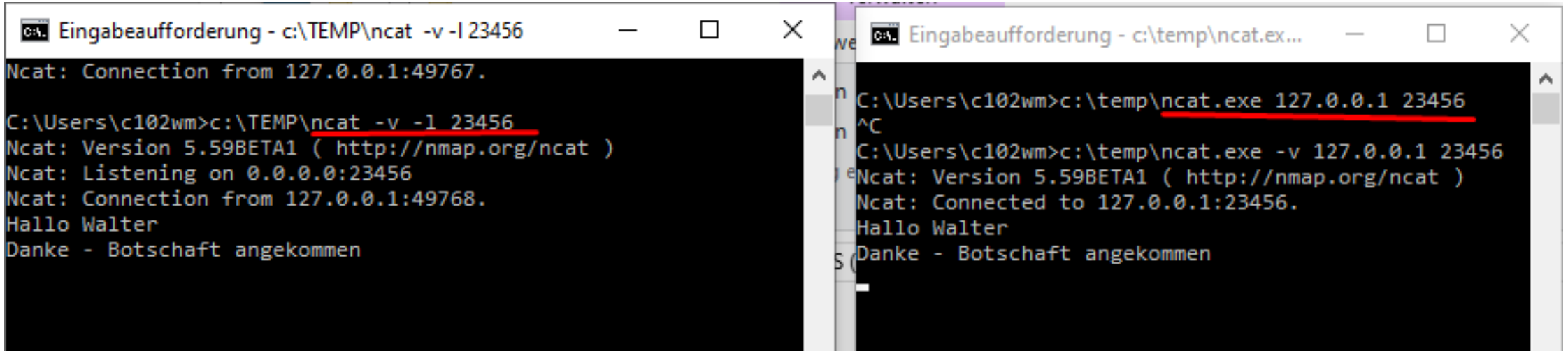
Python sockets/networking

IP-Networking zwischen Klienten und Servern



Python – socket Modul

- Sockets erlauben die Kommunikation
  - zwischen Prozessen auf einem Rechner aber auch
  - zwischen Anwendungen auf verschiedenen Rechnern via TCP/IP

- Doku:
  https://docs.python.org/3/library/socket.html

- Vorgansweise:
  - socket als Objekt anlegen
  - dann Verbindung aufbauen, oder am
  - socket lauschen und Verbindung annehmen

Python – networking client

```python
#!/usr/bin/python
# networking client see https://docs.python.org/3/library/socket.html

import socket               # Import socket module

host = "127.0.0.1"
port = 12345

# s = socket.socket()         # Create a socket object
# s.connect((host, port))     # connect to remote TCP service

# or more modern variant tries IPv4 and IPv6, multiple addresses
s=socket.create_connection(("localhost",port))

s.send("Hallo Walter".encode("UTF8"))   # write data
bytes=s.recv(1024)          # receive data
print("Response from server:",bytes.decode("UTF8"))      # decode and output received data
s.close()                   # Close the socket when done
```

Python – networking server

```
import sys
import socket  # Import socket module

s = socket.socket()  # Create a socket object
host = ''  # unspecified ip - all interfaces on host
port = 12345  # Reserve a port for your service.
s.bind((host, port))  # Bind to the port

s.listen(1)  # Now wait for client connection.
connection, client_address = s.accept()  # Establish / get one connection with client.

try:
    print('Got connection from %s' % str(client_address), file=sys.stderr)

    # Receive the data in small chunks and retransmit it
    while True:
        byte_data = connection.recv(30)     # read maximum of 30 bytes from connection
        data = byte_data.decode("latin1")   # convert byte data to string
        print('received "%s"' % data, file=sys.stderr)
        print('sending data back to the client', file=sys.stderr)
        connection.sendall(("server got: %s" % data).encode("latin1")) # send reply to connection
except:
    print("Unexpected error:", sys.exc_info()[0])
finally:
    connection.close()      # clean up the connection
    s.close()               # clean up listening socket
```

Python und Threads

```python
# Threads and Python
# see https://docs.python.org/3/library/threading.html#thread-objects

import threading
from time import sleep


def count_to(thread_name="default", count_to=10, sleep_time=1.0):
    for i in range(count_to):
        print("Thread %s: %d" % (thread_name, i))
        sleep(sleep_time)


for thread_number in range(3):
    t = threading.Thread(target=count_to, args=(str(thread_number), 10))  # a thread is an object
    t.start()  # start this thread
    sleep(2)  # wait before starting the next one

```

Python – multithreaded server

- Für jede hereinkommende Verbindung
  also erfolgreiches accept
- Wird
  ein connection handler als eigenständiger thread gestartet
- Das Programm kann dann mehrere Klienten gleichzeitig bedienen.