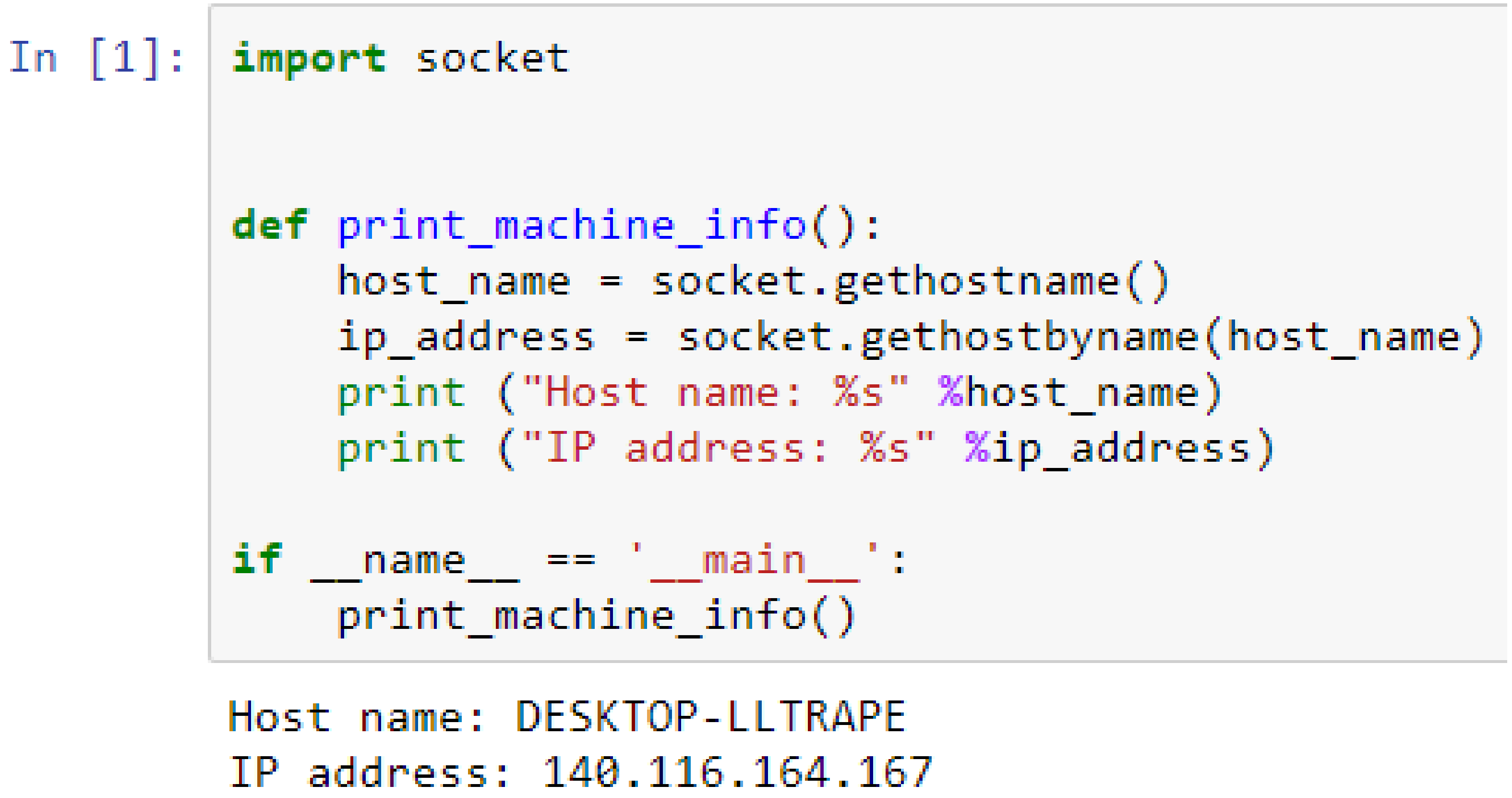
Computer Network Laboratory
Basic Network Programming

Jiawei Chang
Dept. of Computer Science and Information Engineering
National Taichung University of Science and Technology

Outline

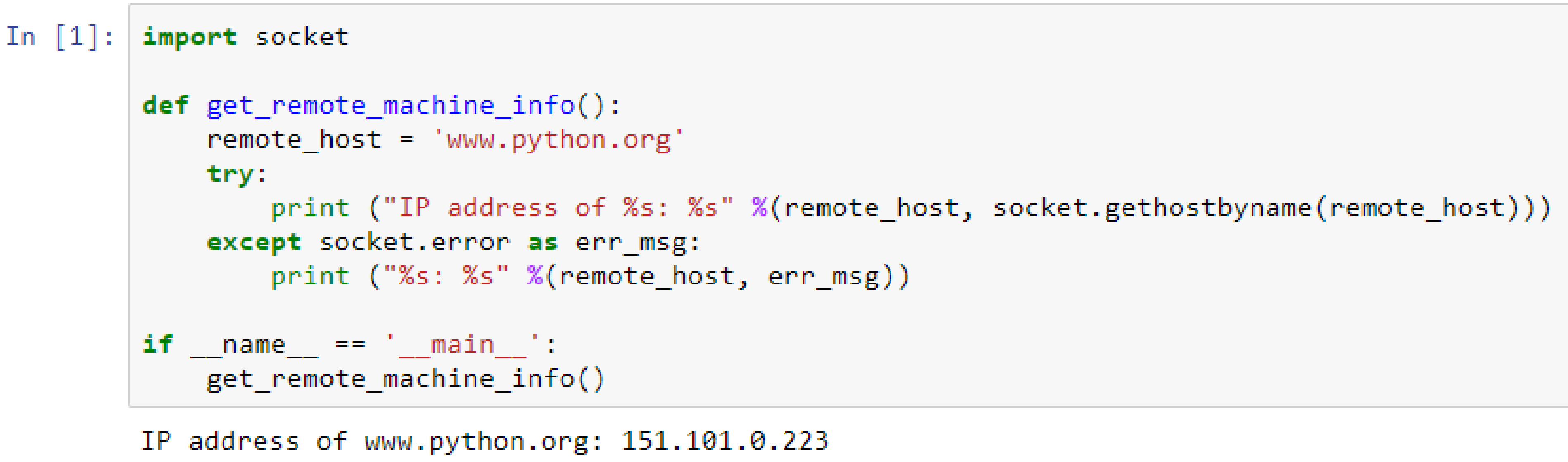
1. local_machine_info
2. remote_machine_info
3. ip4_address_conversion
4. finding_service_name
5. integer_conversion
6. socket_timeout
7. socket_errors
8. modify_buff_size
9. socket_modes
10. reuse_socket_address
11. print_machine_time
12. echo_server and echo_client by TCP
13. echo_server and echo_client by UDP

local_machine_info



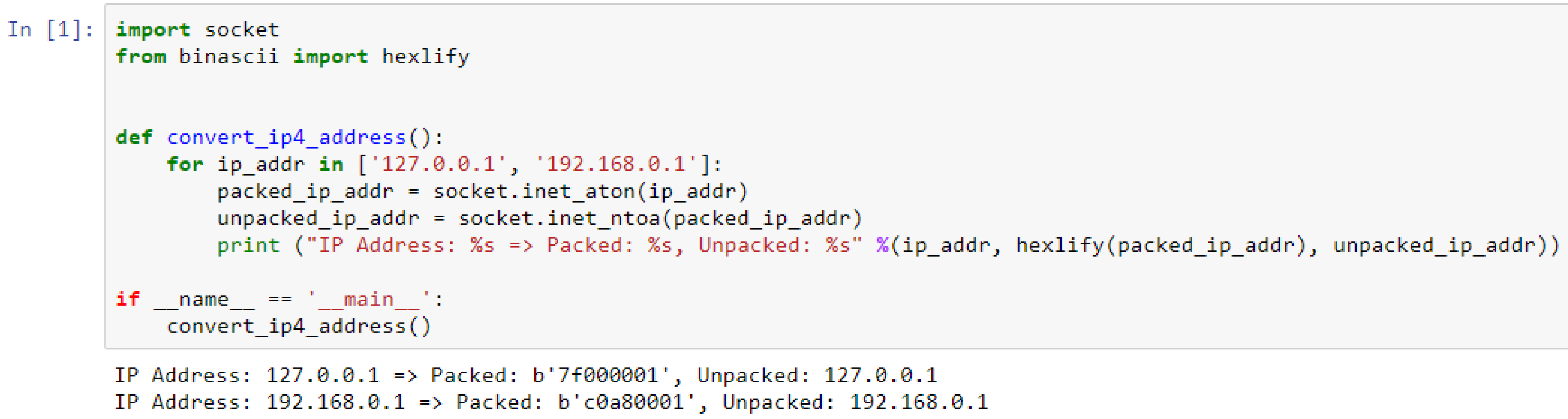
```
In [1]: import socket


        def print_machine_info():
            host_name = socket.gethostname()
            ip_address = socket.gethostbyname(host_name)
            print ("Host name: %s" %host_name)
            print ("IP address: %s" %ip_address)

        if __name__ == '__main__':
            print_machine_info()

        Host name: DESKTOP-LLTRAPE
        IP address: 140.116.164.167
```

remote_machine_info



```
In [1]: import socket

        def get_remote_machine_info():
            remote_host = 'www.python.org'
            try:
                print ("IP address of %s: %s" %(remote_host, socket.gethostbyname(remote_host)))
            except socket.error as err_msg:
                print ("%s: %s" %(remote_host, err_msg))

        if __name__ == '__main__':
            get_remote_machine_info()

        IP address of www.python.org: 151.101.0.223
```

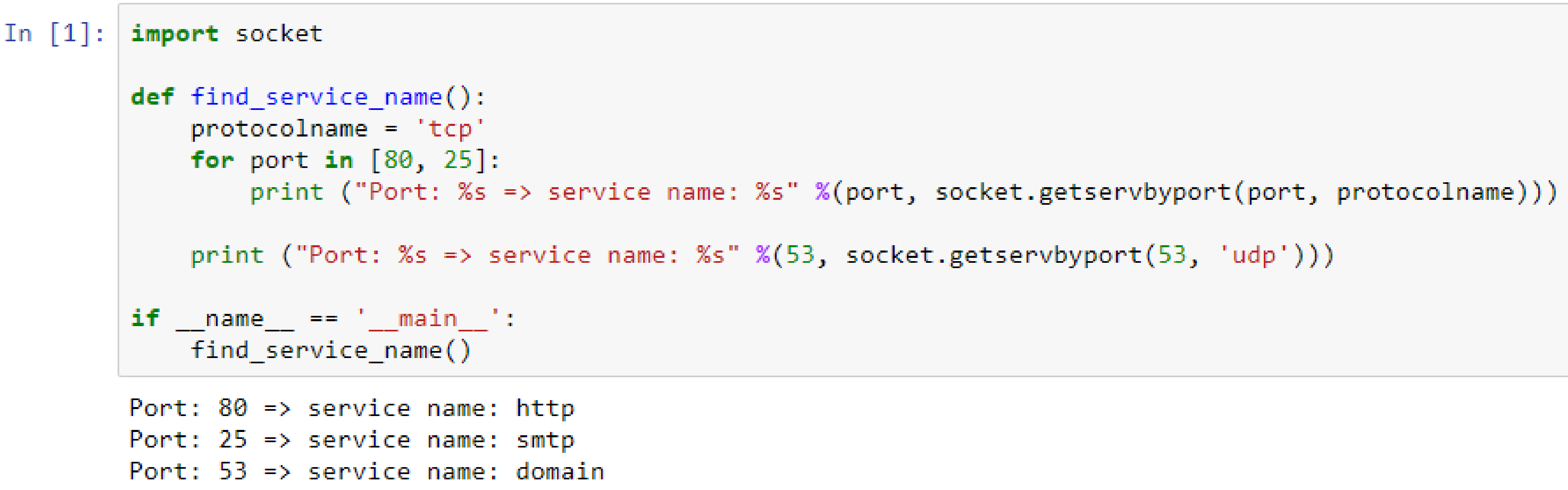
ip4_address_conversion



```
In [1]: import socket
        from binascii import hexlify


        def convert_ip4_address():
            for ip_addr in ['127.0.0.1', '192.168.0.1']:
                packed_ip_addr = socket.inet_aton(ip_addr)
                unpacked_ip_addr = socket.inet_ntoa(packed_ip_addr)
                print ("IP Address: %s => Packed: %s, Unpacked: %s" %(ip_addr, hexlify(packed_ip_addr), unpacked_ip_addr))

        if __name__ == '__main__':
            convert_ip4_address()
```

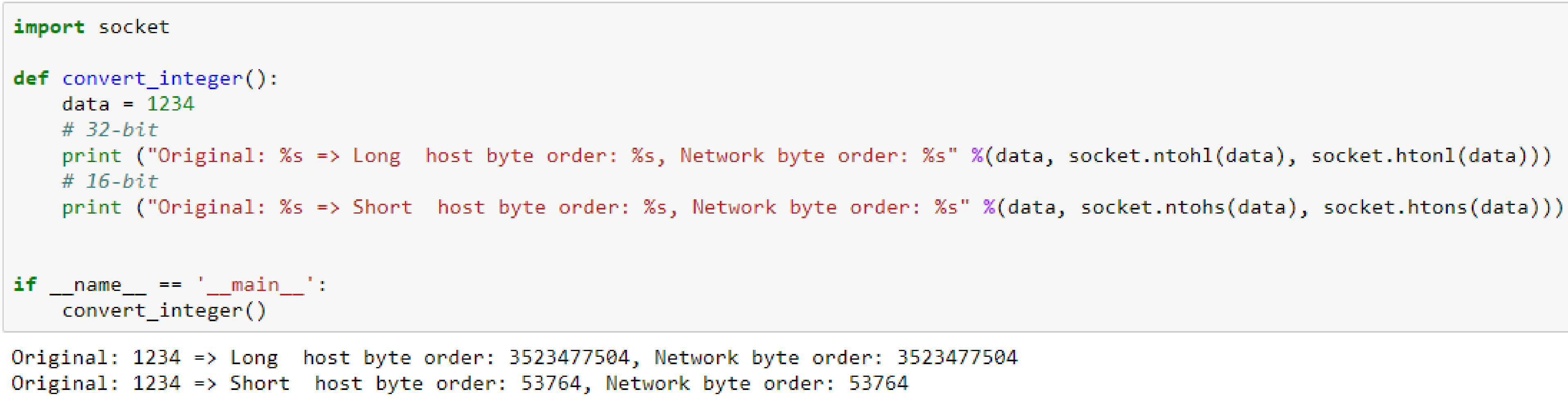
```
IP Address: 127.0.0.1 => Packed: b'7f000001', Unpacked: 127.0.0.1
IP Address: 192.168.0.1 => Packed: b'c0a80001', Unpacked: 192.168.0.1
```

finding_service_name



```
In [1]: import socket

        def find_service_name():
            protocolname = 'tcp'
            for port in [80, 25]:
                print ("Port: %s => service name: %s" %(port, socket.getservbyport(port, protocolname))

            print ("Port: %s => service name: %s" %(53, socket.getservbyport(53, 'udp')))

        if __name__ == '__main__':
            find_service_name()

        Port: 80 => service name: http
        Port: 25 => service name: smtp
        Port: 53 => service name: domain
```

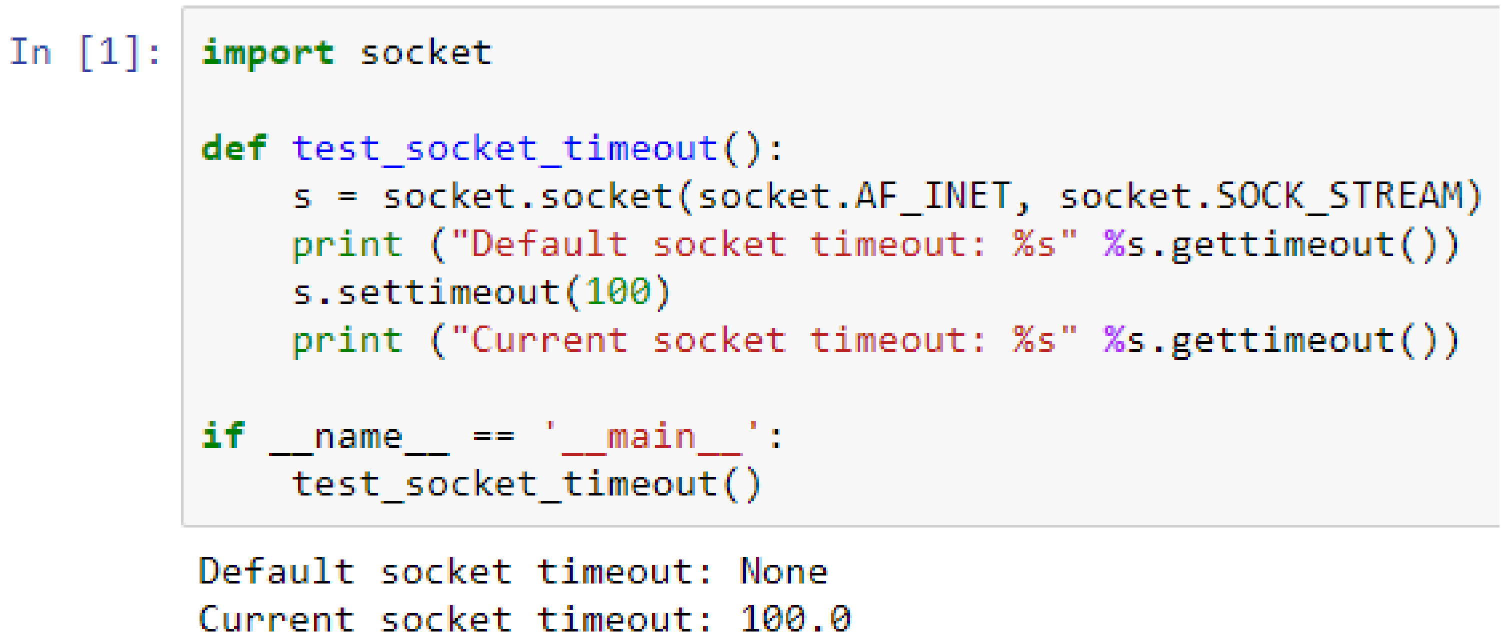
integer_conversion



```
import socket

def convert_integer():
    data = 1234
    # 32-bit
    print ("Original: %s => Long  host byte order: %s, Network byte order: %s" %(data, socket.ntohl(data), socket.htonl(data)))
    # 16-bit
    print ("Original: %s => Short  host byte order: %s, Network byte order: %s" %(data, socket.ntohs(data), socket.htons(data))


if __name__ == '__main__':
    convert_integer()
```

```
Original: 1234 => Long  host byte order: 3523477504, Network byte order: 3523477504
Original: 1234 => Short  host byte order: 53764, Network byte order: 53764
```

socket_timeout



```
In [1]: import socket

        def test_socket_timeout():
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            print ("Default socket timeout: %s" %s.gettimeout())
            s.settimeout(100)
            print ("Current socket timeout: %s" %s.gettimeout())

        if __name__ == '__main__':
            test_socket_timeout()
```

```
Default socket timeout: None
Current socket timeout: 100.0
```

socket_errors (I)

```python
import sys
import socket
import argparse


def main():
    # setup argument parsing
    parser = argparse.ArgumentParser(description='Socket Error Examples')
    parser.add_argument('--host', action="store", dest="host", required=False)
    parser.add_argument('--port', action="store", dest="port", type=int, required=False)
    parser.add_argument('--file', action="store", dest="file", required=False)
    given_args = parser.parse_args()
    host = given_args.host
    port = given_args.port
    filename = given_args.file
```

```python
    # First try-except block -- create socket
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    except socket.error as e:
        print ("Error creating socket: %s" % e)
        sys.exit(1)
```

```python
    # Second try-except block -- connect to given host/port
    try:
        s.connect((host, port))
    except socket.gaierror as e:
        print ("Address-related error connecting to server: %s" % e)
        sys.exit(1)
    except socket.error as e:
        print ("Connection error: %s" % e)
        sys.exit(1)
```

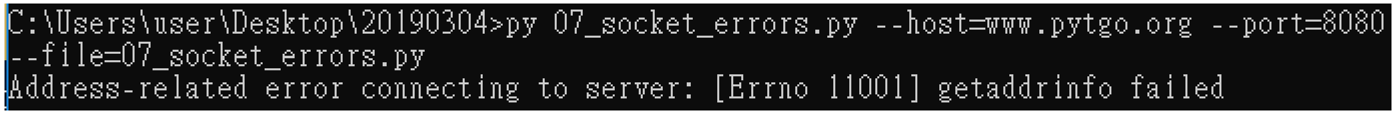
socket_errors (II)

```python
    # Third try-except block -- sending data
    try:
        msg = "GET %s HTTP/1.0\r\n\r\n" % filename
        s.sendall(msg.encode('utf-8'))
    except socket.error as e:
        print ("Error sending data: %s" % e)
        sys.exit(1)
```

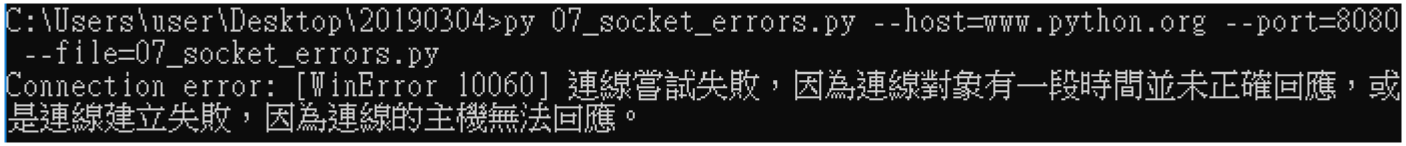
```python
    while 1:
        # Fourth tr-except block -- waiting to receive data from remote host
        try:
            buf = s.recv(2048)
        except socket.error as e:
            print ("Error receiving data: %s" % e)
            sys.exit(1)
        if not len(buf):
            break
        # write the received data
        sys.stdout.write(buf.decode('utf-8'))

if __name__ == '__main__':
    main()
```

socket_errors (III)

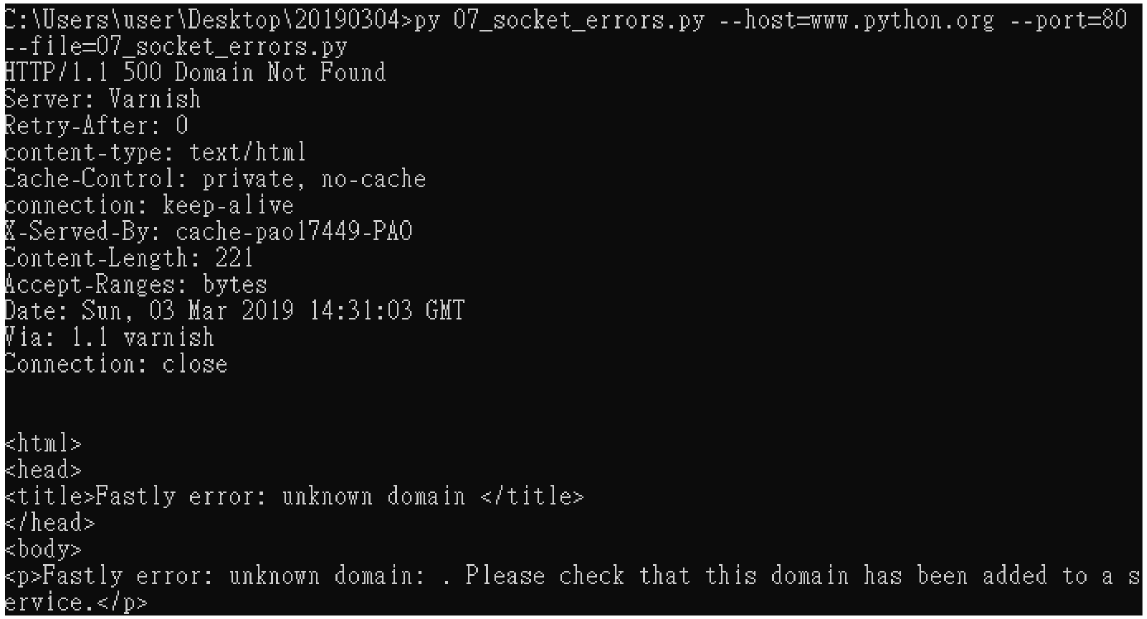
python 07_socket_errors.py --host=<HOST> --port=<PORT> --file=<FILE>



```
C:\Users\user\Desktop\20190304>py 07_socket_errors.py --host=www.pytgo.org --port=8080
--file=07_socket_errors.py
Address-related error connecting to server: [Errno 11001] getaddrinfo failed
```



```
C:\Users\user\Desktop\20190304>py 07_socket_errors.py --host=www.python.org --port=8080
 --file=07_socket_errors.py
Connection error: [WinError 10060] 連線嘗試失敗，因為連線對象有一段時間並未正確回應，或
是連線建立失敗，因為連線的主機無法回應。
```



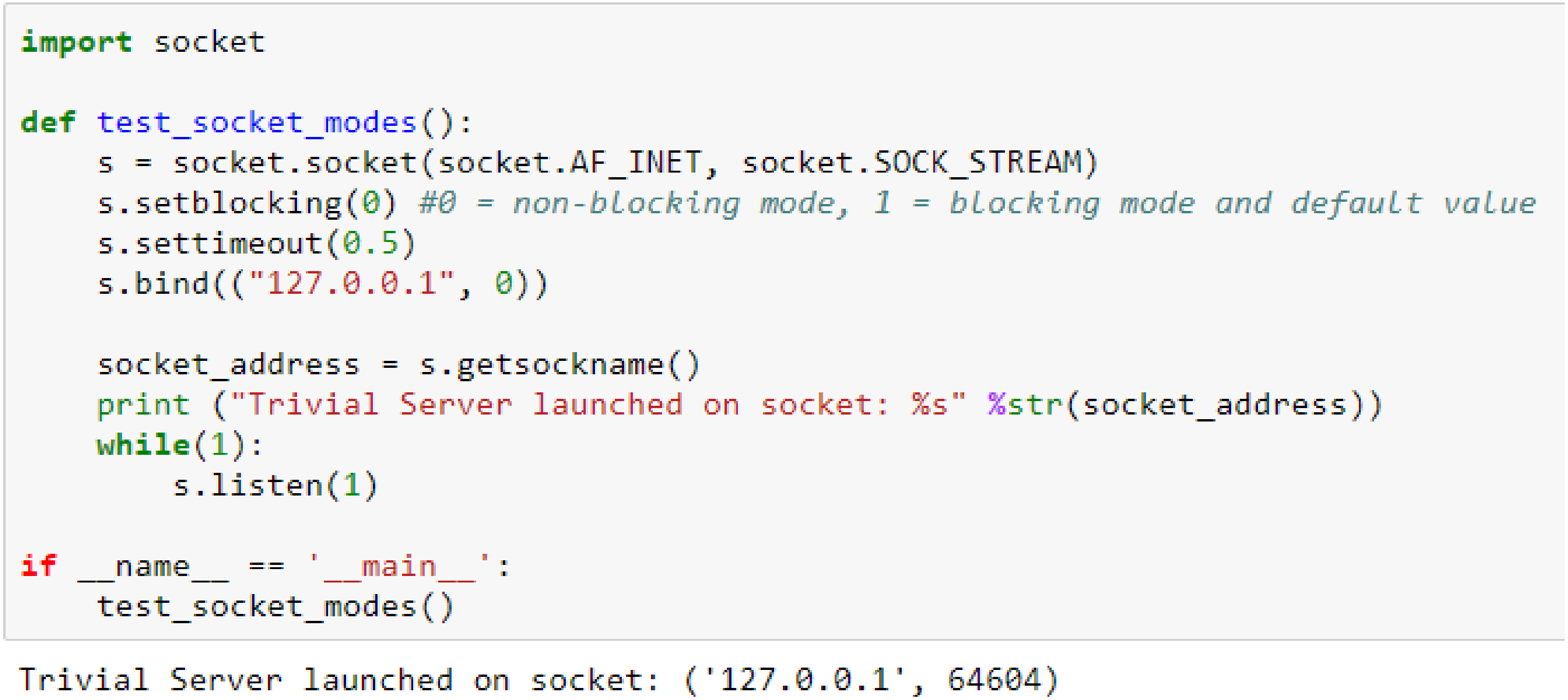
```
C:\Users\user\Desktop\20190304>py 07_socket_errors.py --host=www.python.org --port=80
--file=07_socket_errors.py
HTTP/1.1 500 Domain Not Found
Server: Varnish
Retry-After: 0
content-type: text/html
Cache-Control: private, no-cache
connection: keep-alive
X-Served-By: cache-pao17449-PAO
Content-Length: 221
Accept-Ranges: bytes
Date: Sun, 03 Mar 2019 14:31:03 GMT
Via: 1.1 varnish
Connection: close


<html>
<head>
<title>Fastly error: unknown domain </title>
</head>
<body>
<p>Fastly error: unknown domain: . Please check that this domain has been added to a s
ervice.</p>
```

modify_buff_size

```python
import socket

SEND_BUF_SIZE = 4096
RECV_BUF_SIZE = 4096

def modify_buff_size():
    sock = socket.socket( socket.AF_INET, socket.SOCK_STREAM )

    # Get the size of the socket's send buffer
    bufsize = sock.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
    print ("Buffer size [Before]:%d" %bufsize)

    sock.setsockopt(socket.SOL_TCP, socket.TCP_NODELAY, 1)
    sock.setsockopt(
            socket.SOL_SOCKET,
            socket.SO_SNDBUF,
            SEND_BUF_SIZE)
    sock.setsockopt(
            socket.SOL_SOCKET,
            socket.SO_RCVBUF,
            RECV_BUF_SIZE)
    bufsize = sock.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
    print ("Buffer size [After]:%d" %bufsize)

if __name__ == '__main__':
    modify_buff_size()
```

```
Buffer size [Before]:65536
Buffer size [After]:4096
```

socket_modes



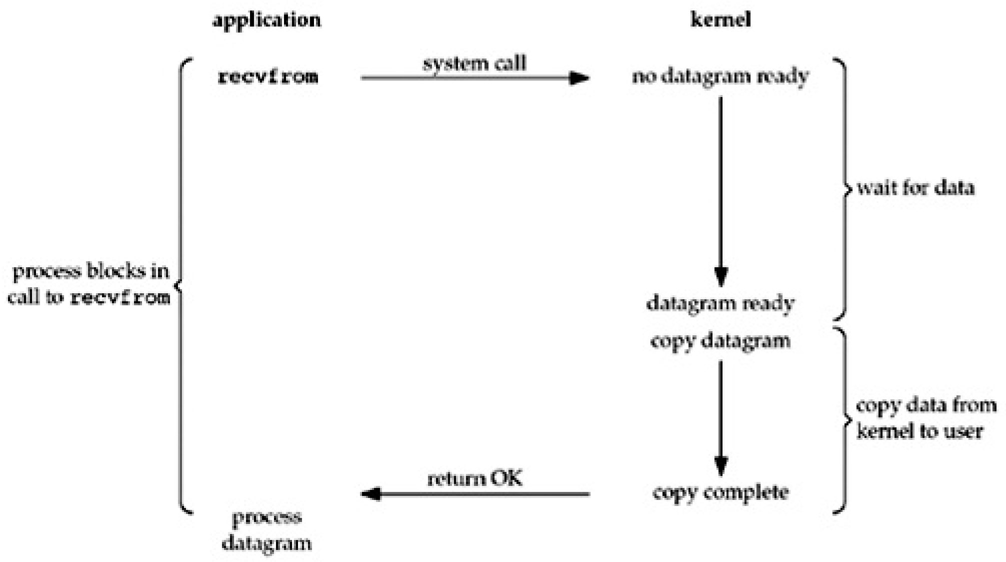
```python
import socket

def test_socket_modes():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setblocking(0) #0 = non-blocking mode, 1 = blocking mode and default value
    s.settimeout(0.5)
    s.bind(("127.0.0.1", 0))

    socket_address = s.getsockname()
    print ("Trivial Server launched on socket: %s" %str(socket_address))
    while(1):
        s.listen(1)

if __name__ == '__main__':
    test_socket_modes()
```

```
Trivial Server launched on socket: ('127.0.0.1', 64604)
```

Blocking and non-Blocking

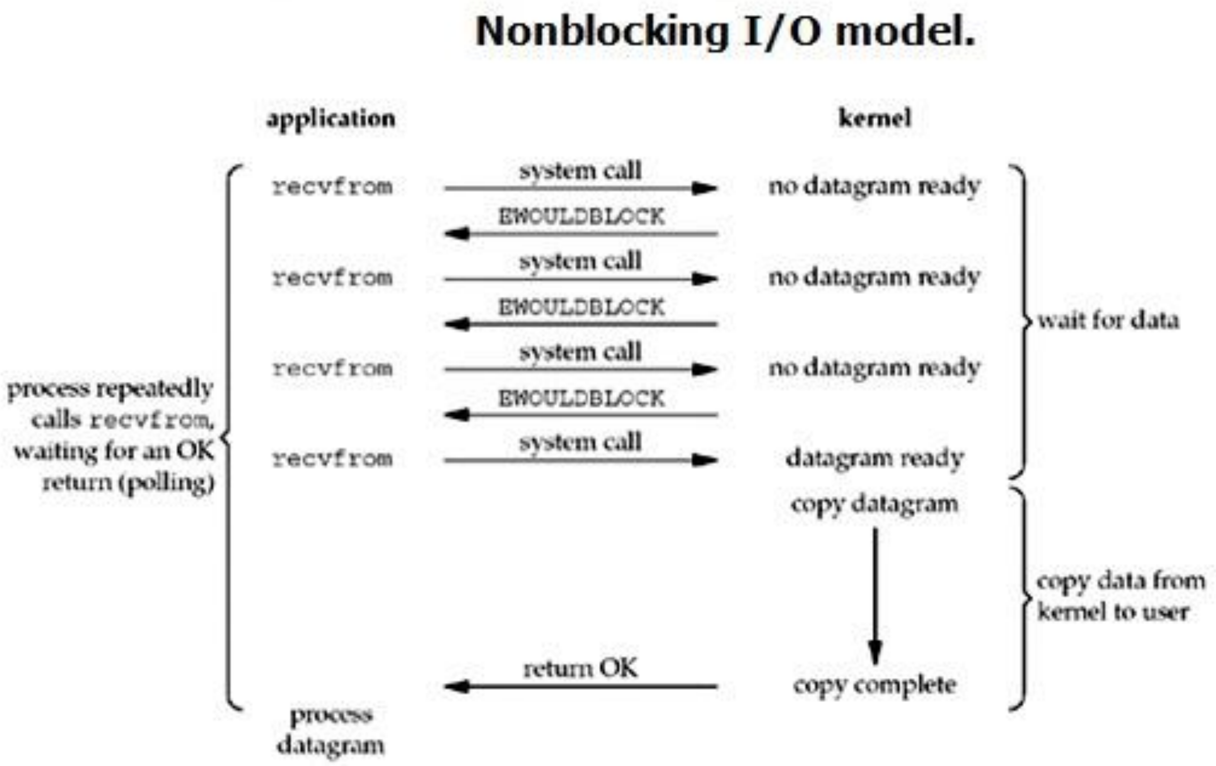
- 在預設的狀況，Server 通常是開啟 block 機制(同步)，將導致程式在執行時會被阻塞住，導致暫停執行。
- 非同步 I/O 的想法其實很單純，假如程式在執行過程中因為 I/O 暫停，但如果不會被阻塞住就能暫時把控制權切換給其它程式，這樣就不會浪費執行時間。

Blocking I/O model.

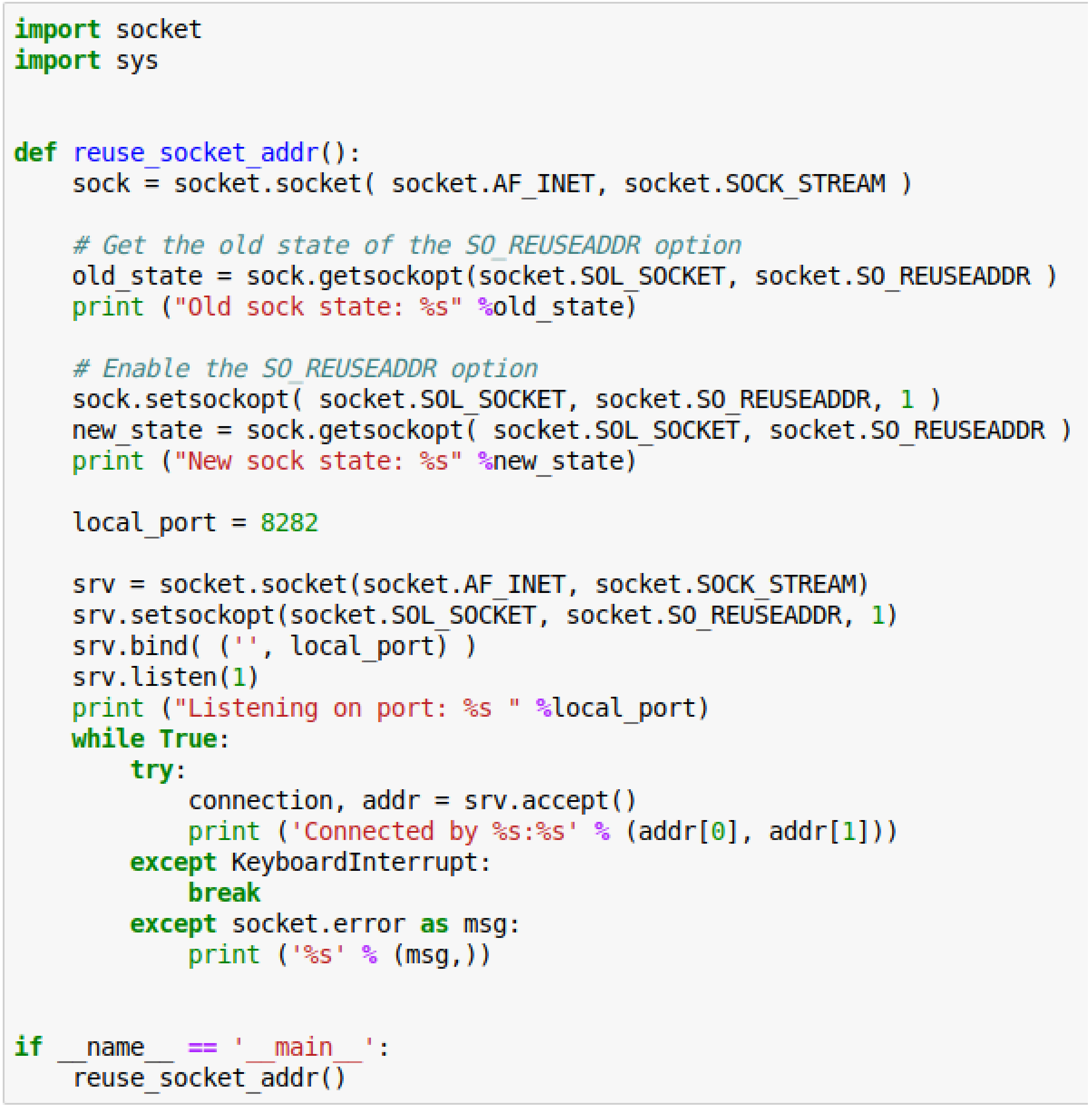


Blocking and non-Blocking

1. 當使用者程序發出read操作時，如果kernel中的資料還沒有準備好，那麼它並不會block使用者程序，而是立刻返回一個error。
2. 從使用者程序角度講 ，它發起一個read操作後，並不需要等待，而是馬上就得到了一個結果。
3. 使用者程序判斷結果是一個error時，它就知道資料還沒有準備好，於是它可以再次傳送read操作。
4. 一旦kernel中的資料準備好了，並且又再次收到了使用者程序的system call，那麼它馬上就將資料拷貝到了使用者記憶體，然後返回。
5. 所以，使用者程序其實是需要不斷的主動詢問kernel資料好了沒有。



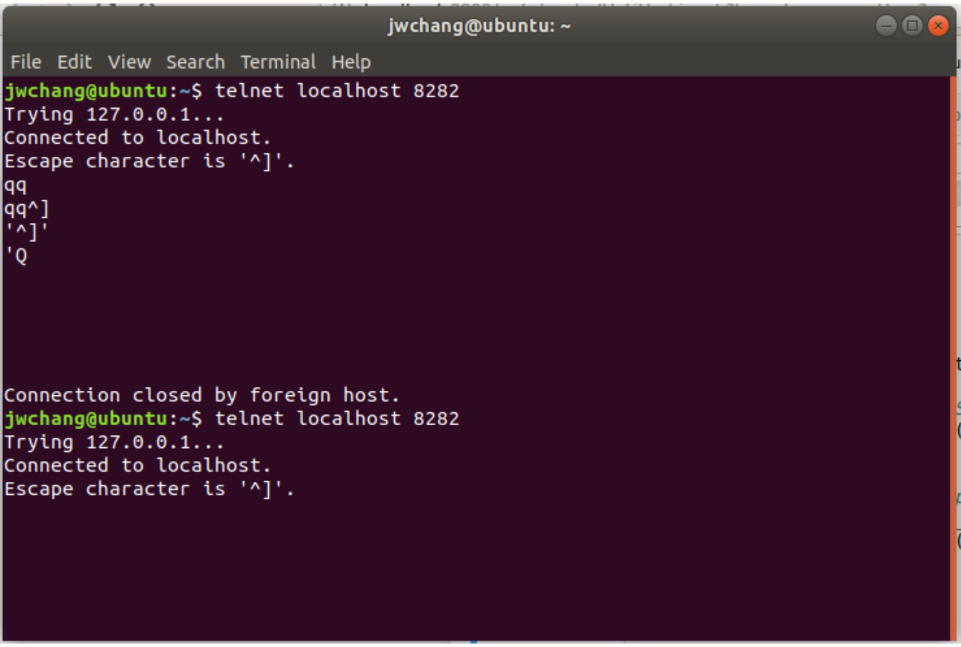
reuse_socket_address



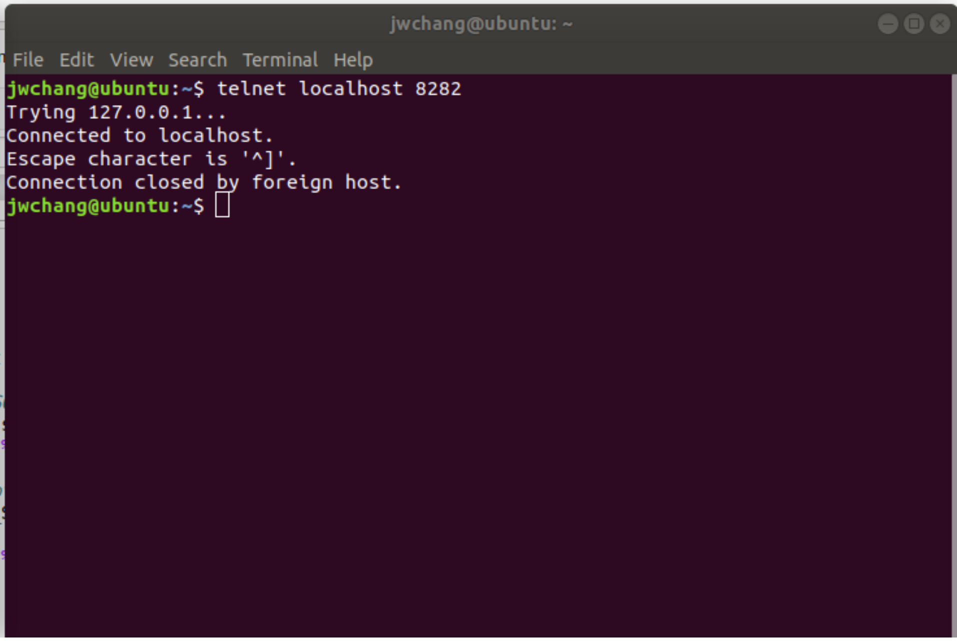
```python
import socket
import sys


def reuse_socket_addr():
    sock = socket.socket( socket.AF_INET, socket.SOCK_STREAM )

    # Get the old state of the SO_REUSEADDR option
    old_state = sock.getsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR )
    print ("Old sock state: %s" %old_state)

    # Enable the SO_REUSEADDR option
    sock.setsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR, 1 )
    new_state = sock.getsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR )
    print ("New sock state: %s" %new_state)

    local_port = 8282

    srv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    srv.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    srv.bind( ('', local_port) )
    srv.listen(1)
    print ("Listening on port: %s " %local_port)
    while True:
        try:
            connection, addr = srv.accept()
            print ('Connected by %s:%s' % (addr[0], addr[1]))
        except KeyboardInterrupt:
            break
        except socket.error as msg:
            print ('%s' % (msg,))


if __name__ == '__main__':
    reuse_socket_addr()
```

```
Old sock state: 0
New sock state: 1
Listening on port: 8282
Connected by 127.0.0.1:36272
Connected by 127.0.0.1:36406
Connected by 127.0.0.1:36412
```

reuse_socket_address

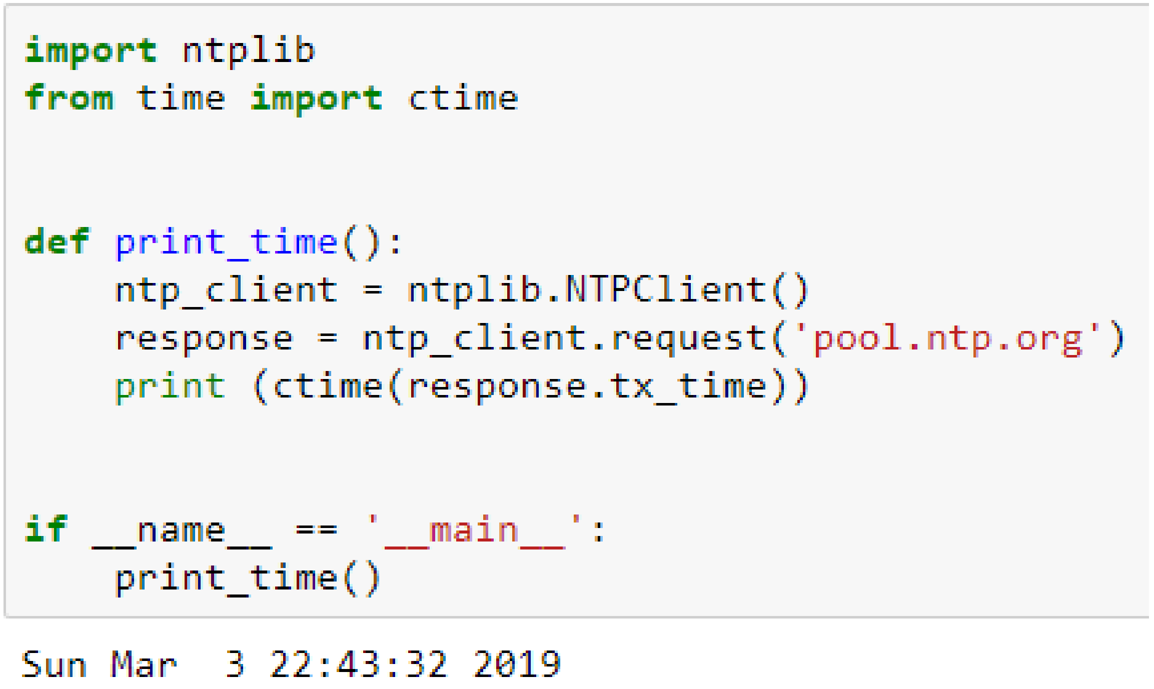


```
jwchang@ubuntu:~$ telnet localhost 8282
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
qq
qq^]
'^]'
'Q





Connection closed by foreign host.
jwchang@ubuntu:~$ telnet localhost 8282
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```



```
jwchang@ubuntu:~$ telnet localhost 8282
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Connection closed by foreign host.
jwchang@ubuntu:~$
```

print_machine_time



```
import ntplib
from time import ctime


def print_time():
    ntp_client = ntplib.NTPClient()
    response = ntp_client.request('pool.ntp.org')
    print (ctime(response.tx_time))


if __name__ == '__main__':
    print_time()
```

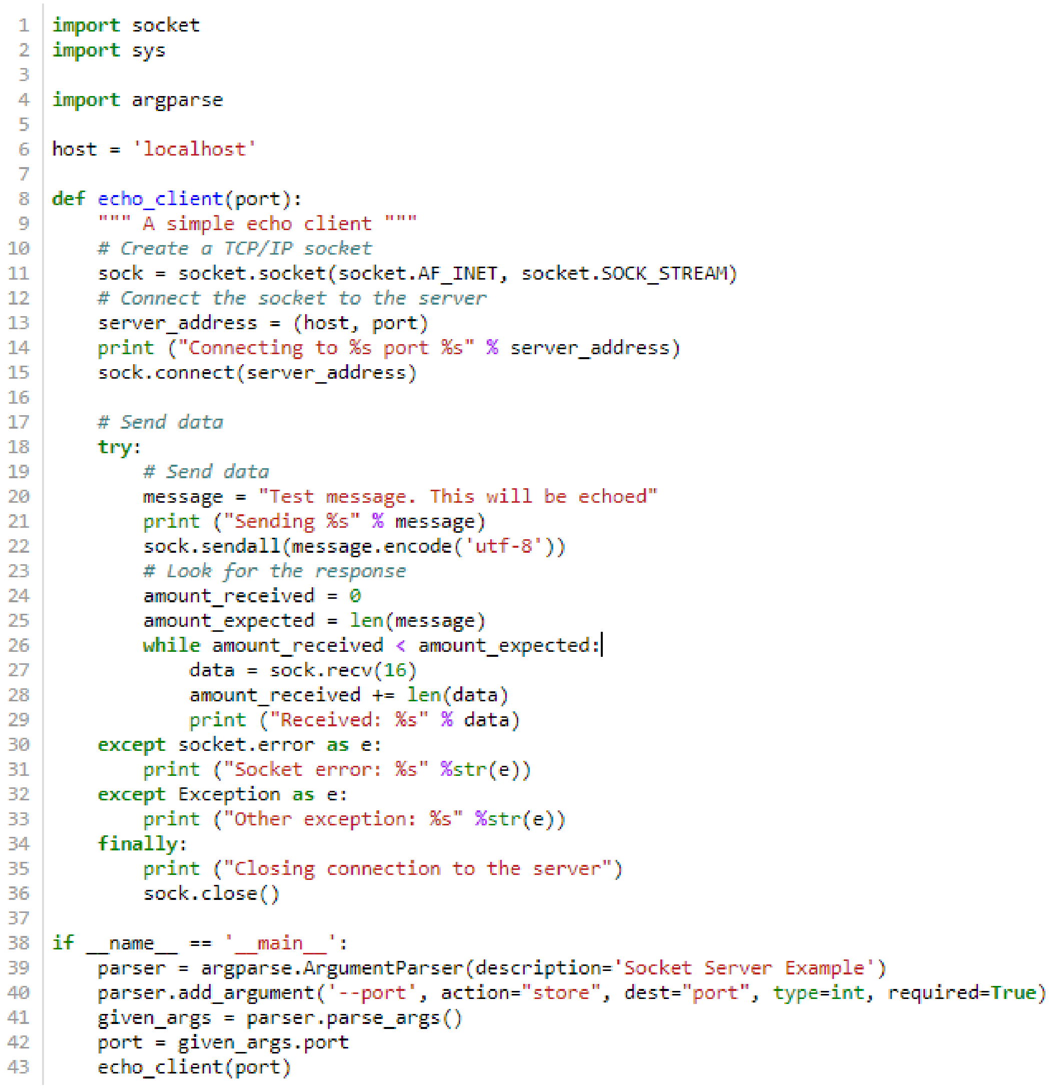
```
Sun Mar  3 22:43:32 2019
```

echo_server by TCP

```python
import socket
import sys
import argparse

host = 'localhost'
data_payload = 2048
backlog = 5


def echo_server(port):
    """ A simple echo server """
    # Create a TCP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Enable reuse address/port
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    # Bind the socket to the port
    server_address = (host, port)
    print ("Starting up echo server  on %s port %s" % server_address)
    sock.bind(server_address)
    # Listen to clients, backlog argument specifies the max no. of queued connections
    sock.listen(backlog)
    while True:
        print ("Waiting to receive message from client")
        client, address = sock.accept()
        data = client.recv(data_payload)
        if data:
            print ("Data: %s" %data)
            client.send(data)
            print ("sent %s bytes back to %s" % (data, address))
        # end connection
        client.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_server(port)
```

```
python 12_echo_server_TCP.py --port=<PORT>
```

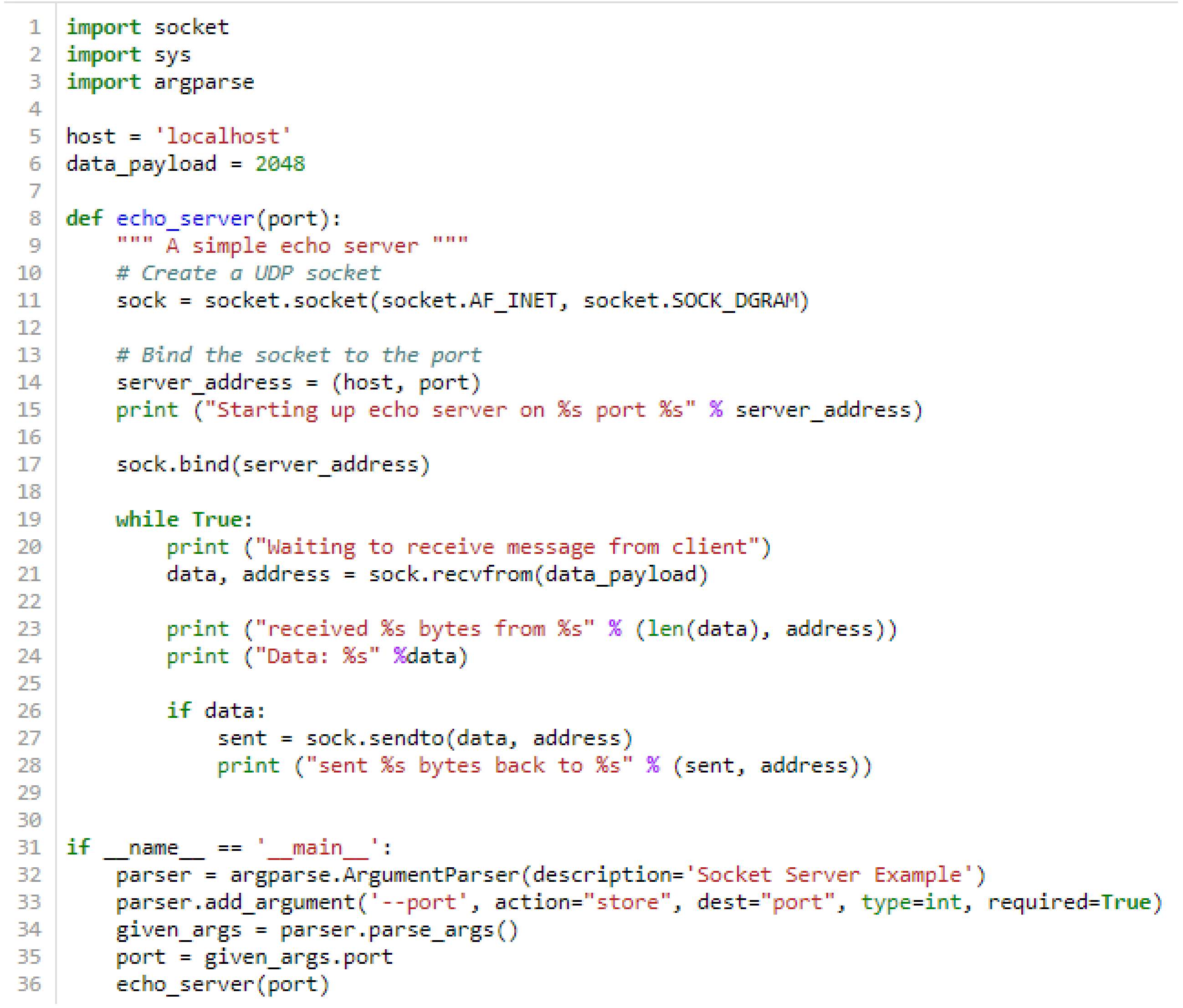
echo_client by TCP



```python
import socket
import sys

import argparse

host = 'localhost'

def echo_client(port):
    """ A simple echo client """
    # Create a TCP/IP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Connect the socket to the server
    server_address = (host, port)
    print ("Connecting to %s port %s" % server_address)
    sock.connect(server_address)

    # Send data
    try:
        # Send data
        message = "Test message. This will be echoed"
        print ("Sending %s" % message)
        sock.sendall(message.encode('utf-8'))
        # Look for the response
        amount_received = 0
        amount_expected = len(message)
        while amount_received < amount_expected:
            data = sock.recv(16)
            amount_received += len(data)
            print ("Received: %s" % data)
    except socket.error as e:
        print ("Socket error: %s" %str(e))
    except Exception as e:
        print ("Other exception: %s" %str(e))
    finally:
        print ("Closing connection to the server")
        sock.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_client(port)
```

```
python 12_echo_client_TCP.py --port=<PORT>
```

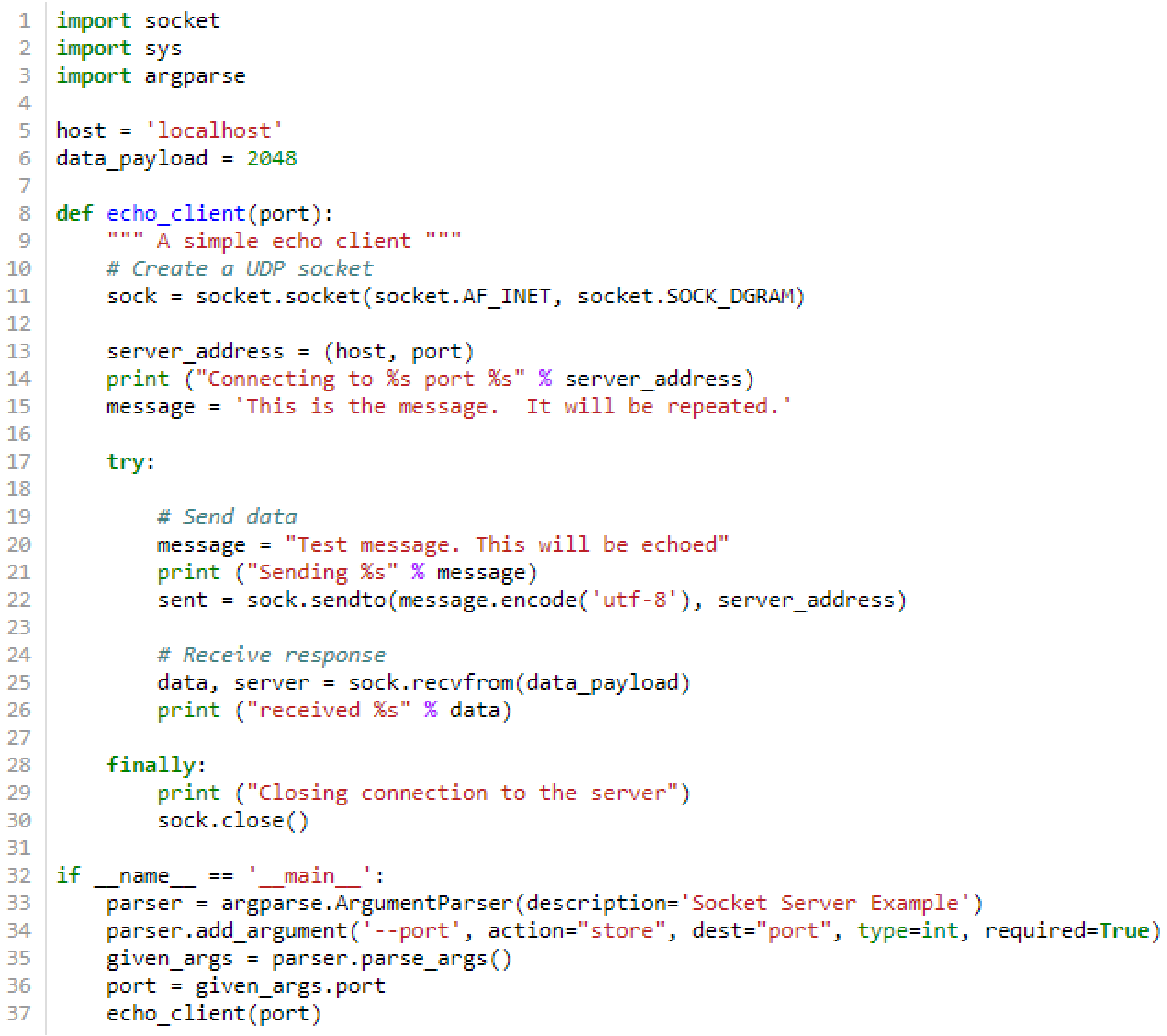
echo_server by UDP



```python
import socket
import sys
import argparse

host = 'localhost'
data_payload = 2048

def echo_server(port):
    """ A simple echo server """
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Bind the socket to the port
    server_address = (host, port)
    print ("Starting up echo server on %s port %s" % server_address)

    sock.bind(server_address)

    while True:
        print ("Waiting to receive message from client")
        data, address = sock.recvfrom(data_payload)

        print ("received %s bytes from %s" % (len(data), address))
        print ("Data: %s" %data)

        if data:
            sent = sock.sendto(data, address)
            print ("sent %s bytes back to %s" % (sent, address))


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_server(port)
```

python 13_echo_server_UDP.py --port=<PORT>

echo_client by UDP



```python
import socket
import sys
import argparse

host = 'localhost'
data_payload = 2048

def echo_client(port):
    """ A simple echo client """
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    server_address = (host, port)
    print ("Connecting to %s port %s" % server_address)
    message = 'This is the message.  It will be repeated.'

    try:

        # Send data
        message = "Test message. This will be echoed"
        print ("Sending %s" % message)
        sent = sock.sendto(message.encode('utf-8'), server_address)

        # Receive response
        data, server = sock.recvfrom(data_payload)
        print ("received %s" % data)

    finally:
        print ("Closing connection to the server")
        sock.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_client(port)
```

python 13_echo_client_UDP.py --port=<PORT>

延伸閱讀

- Socket Programming in Python (Guide)
  - https://realpython.com/python-sockets/#socket-api-overview
- Python 网络编程
  - http://www.runoob.com/python/python-socket.html

Resource is available by
https://jiaweichang.github.io/biography/

THANKS