

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ST JOSEPH ENGINEERING COLLEGE, MANGALURU-28



Computer Networks(18CS52/17CS52/15CS52) Assignment
Report
2020-2021

Topic Name:	Guess the word	
Team No:	6	
Team Members:	USN	Name
	4SO18CS086	Novin Misquith
	4SO18CS088	Pranith Rao
	4SO18CS124	Welroy Jeevan Gonsalves
	4SO18CS125	Wenzel Audrin Barboza
Submission Date:	05/01/2021	
Total Marks Awarded:		
Staff Signature:		

COMPUTER NETWORKS ASSIGNMENT RUBRICS

	4	3	2	1	0	Marks Awarded
Meets Computational Specifications (2 x) Max: 8 marks	The program meets all of the computational specifications	The program produces the correct results and displays them correctly for almost all computational specifications	The program produces correct results for most computational specs, has a few bugs	The program produces incorrect results, has several bugs	The program does not work or has many bugs	
Readability (1 X) Max: 4 marks	The code is well organized and very easy to understand, with clear comments both in-line and in headers	The code is pretty well organized, fairly easy to read, and has good comments	The code has some organization, is a challenge to read, and has minimal comments	The code is readable only by someone who knows what it is supposed to do, has few comments	The code is poorly organized and very difficult to read, with no comments	
Error Handling (1 X) Max: 4 marks	The program checks for all error conditions and handles them appropriately	The program checks for most error conditions and handles them appropriately	The program checks for some error conditions and handles them appropriately	The program checks for few error conditions and doesn't handle them appropriately	The program does not check error conditions	
Individual Contribution (1 X) Max: 4 marks	The individual contributed in a valuable way to the project. The individual is able to articulate the working of the program.	The individual did not contribute as heavily as other members but did meet all the responsibilities. Able to articulate the program	The individual did not contribute . Did meet all responsibilities but not able to articulate the working of program	The individual did not contribute neither did meet all the responsibilities but is able to articulate the working of program	The individual did not contribute nor met all responsibilities. Failed to articulate the working of program	

Total Marks:

Guess the word

Table of contents

Chapter 1 Introduction

1.1 Overview of the Topic

1.2 Scope and Importance

Chapter 2 Software Requirement Specification

2.1 Functional Requirements

2.2 Non-Functional Requirements

Chapter 3 Design

3.1 Abstract Design

Chapter 4 Implementation

4.1 Pseudo code

4.2 Code Implementation

Chapter 5 Testing

Chapter 6 Screenshots

Chapter 6 Conclusion and Future Scope

References 18

Chapter 1

Introduction

1.1 Overview of the Topic

A guess (or an act of guessing) is a swift conclusion drawn from data directly at hand, and held as probable or tentative, while the person making the guess (the guesser) admittedly lacks material for a greater degree of certainty. In many of its uses, the meaning of guessing is assumed as implicitly understood. Guessing may combine elements of deduction, induction, abduction, and the purely random selection of one choice from a set of given options. Guessing may also involve the intuition of the guesser, who may have a "gut feeling" about which answer is correct without necessarily being able to articulate a reason for having this feeling.

This project is an example of a guess game application. It is made up of client and server and uses socket

programming. To start the game, our client should get connected to server where they can enter the initial words to be guessed by the players. Then the other players join the game and try to guess the true statement entered by the first user

1.2 Scope and Importance:

The 'guess the sentence/word' game is essentially a text communication software. We need a minimum of two players to play this game up to a maximum of 6. The text entered by the first player is stored as reference text in the server and the text entered by the rest of the users is compared with the text entered by the first user. TCP/IP protocol is being used in this project.

This game can be played for mere entertainment purposes or to check likeness of thinking between people.

Chapter 2

Software Requirements Specification

2.1 Functional Requirements:

Functional requirements define the basic system behavior. Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs. This project has the following functional requirements:

Input text Field- This aspect allows the first player to type in the message that they want to store as the message to be guessed. Thereafter the second user and so on can enter their guesses in the same field.

Display field - The display field in the command prompt displays the messages sent by the user and returns if the statement is true or false.

Send- This will allow the user to send messages.

Back- This aspect will close the chat application

2.2 Non Functional Requirements:

The non-functional requirements are concerned about the operating system environment, reliability, response time and store occupancy. These are not directly experienced by the user. In our project we have kept in mind the following:

Time- Software is not resource consuming and takes very less time for execution.

Efficiency- This software will respond quickly to user input and display the output.

Portability- Software can be used on any PC.

Implementation- Software is easy to understand with comments.

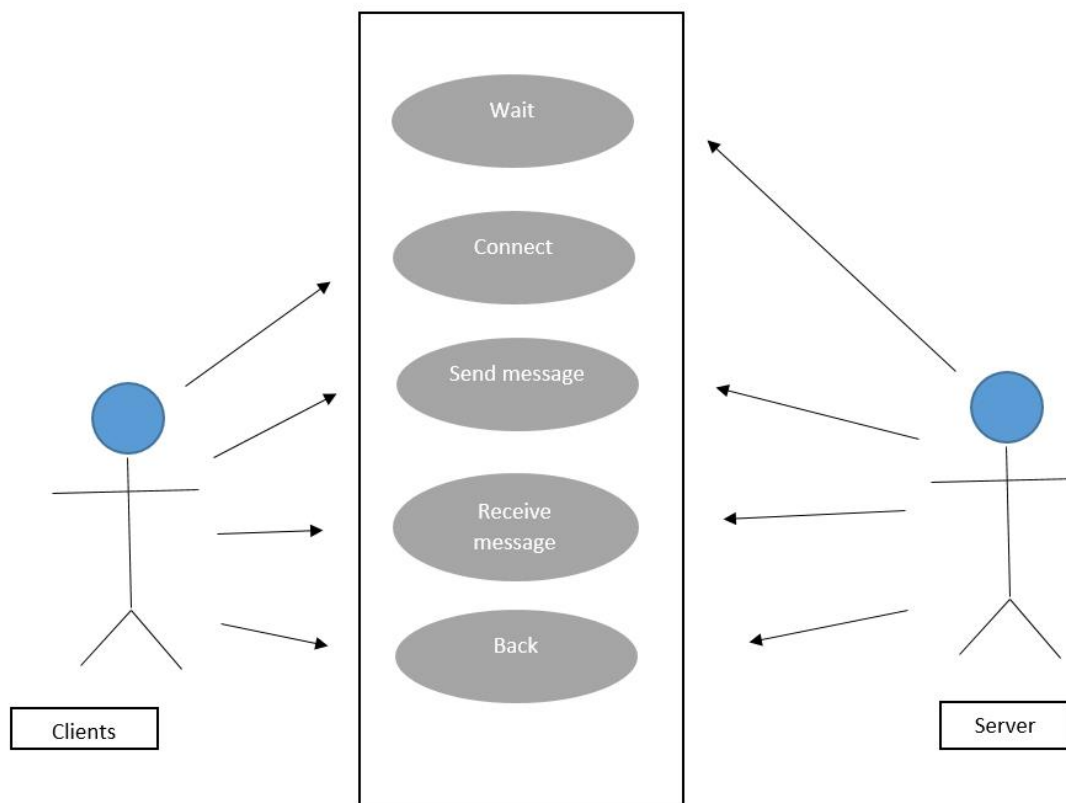
Space- Software does not use much space on the PC

Chapter 3

Design

3.1 Abstract Design

Use case diagram



Chapter 4

Implementation

4.1 Pseudo code

step 1: start

step 2: server fetches the hostname and IP address

server starts running

step 3: Client enters IP address and port number of sever

if server is started then

client running on host (hostname) and port (portnumber)

receives message from server

if message is guess

client enters the true statement

client enters the false statement

waits for second client to join

if second client joins

sends the true and false statement to him and asks him to guess

receives the answer, evaluates and prints the result

else if message is Guess the TRUE Statement Wait for Player 1 to enter the details...Enter which is true 1 or 2

receives message from client 1

enters the ans

else

prints wrong

end if

Guess the word

else

no communication

end if

step 3: server waits for client for connection

if client is connected then

connection established

if client number == 1

sends message enter a true statement,enter a false statement

else

prints the true statement

prints the false statement

asks the client which is true

end if

else

server waits for client for connection

end if

4.2 Code Implementation

Server code:

```
import socket
import sys
import random
clientNumber = 0
clients = []
statements = {}
order = [None]
```


Guess the word

```
def send(c, msg): # message entered by the client is sent to server
    packet = bytes(str(msg), 'utf-8') # messages are converted to bytes
    c.send(packet) # messages in the form of bytes are sent to server

def guess(num):
    for k, v in statements.items(): # reading values from dictionary
        if order[num] == v:
            if k == 'True':
                broadcast("Correct Answer")
                break
            else:
                broadcast("Incorrect Answer")
                break

def broadcast(msg): # send message to all the connected clients
    for c in clients:
        try:
            send(c, msg)
        except:
            # removes the client from the client array if connection is
            failed and message is not sent
            clients.remove(c)

def genrateStatements():
    val = random.sample(list(statements.values()), len(statements))
    order.append(val[0])
    order.append(val[1])
    msg = '1. '+order[1]+'\\n2. '+order[2]
    return msg

hostname = socket.gethostname() # will fetch the system host name
ip = socket.gethostbyname(hostname) # will fetch the system's IP address
port = 9999
```

Guess the word

```
# AF_INET represents internet address family and SOCK_STREAM represents
protocol that is used to transport messages
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print('Server running on Host ' + ip + ' | Port ' + str(port))

# Bind and Listen
try:
    s.bind((ip, port)) # establishes connection btw the IP and port
specified
except socket.error as e:
    print(str(e))
s.listen(6) # Listen to n client here 6

while True:
    c, addr = s.accept() # accepts client's IP and socket address
    clientNumber += 1
    clients.append(c) # appends clients to the client list
    print("Connection " + str(clientNumber) +
          " established from: " + str(addr))
    if clientNumber == 1:
        send(c, 'Guess')

        rcvd = c.recv(1024) # used to receive message from the client
        # used to decode the message received from client
        truth = bytes(rcvd).decode('utf-8')
        statements['True'] = truth
        print(truth)
        rcvd1 = c.recv(1024)
        false = bytes(rcvd1).decode('utf-8')
        statements['False'] = false
        print(false)

    else:
        send(c, 'Guess the Truth')
        msg = generateStatements()
        send(c, msg)
        rcvd1 = c.recv(1024)
        ans = bytes(rcvd1).decode('utf-8')
```

Guess the word

```
guess(int(ans))
```

Client code:

```
import socket
```

```
import sys
```

```
hostname = socket.gethostname() # will fetch the system host name
```

```
ip = socket.gethostbyname(hostname) # will fetch the system's IP address
```

```
port = 9999
```

```
s = socket.socket() # creating socket object
```

```
print('Client running on Host ' + ip + ' | Port ' + str(port))
```

```
s.connect((ip, port)) # establishes connection between IP address and port
```

```
def send(c, msg): # message entered by the client is sent to server
```

```
    packet = bytes(str(msg), 'utf-8')
```

```
    c.send(packet) # messages in the form of bytes are sent to server
```

```
msg = s.recv(1024).decode('utf-8') # message received from server is decoded
```

```
if msg == 'Guess':
```

```
    print('Enter a True statement')
```

```
    msg1 = input()
```

```
    send(s, msg1) # sends the true statement to server to broadcast
```

```
    print('Enter a False statement')
```

```
    msg2 = input()
```

```
    send(s, msg2) # sends the false statement to server to broadcast
```

```
    # message received from server is decoded
```

```
    msg = s.recv(1024).decode('utf-8')
```

```
    print(msg+" guessed by Player 2")
```

```
else:
```

```
    print(' Guess the TRUE Statement\n Wait for Player 1 to enter the  
details...\n Enter which is true 1 or 2')
```

```
    msg = s.recv(1024).decode('utf-8')
```

```
    print(msg)
```

Guess the word

```
ans = input()  # takes input from player 2
if ans == '1' or ans == '2':
    send(s, ans)
    msg1 = s.recv(1024)
    data = bytes(msg1).decode('utf-8')
    print(data)
else:
    print('Wrong')

# terminate the connection btw the client and server
s.shutdown(socket.SHUT_RDWR)
s.close()
```

Chapter 5

Testing

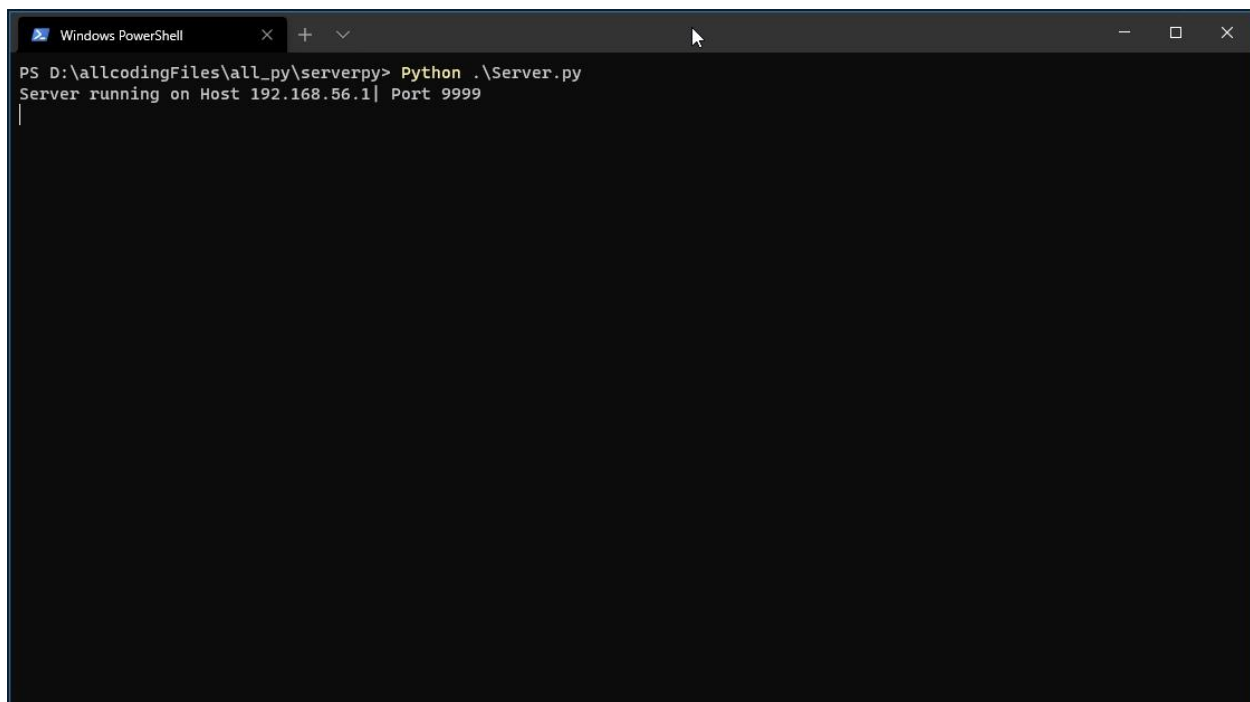
In software testing, error guessing is a test method in which test cases used to find bugs in programs are established based on experience in prior testing. The scope of test cases usually rely on the software tester involved, who uses past experience and intuition to determine what situations commonly cause software failure, or may cause errors to appear. Typical errors include divide by zero, null pointers, or invalid parameters. Error guessing has no explicit rules for testing; test cases can be designed depending on the situation, either drawing from functional documents or when an unexpected/undocumented error is found while testing operations

Guess the word

Chapter 6

Screenshots

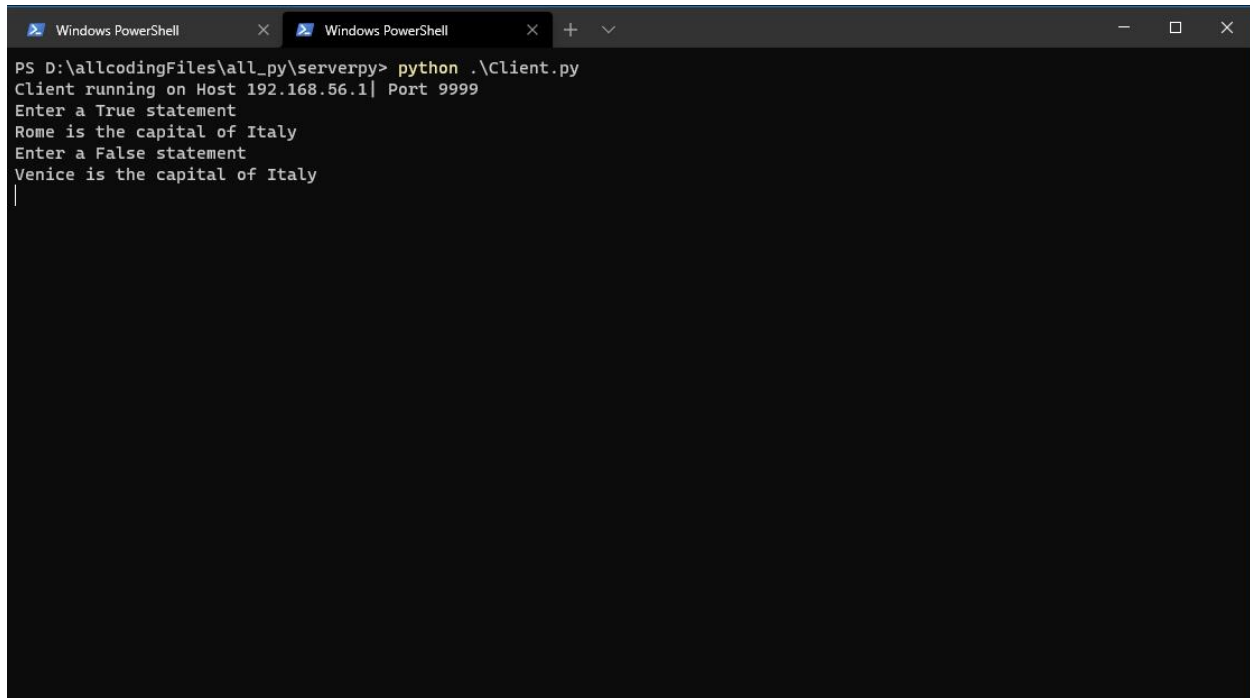
Setting up server:

A screenshot of a Windows PowerShell window. The title bar reads "Windows PowerShell". The command prompt shows the user running the command "Python .\Server.py" in the directory "D:\allcodingFiles\all_py\serverpy". The output of the command is "Server running on Host 192.168.56.1 Port 9999". The rest of the window is dark and empty.

```
PS D:\allcodingFiles\all_py\serverpy> Python .\Server.py
Server running on Host 192.168.56.1 Port 9999
```

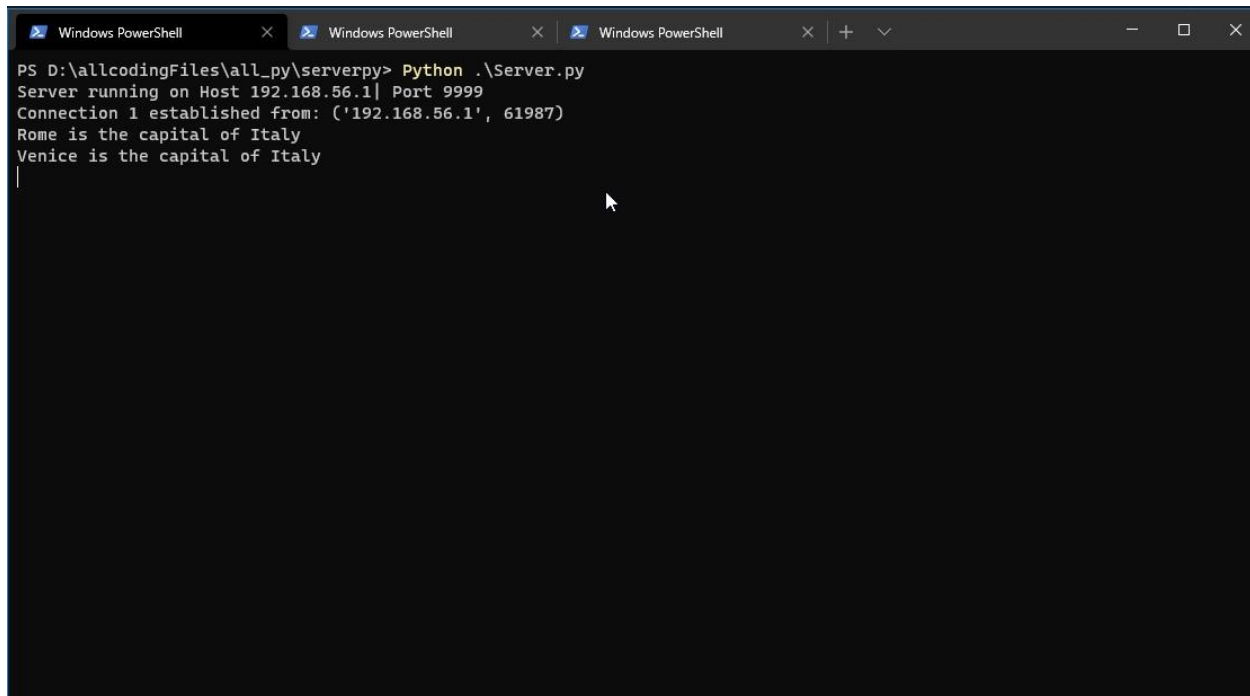
Guess the word

Client1 entering true and false statement:



```
PS D:\allcodingFiles\all_py\serverpy> python .\Client.py
Client running on Host 192.168.56.1| Port 9999
Enter a True statement
Rome is the capital of Italy
Enter a False statement
Venice is the capital of Italy
|
```

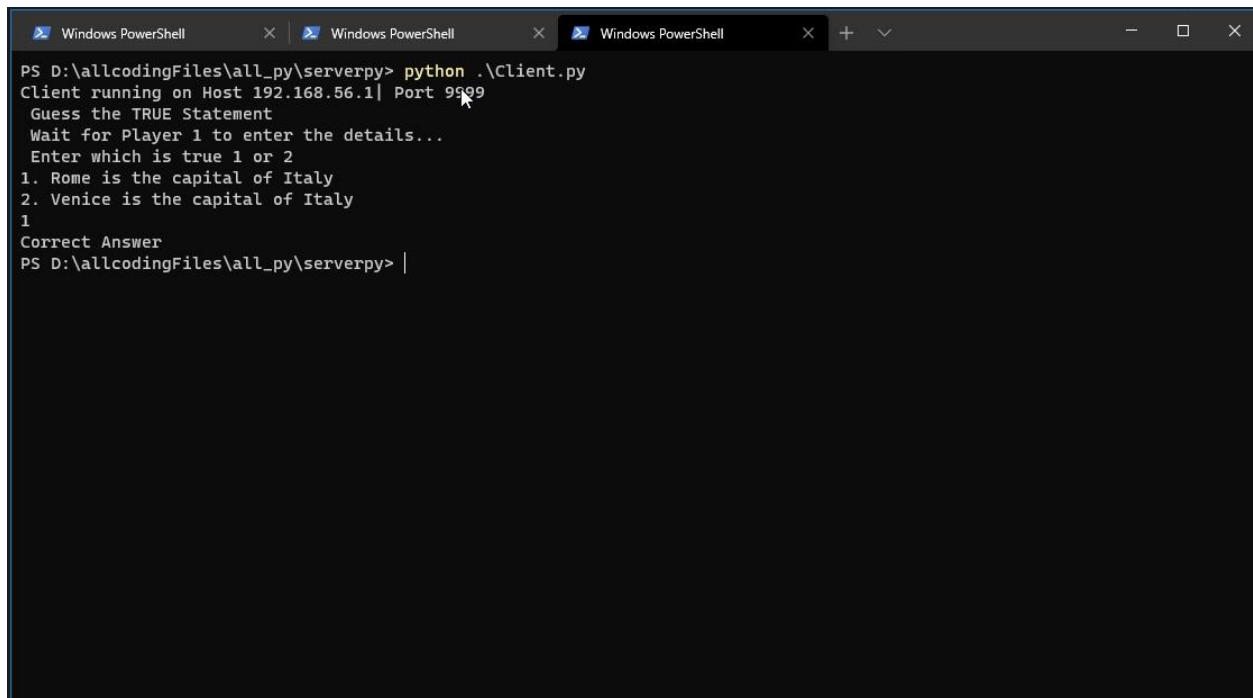
Statements stored in the server



```
PS D:\allcodingFiles\all_py\serverpy> Python .\Server.py
Server running on Host 192.168.56.1| Port 9999
Connection 1 established from: ('192.168.56.1', 61987)
Rome is the capital of Italy
Venice is the capital of Italy
|
```

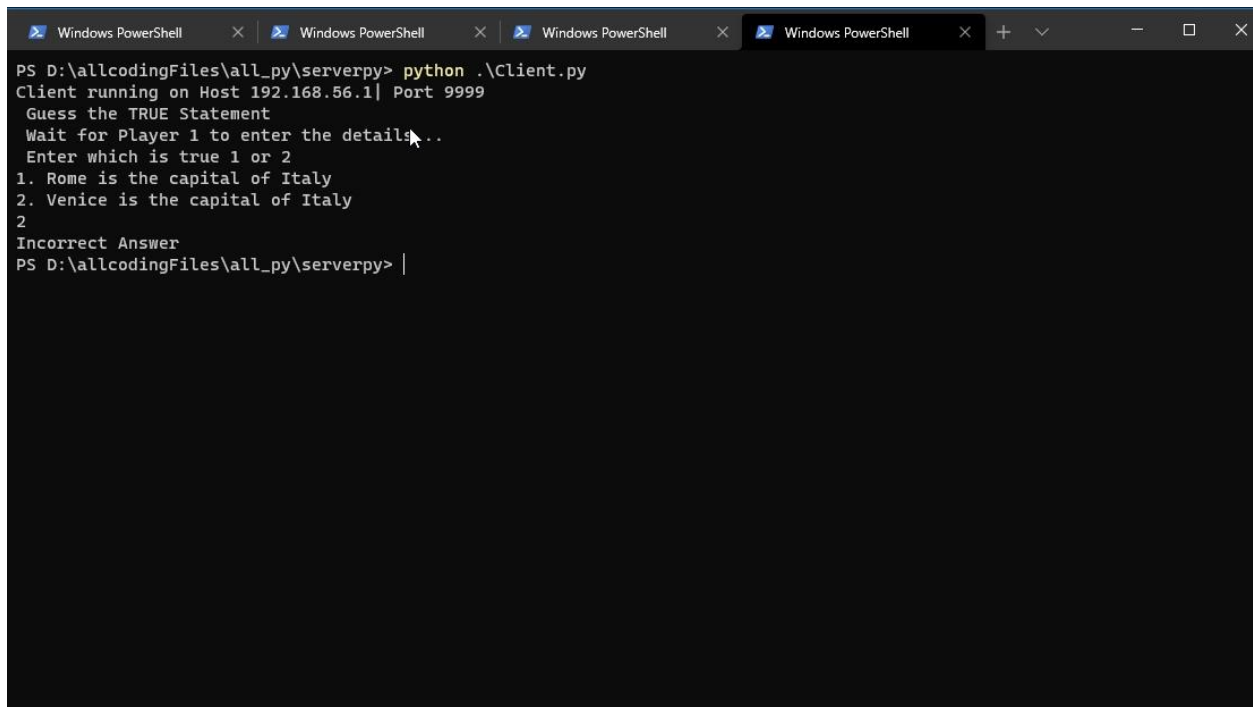
Guess the word

Client2 guessing the statement:



```
PS D:\allcodingFiles\all_py\serverpy> python .\Client.py
Client running on Host 192.168.56.1| Port 9999
Guess the TRUE Statement
Wait for Player 1 to enter the details...
Enter which is true 1 or 2
1. Rome is the capital of Italy
2. Venice is the capital of Italy
1
Correct Answer
PS D:\allcodingFiles\all_py\serverpy> |
```

Client3 guessing the statement:



```
PS D:\allcodingFiles\all_py\serverpy> python .\Client.py
Client running on Host 192.168.56.1| Port 9999
Guess the TRUE Statement
Wait for Player 1 to enter the details...
Enter which is true 1 or 2
1. Rome is the capital of Italy
2. Venice is the capital of Italy
2
Incorrect Answer
PS D:\allcodingFiles\all_py\serverpy> |
```


Chapter 7

Conclusion and Future Scope:

The main objective of the project is to develop a simple client-server guessing game using sockets. Socket allows faster transmission of messages from sender to receiver.

This is simple but an efficient game and can be further enhanced by adding these services:

- There can be a pool of users sharing similar ideas who give in the sentences to be guessed.
- Multiple accounts
- Set of statements stored in server
- Input via microphone
- A website application