

HANGMAN



* This project is made in Visual Studio 2019, so if you want to see the code, you can either open the .py files, or the .sln files

The project contains a pair of client – server applications (made with Python sockets) that implements the game „Hangman”. The rules are simple:

- the client, the moment he/she wants to start the game, sends the start command to the server;
- the server chooses a random word from the 'hangwords.txt' file. It sends a version of this word to the client. This version retains only the letters that appear at the beginning and at its end, the rest of the letters being replaced by underlines. Along with this, the server also sends a list of letters from the word HANGMAN which shows how many times the client has made a mistake and how many guesses it still has; at first there is no letter from HANGMAN because there's no mistake ... yet 😊.
- after the client receives the sketch of the word he/she has to guess, he/she will send letters to the server till the game finishes. If the letter received from the client exists in the

word that needs to be guessed, then the underline from the position of that letter in the word is replaced by that letter changing the version of the word that is sent to the client. Otherwise, a letter from the word 'HANGMAN' is added to the list. The current version of the word to be guessed along with the letters from HANGMAN are added to the message sent to the client to show him/her the progress.

Too complicated?

Let's check the code and see how it works.

Maybe things are getting better . . .

Let's start with the client, more precisely, with the beginning of the Client program:

```
import socket

host = '127.0.0.1'
port = 6200

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

At the beginning of the program, the 'socket' library is imported so that we can use the sockets, although this is completely obvious 😊.

Then we choose the IP address and port number of the server to be able to connect the applications. The address 127.0.0.1 was chosen because both applications were created on the same device. The port 6200 was randomly chosen for the sake of the project.

'client_socket' is an object that inherits the attributes of the 'socket' class by calling the class constructor. It helps us send the messages to the server.

After that we declare 3 variables. Their role is explained in the comments from the code:

```
finished = False  #'finished' checks if the game is over, if the word was found or if the word HANGMAN was completed.  
st = False  #'st' is a variable that checks if the start command has been sent to the server  
start = ''  #'start' is a string that gets the start message input
```

Now we are getting to the most important part from the Client application: we'll see how to send messages to the server and how to get the word we need to guess from the server. The message exchange takes place in a while structure. The instructions in the structure repeat as long as the boolean variable 'finished' (which checks if the game is over) is False.

```
while finished == False:  
  
    # It is checked if the "START" message has been sent  
    if st == False:  
  
        while start.upper() != "START":  
            start = input("Input the start command(START): ")
```

First we check if the start command has been sent (it must be sent so the game begins). The boolean variable 'st' checks this. 'st' is currently False.

We have to write the "START" command. The size of the characters is ignored because it is automatically converted to uppercase and then compared to "START".

```
while start.upper() != "START":
```

As long as the right command is not entered, the input will continue to be requested so that the application can move on.

```
while start.upper() != "START":  
    start = input("Input the start command(START): ")
```

After writing "START" and send it to the server, the client receives the structure of the word that needs to be guessed, and 'st' becomes True (the command has been sent).

```
while start.upper() != "START":  
    start = input("Input the start command(START): ")  
  
# The message "START" is sent to the server, then the word to be guessed is received and displayed.  
start = start.upper()  
client_socket.sendto(start.encode(), (host, port))  
message, serverAdress = client_socket.recvfrom(100)  
print(message.decode())  
st = True
```

The word that has to be guessed;
Each letter that is different from the first
and the last one is replaced with an underline

Here are the
letters that we
have to guess;
this part will be
explained in more
detail below

```
Input the start command(START): Something wrong
Input the start command(START): Something definitely wrong
Input the start command(START): sTaRt
[I__E__I_E][]
Type a letter:
```

This command
is case insensitive

Let's get back to this line: `client_socket.sendto(start.encode(), (host, port))`

Using the `client_socket` object, we send the encoded message to the server that has the address and the port from the tuple.

Now let's go to the server and see what happens when the message is received.

But before that, we should take a look at the beginning of the server program:

```
import socket
from random import randint

port = 6200

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('', port))

print("The server is ready!")

hangman = ['H', 'A', 'N', 'G', 'M', 'A', 'N']
```

We give the port its value (6200 is the same value that the port in the Client has).

Then 'server_socket' inherits the attributes of the 'socket' class by calling the class constructor. On the next line, the server binds with the client that uses the port 6200.

We then print a message to show that the server is ready to do its job.

'hangman' is a list that contains the letters of the word "HANGMAN". The letters from this list will be added to the message that will be sent to the client each time the client make a wrong guess.

The exchange of messages and the verification of the letters take place in a 'while' structure, but this time there is no condition for this 'while' to end, because the server must always be ready in order to be accessed by a new Client.

In the first instruction inside of 'while' a message and the Client's address are received from the Client. The message is verified. The first time it is checked if it is "START" and indeed, in the Client we sent the command "START" to the Server.

```
while True:

    # The message is received from the client
    message, clientAddress = server_socket.recvfrom(100)

    # Checking if the received message is the "START" command
    if message.decode() == "START":
```

After receiving the start command, the server opens the 'hangwords.txt' file and reads the words in it (each word is on a different line), and then chooses a random number, less than the number of lines in the text file , and the word at that position becomes the word that needs to be guessed by the Client.

```
if message.decode() == "START":

    f = open('hangwords.txt')
    words = f.readlines()
    w = len(words)
    k = randint(0, w - 1)

    ln = len(words[k])
    word = words[k]
```

Then two lists are declared (the first one - 'guessWord', contains the letters from the word that needs to be guessed, and the second one - 'H', contains letters from the hangman, letters that are added to the list at each mistake of the Client) and a string - 'msg', whose content will consist of the two lists combined. The two lists are currently empty, they only have square brackets because between them will be included the word to be guessed and hangman.

```
guessWord = ['', ''] #The list of letters of the word to be guessed.  
H = ['', ''] #The list in which letters from the word HANGMAN are added after every mistake  
msg = '' #The message that will be sent to the client; it is composed of guessWord and H
```

The version of the word to be guessed is formed (the letters that are equal to the first and last letter of the word are added in 'guessWord', and the others are first replaced with underline, and then added in 'guessWord').

```
for i in range(1, ln):  
    if word[i - 1] == word[0]:  
        guessWord.insert(i, word[0])  
    elif word[i - 1] == word[ln - 2]:  
        guessWord.insert(i, word[ln - 2])  
    else:  
        guessWord.insert(i, '_')
```


'msg' is created by concatenating 'guessWord' and 'H' and is sent to the server:

The completed word is displayed on the server (this line is optional)

The server is ready!
The word is: IMPRESSIVE

```
print("The server is ready!")
```

```
#Creating the message that will be sent to the client
for i in range(len(guessWord)):
    msg = msg + guessWord[i]

for i in range(len(H)):
    msg = msg + H[i]

print('The word is:', word)
server_socket.sendto(msg.encode(), clientAddress) #The message is sent to the client

j = 0 #The counter that shows which letter in HANGMAN has been reached
```

'j' is a counter that adds letters from 'H' to the message sent to the client. It becomes 0 after each START command because each client starts the game with 0 mistakes.

The message is sent to the Client so let's get back to the client program.

```
message, serverAddress = client_socket.recvfrom(100)
print(message.decode())
st = True
```

The message is received and displayed and 'st' becomes True.

[I__E__I_E][]

This is the message sent by the Server in this example. The second list is empty because there is no mistake made, so there is no letter from the list 'H'.

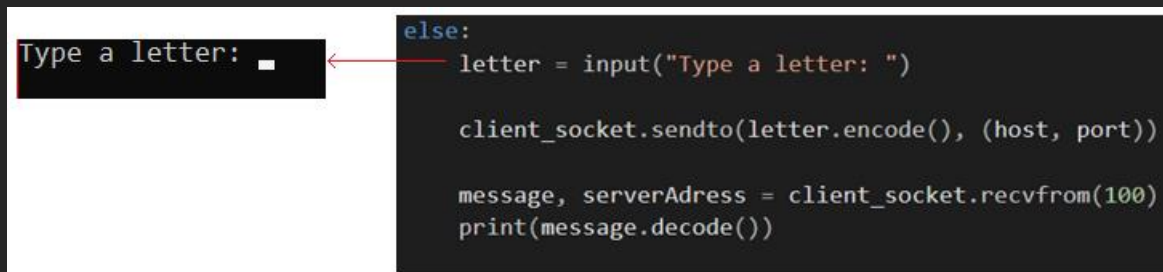
The 'while' structure is repeated again, only this time "START" has been sent, so we jump to the "else" statement:

```
while finished == False:

    # It is checked if the "START" message has been sent
    if st == False: ...

    #If the START command has been sent, the client must enter letters that the application sends to the server, and then the response is displayed.
    else:
```

Inside 'else' a letter is first required to be inserted. The letter is then sent to the Server, and then a reply is received which is displayed.



```
else:
    letter = input("Type a letter: ")

    client_socket.sendto(letter.encode(), (host, port))

    message, serverAdress = client_socket.recvfrom(100)
    print(message.decode())
```

The program checks if there are underlines in the reply, or if the word "HANGMAN" appears in the reply. If one of the two conditions is True, the game ends; otherwise, 'while' is repeated until one of the two conditions becomes True.

```
#If there are no more underlines in the word to be guessed, or if the word HANGMAN is complete, then the game finishes
if not ('_' in message.decode()):
    print("\nCongrats! You've guessed the word! You win!\n")
    finished = True

elif "[HANGMAN]" in message.decode():
    print("\nSorry, but you ran out of guesses! Maybe next time...\n")
    finished = True
```

At the end of the program, after the game is over, the connection to the Server closes.

```
client_socket.close()
```

Let's get back to this line: `client_socket.sendto(letter.encode(), (host, port))`

Here a letter is sent to the Server. Let's see what does the Server do with the letter:

The 'while' structure is repeated again, only this time the message is no longer "START", so the program jumps to the 'else' statement.

```

while True:

    # The message is received from the client
    message, clientAddress = server_socket.recvfrom(100)

    # Checking if the received message is the "START" command
    if message.decode() == "START":
        ...

    #If the received message is not the start command it means that the game has started and the message received is a letter.
    else:

```

Inside 'else', 'msg' becomes empty because it will be formed this time from a new 'guessWord' or a new 'H', depending on how the Client guesses the letter.

```

else:
    msg = ''    # The old message is deleted so that a new one can be created

```

The program checks if the letter appears in the word. If it does, the underline of the letter position (s) in the word is replaced by the letter; if not, a character from 'hangman' is added to 'H', and 'j' increases, moving to the next character in 'hangman'

```

#The letter is received and it is checked if it appears in the word
letter = message.decode()
if letter.upper() in words[k]:
    for i in range(len(word)):
        if letter.upper() == word[i]:
            guessWord[i+1] = word[i]

else:
    H.insert(j+1, hangman[j])
    j = j + 1

```

After that, 'guessWord' and 'H' are added to 'msg', and then the message is sent to the Client.

```

#guessWord and H join in 'msg' and are send to the client
for i in range(len(guessWord)):
    msg = msg + guessWord[i]

for i in range(len(H)):
    msg = msg + H[i]

server_socket.sendto(msg.encode(), clientAdress)

```

This is how a complete message exchange looks like:

```
Input the start command(START): Something wrong
Input the start command(START): Something definitely wrong
Input the start command(START): sTaRt
[I__E__I_E][]
Type a letter: M
[IM__E__I_E][]
Type a letter: p
[IMP_E__I_E][]
Type a letter: q
[IMP_E__I_E][H]
Type a letter: RE
[IMP_E__I_E][H]
Type a letter: QQ
[IMP_E__I_E][HA]
Type a letter: R
[IMPRE__I_E][HA]
Type a letter: L
[IMPRE__I_E][HAN]
Type a letter: Z
[IMPRE__I_E][HANG]
Type a letter: S
[IMPRESSI_E][HANG]
Type a letter: o
[IMPRESSI_E][HANGM]
Type a letter: v
[IMPRESSIVE][HANGM]

Congrats! You've guessed the word! You win!
```

```
Input the start command(START): START
[S____E][ ]
Type a letter: P
[S____E][H]
Type a letter: U
[S____E][HA]
Type a letter: A
[S_A__E][HA]
Type a letter: X
[S_A__E][HAN]
Type a letter: H
[S_A__E][HANG]
Type a letter: N
[S__AN_E][HANG]
Type a letter: P
[S__AN_E][HANGM]
Type a letter: O
[S__AN_E][HANGMA]
Type a letter: R
[S__RAN_E][HANGMA]
Type a letter: L
[S__RAN_E][HANGMAN]

Sorry, but you ran out of guesses! Maybe next time...
```

The server is always ready and receives messages from multiple clients, generating new words after each START command.

```
The server is ready!
The word is: IMPRESSIVE

The word is: STRANGE
```