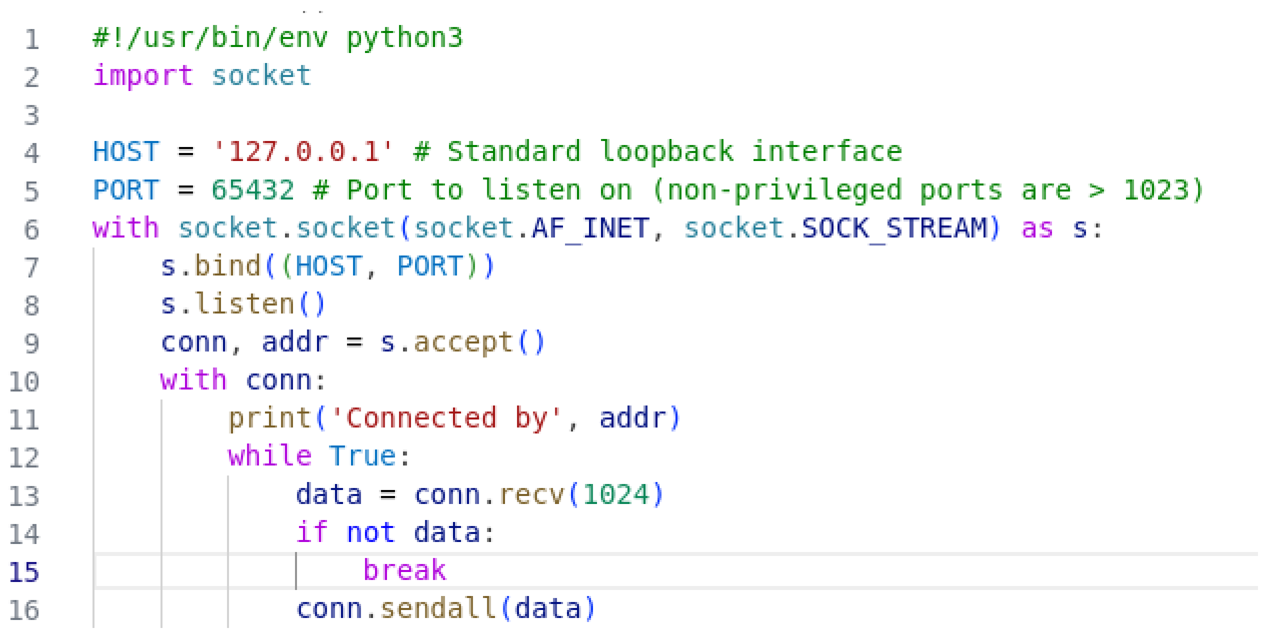
# Codio Activity: Socket Programming

When we communicate over a distributed system, we need to use sockets to control the message sending process. In this activity, you will create sockets and communicate between distributed devices.

The code from the following activity has been sourced from Realpython.com. It is available for you in the Codio workspace – Socket Programming.

Copy the following code into a file named echo-server.py:

```
#!/usr/bin/env python3
import socket
HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

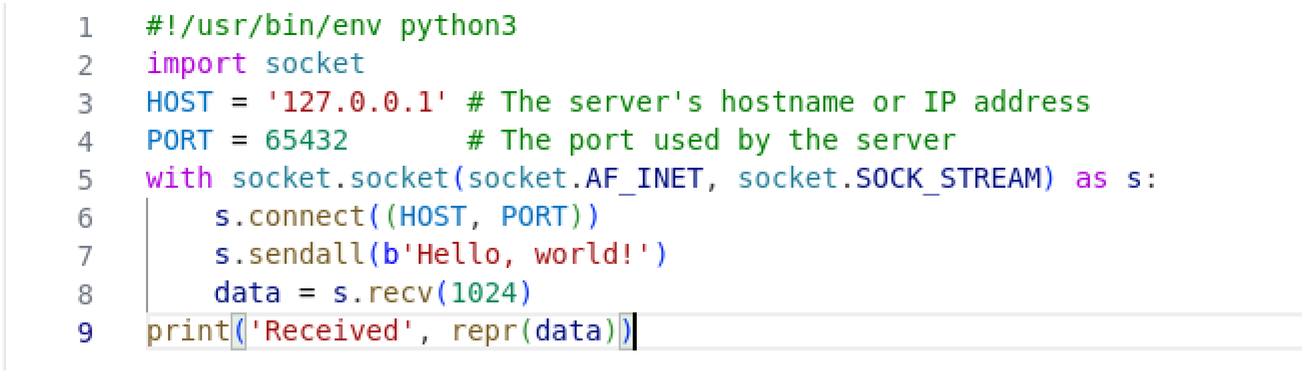


```python
#!/usr/bin/env python3
import socket

HOST = '127.0.0.1' # Standard loopback interface
PORT = 65432 # Port to listen on (non-privileged ports are > 1023)
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

Copy the following code into a file named echo-client.py:

```
#!/usr/bin/env python3
import socket
HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 65432        # The port used by the server
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world')
    data = s.recv(1024)
print('Received', repr(data))
```



```python
#!/usr/bin/env python3
import socket
HOST = '127.0.0.1' # The server's hostname or IP address
PORT = 65432       # The port used by the server
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world!')
    data = s.recv(1024)
print('Received', repr(data))
```

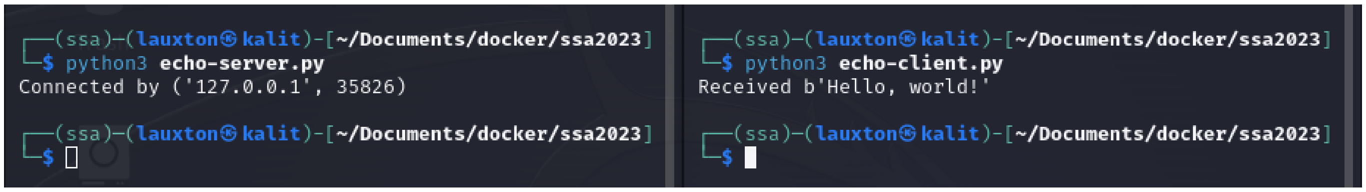
Open a terminal. Start the server by running the following command in a terminal:

```
python3 echo-server.py
```

Open a second terminal. Start the client by running the following command in the terminal:

```
python3 echo-client.py
```

The client and server will now talk with one another.



```
┌──(ssa)─(lauxton㉿kalit)-[~/Documents/docker/ssa2023]
└─$ python3 echo-server.py
Connected by ('127.0.0.1', 35826)

┌──(ssa)─(lauxton㉿kalit)-[~/Documents/docker/ssa2023]
└─$ 
```

```
┌──(ssa)─(lauxton㉿kalit)-[~/Documents/docker/ssa2023]
└─$ python3 echo-client.py
Received b'Hello, world!'

┌──(ssa)─(lauxton㉿kalit)-[~/Documents/docker/ssa2023]
└─$ 
```

## Question 1

In relation to echo-server.py, what is achieved using the command:

`s.bind((HOST, PORT))`?

It appears that the `s.bind` command is requesting to bind a the socket 's' to a local port (in this case '65432') (Python, nd). In other words, `bind` takes whatever socket is defined as 's' and binds it to whatever HOST and PORT has been defined so that the socket can connect and communicate the with port / server (if applicable).

## Question 2

In relation to echo-server.py, what is achieved using the command:

`s.connect((HOST, PORT))`?

`connect` here would be the next step after a successful `bind` attempt, which would then connect the server to the HOST/PORT and leave it open for data transmission. Python (n.d.) provides the metaphor of a docking cargo ship: `bind` is the ship (socket) entering the cargo port while `connect` would be the opening of the ship's doors at dock to unload / reload whatever contents are meant to be delivered.

## Resources

Python (n.d.) *Binding and Listening with Sockets* | Python Programming. pythonprogramming.net. [Available Online] https://pythonprogramming.net/python-binding-listening-sockets/

## Learning Outcomes

- Identify and critically analyse operating system risks and issues, and identify appropriate methodologies, tools and techniques to solve them.
- Evaluate and adapt platforms and systems, using processes such as code refactoring, to produce secure distributed system solutions.

- Critically analyse and evaluate solutions produced.