select

Socket Programming with Select

Concurrency with Select

- Select monitors sockets until they become readable, writable, or a communication error occurs.
- Select makes it easier to monitor multiple concurrent connections, i.e., it is more efficient than writing a native polling loop in Python using socket timeouts or non-blocking sockets (as we discussed previously).
- This is because the monitoring happens in the operating system network layer, instead of in the Python interpreter.

Socket Programming with Select()

- A classic way of server socket programming is for the main code to listen for new connections, then spawn a new execution task whenever a new client connects, e.g., using multiprocessing, threading, etc.
- Select is a different way of handling multiple clients by having one process multiplex the I/O servicing of the clients, behind the scenes.
- The advantage of this is that your server code does not need to create and manage multiple tasks.
- A disadvantage is that unlike the separate task case, where the new task need not concern itself with other clients, the code using select must be aware of them.

Select Arguments

- The main arguments to select are three lists that contain the communication channels (i.e., sockets) to monitor.
  - The first is a list of the sockets to be checked for incoming data that is ready to be read.
  - The second is a list of sockets that will receive outgoing data when there is room in their buffer.
  - The third is a list of sockets that may have an error, i.e., typically a combination of input and output sockets.
  - Normally select blocks until a socket is available for processing. A fourth argument will set a timeout so that other processing can occur instead.

Select Arguments

- Assume we have sockets to read from and write to: **`read_socket, write_socket`**.

```
import select
read_list = [read_socket]
write_list = [write_socket]
```

- Call select:

```
read_ready, write_ready, except_ready =
     select.select(read_list, write_list, [])
```

- Then iterate over the outputs obtained.

Select

- see get_message_eom_select.py
- see coe4dn4_python_queue_v01.odp
- see queue_example.py
- See echo_server_select_queues.py