Übungsblatt 6

Abgabe als ZIP-Datei **Deadline**: 05.11.2020 23:55 Uhr

Ablauf der Übung

Deadline: : 05.11.2020 23:55 Uhr

- Die finalen Python Dateien sollen am E-Campus in der Übungsabgabe als ZIP Datei hochgeladen werden.
- Jedes Python Programm enthält in den ersten Zeilen einen Header in dem der Name, Autor, eine kurze Beschreibung enthalten sind.
- Der Code soll sauber geschrieben und wichtige Code-Zeilen kommentiert werden. Der Output sollte ähnlich wie in der Angabe aussehen!

```
"""
<Name>
<Description>
<Mat-Nr>
"""
```

U6Bsp1

Schreiben Sie einen Echo-Server. Dieser öffnet einen TCP-Socket auf localhost (127.0.0.1) Port 6666.

- Sobald sich ein Client verbindet, kann dieser Nachrichten an den Server schicken und wartet danach auf eine Antwort.
- Der Server wartet auf eine Client-Nachricht, wandelt diese in Großbuchstaben um und schickt den Inhalt an den Client zurück.

Wenn der Client den Befehl „quit“ schickt, wird zuerst die Nachricht übertragen, danach der Client und abschließend der Server sauber beendet.

- d.h. Befehl abfangen und auf beiden Seiten die Sockets schließen

Client Output:

```
Establish Connection...
Connected to 127.0.0.1:6666 is established
Connection Confirmed# IP: 127.0.0.1     Port: 49550

> Hallo
Server Echo: HALLO

> es
Server Echo: ES

> funktioniert, Super!
Server Echo: FUNKTIONIERT, SUPER!
```

Server Output:

```
Waiting for connection...
127.0.0.1:49550 is connected
('127.0.0.1', 49550): Hallo
('127.0.0.1', 49550): es
('127.0.0.1', 49550): funktioniert, Super!
```

Übungsblatt 6

Abgabe als ZIP-Datei **Deadline**: 05.11.2020 23:55 Uhr

U6Bsp2

Erweitern Sie den EchoServer und EchoClient aus Bsp U6Bsp1.
Entwickeln Sie einen Client und einen Server, die miteinander kommunizieren. Für diese Übung werden Sie beauftragt eine Client/Server Kommunikation zu implementieren und diese laut Aufgabenstellung zu erweitern.

Programmablauf

Der Client sendet eingegebene Nachrichten an den Server. Die Nachrichten werden mittels der Standardeingabe (stdin) eingegeben und sofort an den Server gesendet. Der Server zeigt jede empfangene Nachricht auf dem Bildschirm (Logging in eine Datei ebenfalls implementieren) an und leitet diese an alle anderen verbundenen Clients weiter. Jede Nachricht sollte vom Server mit der Absenderkennung vorangestellt werden, bevor sie weitergeleitet wird.

Schritt für Schritt Anleitung:

Für den Einstieg in die Übung könnte der Server aus einer Klasse mit folgenden Funktionen bestehen:

- *__init__* Initialisierung von Host, Port, etc.
- *bind* Anbindung der Sockets, etc.
- *send* Nachrichten versenden
- *receive* Nachrichten empfangen
- *broadcast* Nachrichten an alle verbunden Clients weiterleiten
- *stop* Schließt alle verbundenen Sockets

Die Client-Klasse könnte aus den Funktionen wie folgt bestehen:

- *__init__* Initialisierung von Host, Port, etc.
- *bind* Anbindung der Sockets, etc.
- *send* Nachrichten versenden
- *receive* Nachrichten empfangen
- *stop* *schließt den Client Socket*

1) Versuchen Sie zuerst die Grundfunktion zwischen Server und einem Client herzustellen. Wichtig dabei ist, dass der Server auf verschiedene Sockets hören muss - Threading erstellen und auf Interaktion warten.
    - Serveraufruf/parameter: `Usage: python3 U6Bsp2_Server.py`
    - Clientaufruf/parameter: `Usage: python3 U6Bsp2_Client.py <nickname>`

2) Sobald Nachrichten ausgetauscht werden, können Sie den Code für Verbindungen von mehreren Clients erweitern.
    a. Alle Clients sollen benachrichtigt werden sobald ein Client beitritt oder den Chat verlässt.
    b. Jeder Client soll mittels „*exit*“ den Chat verlassen können

c. Wenn ein Client „*list_user*" an den Server sendet, sollen ihm alle anderen verbundenen Clients mit der Anzahl der bereits gesendeten Nachrichten aufgelistet werden.
d. Jeder Client soll zu Beginn einen Nicknamen angeben (z.B.: Parameterübergabe oder Input-Abfrage) und diesen soll der Server zu den Socket Information dazu speichern (Bsp. Nickname als 1. Nachricht an Server senden; speichern als Dictionary, socket:nickname)

```
Usage: python3 Client.py <host> <port> <nickname>
```

3) Schreiben Sie alle Events (Nachrichten, neuer User, User beendet Verbindung, Schlüsselwörter wie list_user) die der Server empfängt als Info in eine Logging Datei server.log
   - Alle Fehlermeldung als Logging Error in dieselbe Datei

4) **[optional]** Implementierung Sie eine Verschlüsselung[1] der Kommunikation mittels AES-Algorithmus. Es soll davon ausgegangen werden, dass der gemeinsame Schlüssel beiden Kommunikationspartnern bekannt ist (Schlüsselaustausch optional implementierbar).
   a) Erweitern Sie den Code und fügen Sie die Funktionen *encrypt*, *decrypt* und *padding* hinzu.
      - *encrypt* Hier wird die zu versendete Nachricht verschlüsselt
      - *decrypt* Empfangene Nachrichten werden entschlüsselt
      - *padding* Informieren Sie sich über AES padding

5) **[optional]** Programmieren Sie ein GUI[2] für die verbunden Clients. Bsp.: ein Fenster mit Meldungen des Servers bzw. Nachrichten anderer Clients und einen Eingabebereich für selbstgeschriebenen Nachrichten ist zu erstellen.
   Ab Python3.6 bietet sich das python3.6-tk[3] Modul an:
   - *sudo apt-get install python3.6-tk*
   - *import tkinder*

[1] Empfehlenswert ist das Modul pycrypto oder pycryptodome
[2] Kapitel 1-4
[3] https://docs.python.org/3/library/tk.html

Übungsblatt 6

Abgabe als ZIP-Datei **Deadline**: 05.11.2020 23:55 Uhr

Möglicher Client Output:

```
Connected to 127.0.0.1:5555 is established
>ich grüße euch ;)
max: ICH SENDE
max: VIELE
max: NACHRICHTEN
max: HAHAHA
fredi: GENAU

>list_user
('127.0.0.1', 49662) - oliver - Anzahl d. Nachricht: 1
('127.0.0.1', 49673) - max - Anzahl d. Nachricht: 5
('127.0.0.1', 49675) - fredi - Anzahl d. Nachricht: 2

('127.0.0.1', 49673)- max hat den Chat verlassen
('127.0.0.1', 49675)- fredi hat den Chat verlassen

>quit
Server is Down. You are now Disconnected. Press enter to exit...
```

Möglicher Server Output bzw. Logdatei:

```
Server is running
Waiting for connection...
NEW USER: [('127.0.0.1', 49662), 'oliver', 0]
[('127.0.0.1', 49662), 'oliver', 1]: hello
NEW USER: [('127.0.0.1', 49673), 'max', 0]
[('127.0.0.1', 49673), 'max', 1]: servus
NEW USER: [('127.0.0.1', 49675), 'fredi', 0]
[('127.0.0.1', 49675), 'fredi', 1]: was los mit euch?
NEW USER: [('127.0.0.1', 49678), 'sisi', 0]
[('127.0.0.1', 49678), 'sisi', 1]: ich grüße euch ;)
[('127.0.0.1', 49673), 'max', 2]: ich sende
[('127.0.0.1', 49673), 'max', 3]: viele
[('127.0.0.1', 49673), 'max', 4]: Nachrichten
[('127.0.0.1', 49673), 'max', 5]: hahaha
[('127.0.0.1', 49675), 'fredi', 2]: genau
LEFT CHAT: [('127.0.0.1', 49673), 'max', 3]
LEFT CHAT: [('127.0.0.1', 49675), 'fredi', 2]:
```