Python: Networking and Databases

Diptesh Kanojia

e-Yantra Team
ERTS Lab, IIT Bombay

IIT Bombay
May 3, 2021

Agenda for Discussion

1. Python: Network Programming
   - The What! and the Why?
   - The How?
2. Python: Databases
   - The what! and the Why?

What is "Network Programming"

Networking services between two machines allow them to establish connections. These connections can serve various purposes like sending and receiving messages, sharing files *etc.*
Python provides two levels of access to network services.

1. Low-level Access
2. High-level Access

At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connection-less protocols. Application level network protocols can also be accessed using high-level access provided by Python libraries. These protocols are HTTP, FTP, etc.

Socket Connections: What is it?

- A socket is the end-point in a flow of communication between two programs or communication channels operating over a network. They are created using a set of programming request called socket API (Application Programming Interface).
- Python's socket library offers classes for handling common transports as a generic interface. Sockets use protocols for determining the connection type for port-to-port communication between client and server machines.
- The protocols are used for:
  - Domain Name Servers (DNS)
  - IP addressing
  - E-mail
  - FTP (File Transfer Protocol) etc...

Socket Connections: Modules

After you defined the socket, you can you can use several methods to manage the connections. Some of the important server socket methods are:

- `listen()`: is used to establish and start TCP listener.
- `bind()`: is used to bind address (host-name, port number) to the socket.
- `accept()`: is used to TCP client connection until the connection arrives.
- `connect()`: is used to initiate TCP server connection.
- `send()`: is used to send TCP messages.
- `recv()`: is used to receive TCP messages.
- `sendto()`: is used to send UDP messages
- `close()`: is used to close a socket.

How to establish it?

Python has socket method that let programmers' set-up different types of socket virtually. An sample code snippet to help establish such a connection would be:

```
g = socket.socket(socket_family, type_of_socket, protocol=value)
```

For example, if we want to establish a TCP socket, we can write the following short code snippet:

```
# imports everything from 'socket'
from socket import *
# use 'socket.socket()' - the 'function'
tcp1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Understanding Socket Connections

- Each machine has a unique IP Address.
- Each Socket connection is established on a specific port and once a port is in use, it should not be used for another connection.
- Each connection needs to be maintained on both the 'ends'.
- The Socket API in Python allows you to send and recieve data.
- Types of Socket are:
  1. Stream Sockets (Connection oriented/TCP) [SOCK_STREAM]
  2. Datagram Sockets (Connection-less/UDP) [SOCK_DGRAM]
  3. Others (Raw Sockets)
- In the following slide, we shall see an example usage of the stream socket usage.

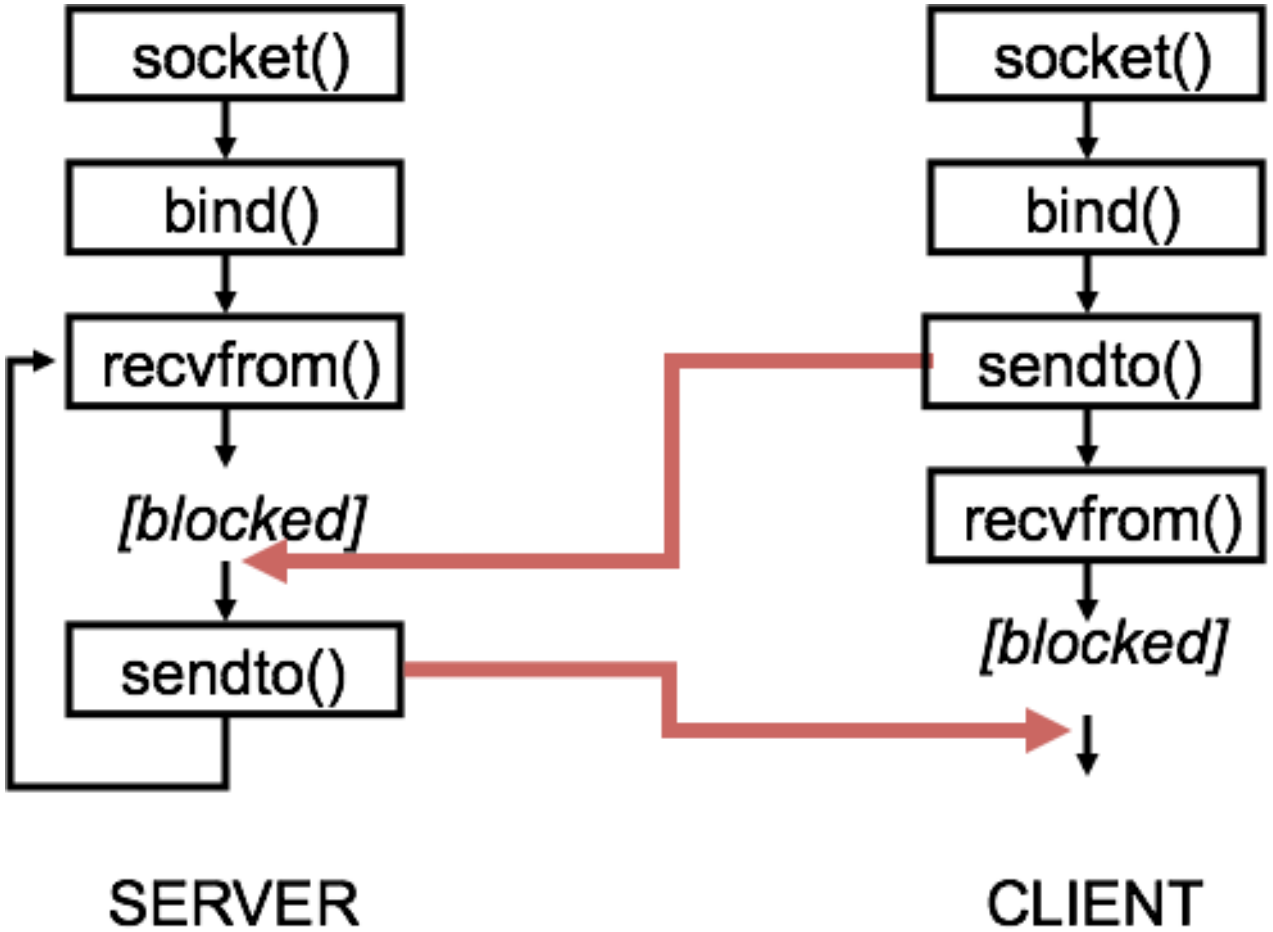
Socket Connection: Code Snippet [Host Machine]

```
import socket

HOST = socket.gethostname()
PORT = 10002 # You can choose any port which is not in use.

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)

connection, address = s.accept()
print("Connected by Client IP: ", address)

while True:
    data = connection.recv(1024)
    if not data: break
    connection.send(data)
connection.close()
```

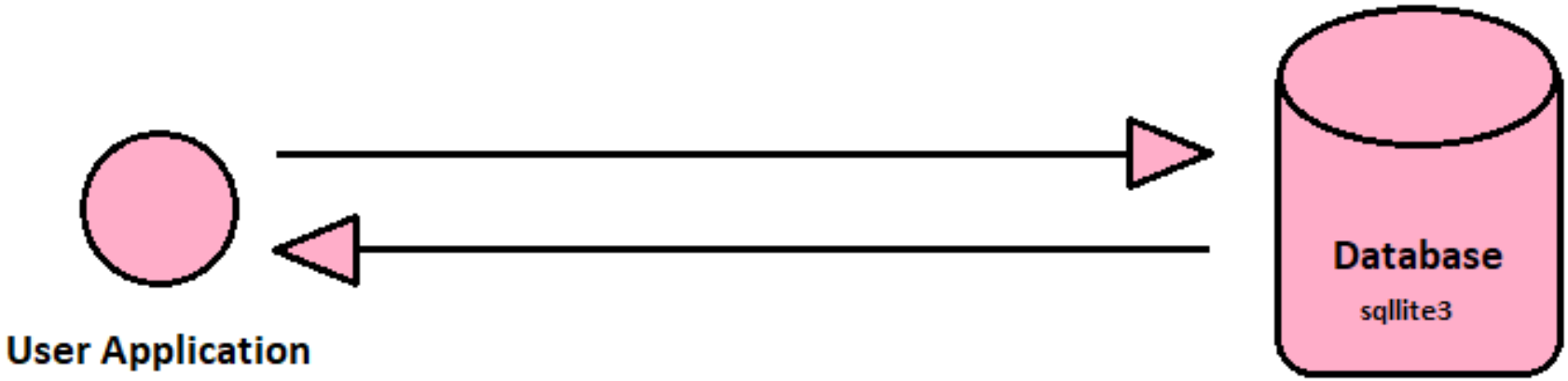
Socket Connection: Code Snippet [Client Machine]

```
import socket

HOST = socket.gethostname()
PORT = 10002 # Use the same port on the same machine.

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))

s.send(b"Hello World")
data = s.recv(1024)
s.close()
print("Received: ", repr(data))
```

Connection-less Service



Databases in Python



- There are many different databases that one can use e.g. MS Access, Microsoft Sequel Server, MySQL.
- For this course, we will use a lightweight database called SQLite3.
- We can now create and access databases.

We can import the SQLite3 library as follows:

```
import sqlite3
```

How to connect?

Next, we use the inbuilt function connect() to open a connection between our python program and the SQLite database. Connect to an sqlite database as follows:

```
import sqlite3
mydb = sqlite3.connect('awesome.db')
# DB file should be in the same folder in this case.
```

If your database file is in some other folder, you can also use absolute paths like this:

```
mydb = sqlite3.connect('/home/diptesh/ninja.db')
```

Create a New Database

```
import sqlite3
from sqlite3 import Error

def create_connection(db_file):
    # function to create an SQLite database
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print(sqlite3.version)
    except Error as e:
        print(e)
    finally:
        if conn:
            conn.close()

create_connection(r"path to where you want the file")
```

Create a New Table and Insert Data

```
c = mydb.cursor()

# create a table of students
c.execute("""CREATE TABLE students (rollnumber int,
    firstname varchar(10),
    surname varchar(10),
    DOB date)
    """)

# insert data into this table
c.execute("""INSERT INTO students VALUES (1540,
    'diptesh','kanojia','2/11/1987')""")

# Make sure you are inserting data in the right order
# and with an appropriate data type.
```

Saving Data, Reading Data, and Closing the Connection

```
# Save changes using the commit() function
mydb.commit()

# Retrieve the data from "students" table
c.execute("""SELECT * FROM students""")
print(c.fetchall())
"""
The output should be like:
[(1540,'diptesh','kanojia','2/11/1987')]
"""

# Close the database connection
mydb.close()
```

References

- Python: Basics
- W3Schools: Python
- e-Yantra Homepage

Thank You!

Author: Diptesh Kanojia
Contributor: Prashant K. Sharma

Post your queries at: resources@e-yantra.org