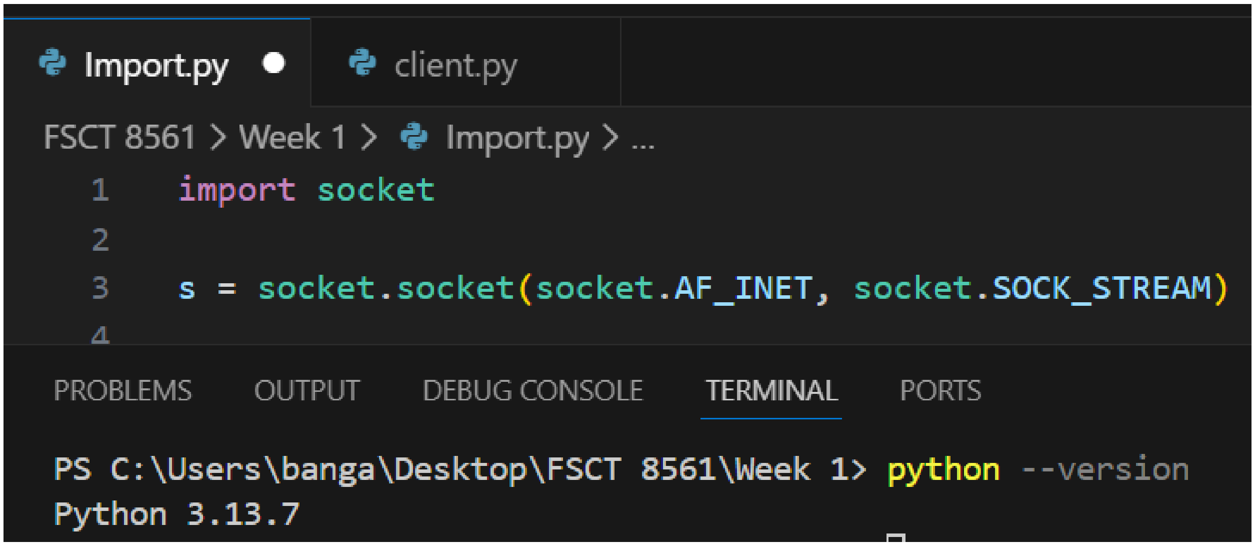
FSCT 8561 – Lab 0: Introduction to Network Protocols using Python Sockets

By: Jose Bangate, A01271709

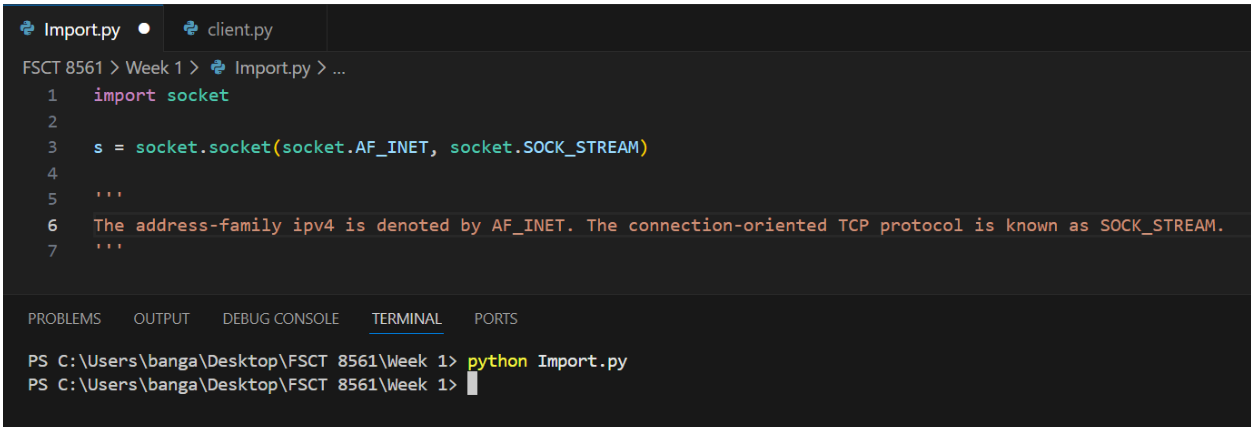
Task 1: Environment Verification



```python
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

```

```
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> python --version
Python 3.13.7
```

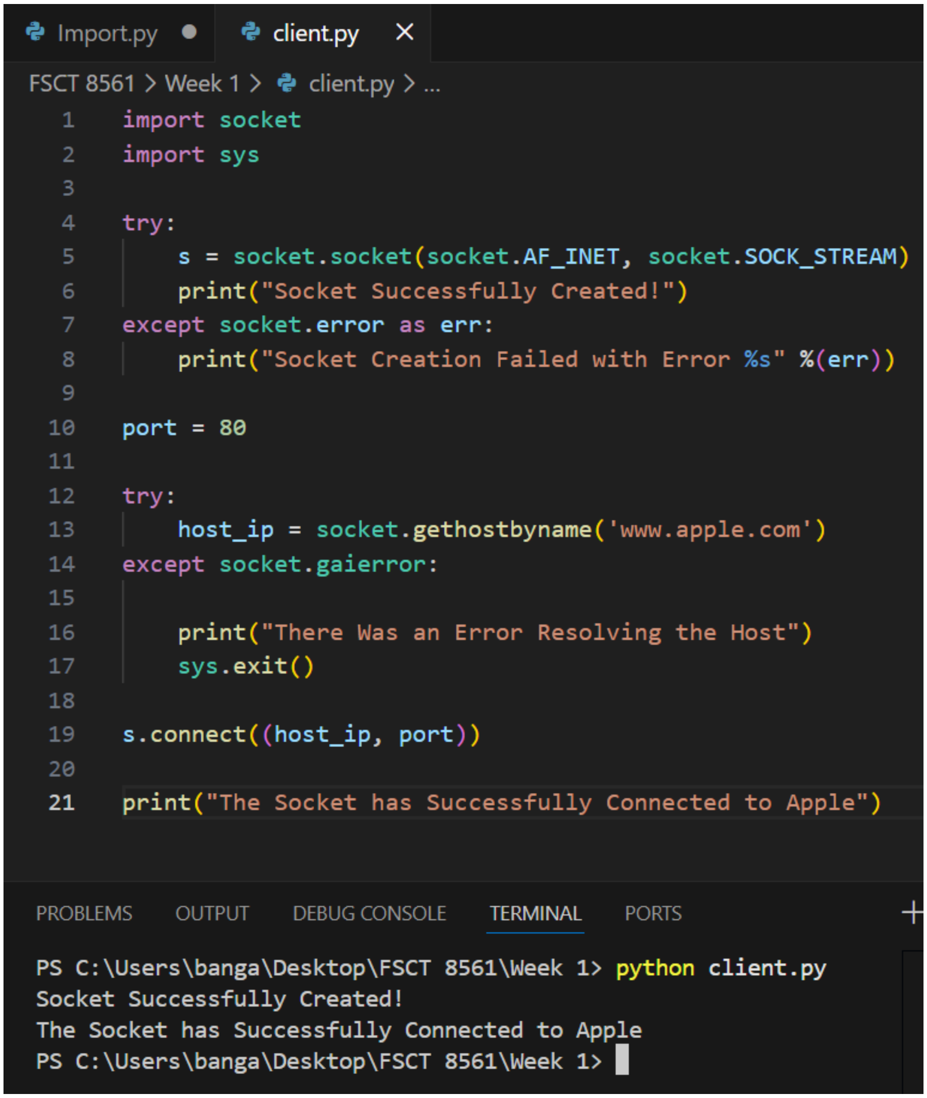
Task 2: Understanding Sockets



```python
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

'''
The address-family ipv4 is denoted by AF_INET. The connection-oriented TCP protocol is known as SOCK_STREAM.
'''
```

```
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> python Import.py
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> 
```

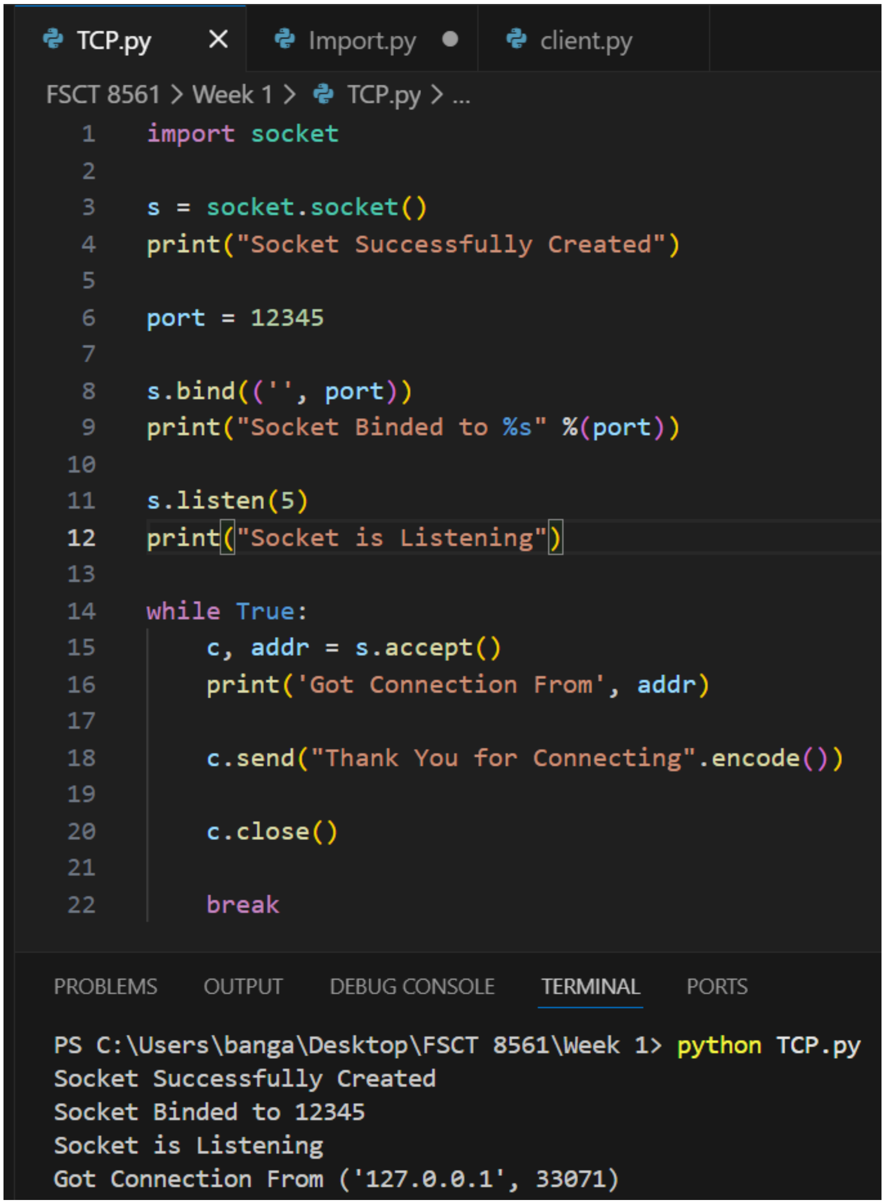
Task 3: Simple Client Connection



```python
import socket
import sys

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Socket Successfully Created!")
except socket.error as err:
    print("Socket Creation Failed with Error %s" %(err))

port = 80

try:
    host_ip = socket.gethostbyname('www.apple.com')
except socket.gaierror:

    print("There Was an Error Resolving the Host")
    sys.exit()

s.connect((host_ip, port))

print("The Socket has Successfully Connected to Apple")
```

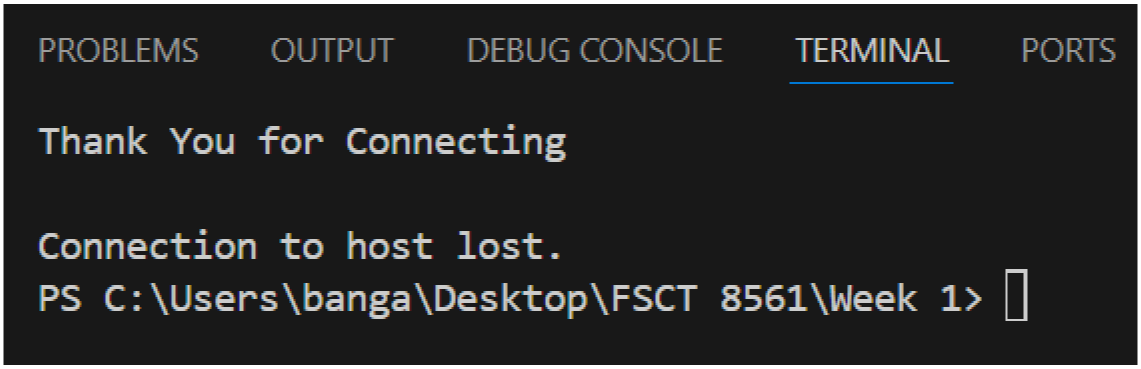
```
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> python client.py
Socket Successfully Created!
The Socket has Successfully Connected to Apple
PS C:\Users\banga\Desktop\FSCT 8561\Week 1>
```

Task 4: Create a TCP Server:



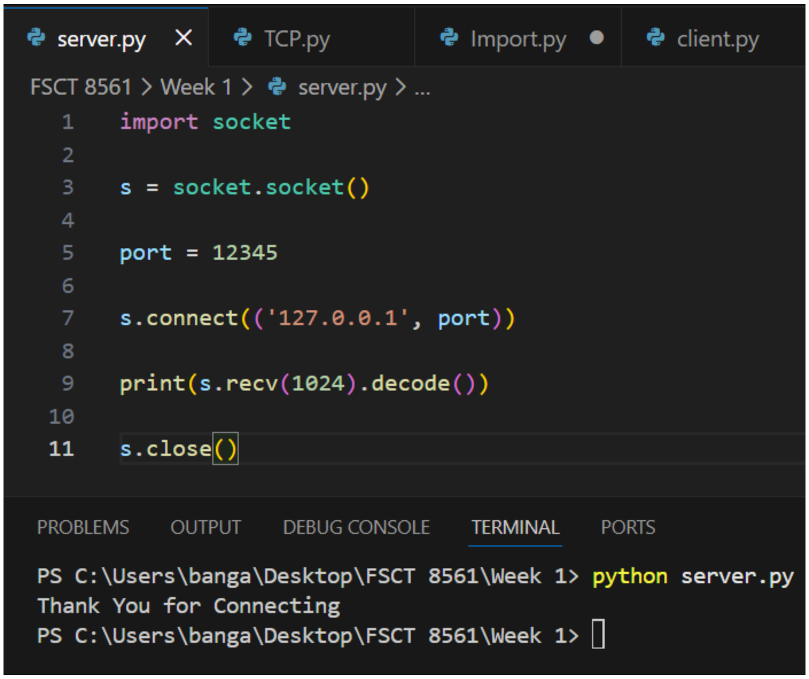
```python
import socket

s = socket.socket()
print("Socket Successfully Created")

port = 12345

s.bind(('', port))
print("Socket Binded to %s" %(port))

s.listen(5)
print("Socket is Listening")

while True:
    c, addr = s.accept()
    print('Got Connection From', addr)

    c.send("Thank You for Connecting".encode())

    c.close()

    break
```

```
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> python TCP.py
Socket Successfully Created
Socket Binded to 12345
Socket is Listening
Got Connection From ('127.0.0.1', 33071)
```



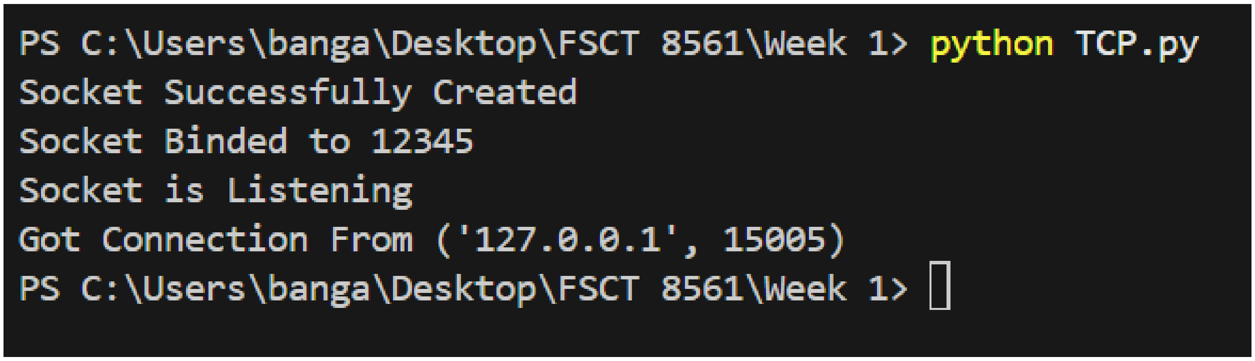
```
Thank You for Connecting

Connection to host lost.
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> 
```

Task 5: Create a TCP Client



```python
import socket

s = socket.socket()

port = 12345

s.connect(('127.0.0.1', port))

print(s.recv(1024).decode())

s.close()
```

```
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> python server.py
Thank You for Connecting
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> 
```



```
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> python TCP.py
Socket Successfully Created
Socket Binded to 12345
Socket is Listening
Got Connection From ('127.0.0.1', 15005)
PS C:\Users\banga\Desktop\FSCT 8561\Week 1> 
```

Task 6: Security Reflection

- **Three Security Risks**

1. **Sensitive Data Exposure:** Programs that use raw sockets have access to all incoming packets, not just their own, which raises the possibility of inadvertently logging or exposing other traffic, which could result in the leakage of network data from other programs.
2. **Privilege Escalation & Misuse:** To generate raw sockets on most systems, you need administrator or root rights. The impact of vulnerability or exploits is

increased when code is executed with higher privileges, giving attackers more control.

3. **Packet Spoofing & Injection Risk:** Attackers can utilize raw sockets to create arbitrary packets, including ones with fake source addresses, for denial-of-service, man-in-the-middle, and session hijacking attacks.

- **Input Validation, Access Control, and Protocol Awareness**
  1. By preventing malicious or corrupted data from being processed or sent, input validation helps prevent attacks like buffer overflow and protocol misuse that could compromise systems or crash services. Only well-formed packets or parameters are accepted when network inputs are validated.
  2. Access control makes sure that resources on a network can only by accessed or communicated with by authorized users, devices, or services. Robust access control restricts unwanted access and lowers the possibility of breaches and attackers' lateral movement.
  3. Understanding and upholding the anticipated behaviour of network protocols is known as protocol awareness. It lowers the possibility of protocol abuse, data damage, or security flaws brought on by erroneous messages by assisting systems in identifying and rejecting unexpected or non-compliant traffic.
- **Secure your Simple Client-server Application**
  1. **Using Encryption (TLS/SSL):** To prevent eavesdropping and manipulation of data in transit, encrypt all network traffic using TLS (e.g., via ssl modules or frameworks that enable HTTPS).