Problem Statement

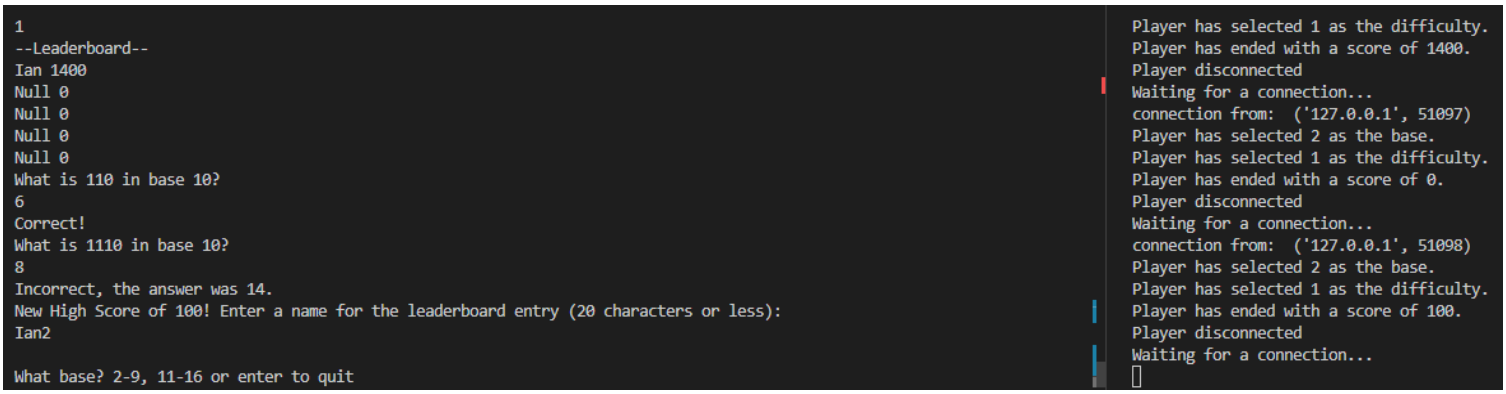
It can be hard to compare high scores in games with other players sometimes. If someone has a very high score, they may be accused of cheating or lying. With games that run and save scores entirely locally, people could easily change their scores in the files or just say they got some score.

Having the main components of the game run on a server makes it much more difficult to cheat and lying impossible. I designed a simple game with the important parts like the answer and score files on the server side.

Hardware and Software Details

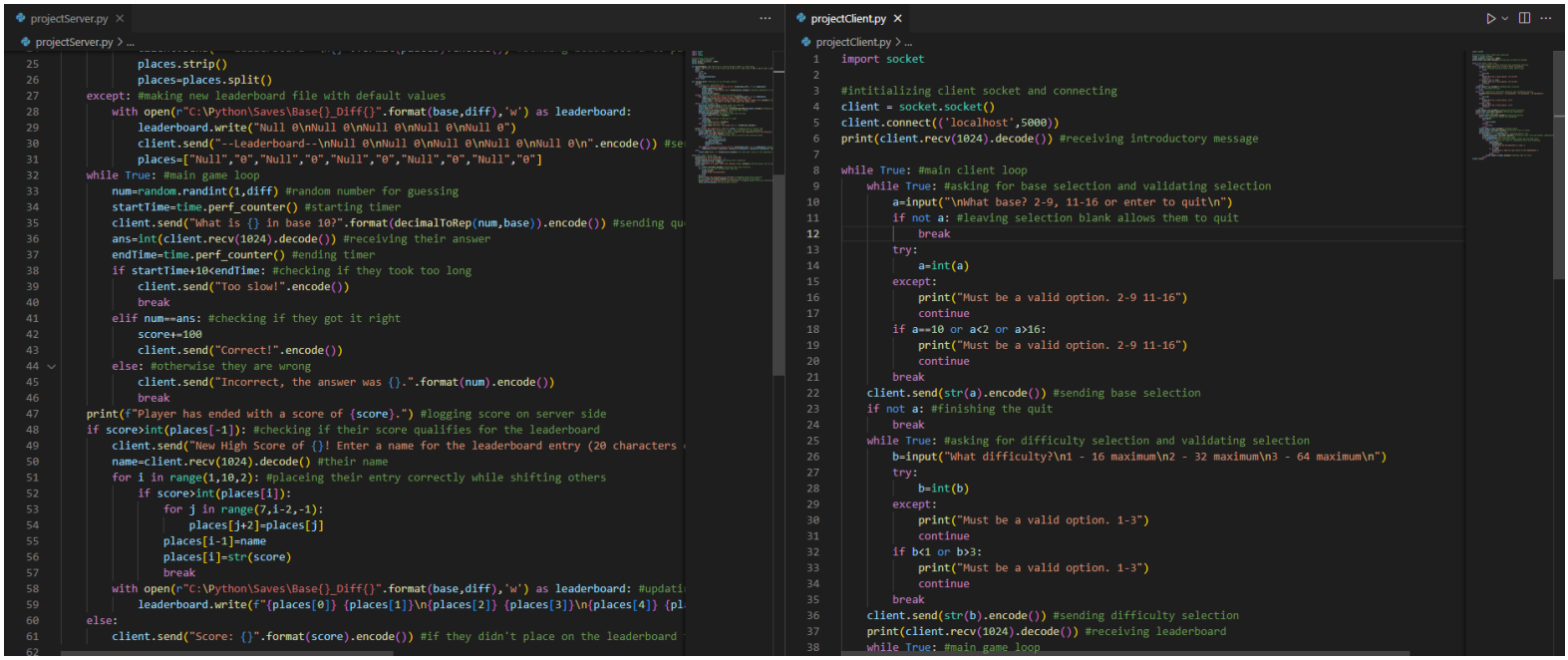
CPU - AMD Ryzen 5 3600
RAM - 8x2GB DDR3 2133MHz
SSD - Samsung SSD 860 EVO
GPU - AMD Radeon RX 5700 XT
Python version 3.11.3
Visual Studio Code version 1.77.3

Results



Above is an example of output with the client on the left and server on the right. On the client side it shows the leaderboard for the selected base and difficulty with an entry from a previous game. Then two questions, one was answered correctly and the other was not. A score greater than one on the leaderboard was reached so a name was picked for the entry.

On the server side it shows when players connected and disconnected. It also shows what base and difficulty they select as well as their final score.



projectServer.py

```
            places.strip()
            places=places.split()
    except: #making new leaderboard file with default values
        with open(r"C:\Python\Saves\Base{}_Diff{}".format(base,diff),'w') as leaderboard:
            leaderboard.write("Null 0\nNull 0\nNull 0\nNull 0\nNull 0")
            client.send("--Leaderboard--\nNull 0\nNull 0\nNull 0\nNull 0\nNull 0\n".encode()) #se
            places=["Null","0","Null","0","Null","0","Null","0","Null","0"]
    while True: #main game loop
        num=random.randint(1,diff) #random number for guessing
        startTime=time.perf_counter() #starting timer
        client.send("What is {} in base 10?".format(decimalToRep(num,base)).encode()) #sending qu
        ans=int(client.recv(1024).decode()) #receiving their answer
        endTime=time.perf_counter() #ending timer
        if startTime+10<endTime: #checking if they took too long
            client.send("Too slow!".encode())
            break
        elif num==ans: #checking if they got it right
            score+=100
            client.send("Correct!".encode())
        else: #otherwise they are wrong
            client.send("Incorrect, the answer was {}.".format(num).encode())
            break
    print(f"Player has ended with a score of {score}.") #logging score on server side
    if score>int(places[-1]): #checking if their score qualifies for the leaderboard
        client.send("New High Score of {}! Enter a name for the leaderboard entry (20 characters
        name=client.recv(1024).decode() #their name
        for i in range(1,10,2): #placeing their entry correctly while shifting others
            if score>int(places[i]):
                for j in range(7,i-2,-1):
                    places[j+2]=places[j]
                places[i-1]=name
                places[i]=str(score)
                break
        with open(r"C:\Python\Saves\Base{}_Diff{}".format(base,diff),'w') as leaderboard: #updati
            leaderboard.write(f"{places[0]} {places[1]}\n{places[2]} {places[3]}\n{places[4]} {pl
    else:
        client.send("Score: {}".format(score).encode()) #if they didn't place on the leaderboard
```

projectClient.py

```
import socket

#intitializing client socket and connecting
client = socket.socket()
client.connect(('localhost',5000))
print(client.recv(1024).decode()) #receiving introductory message

while True: #main client loop
    while True: #asking for base selection and validating selection
        a=input("\nWhat base? 2-9, 11-16 or enter to quit\n")
        if not a: #leaving selection blank allows them to quit
            break
        try:
            a=int(a)
        except:
            print("Must be a valid option. 2-9 11-16")
            continue
        if a==10 or a<2 or a>16:
            print("Must be a valid option. 2-9 11-16")
            continue
        break
    client.send(str(a).encode()) #sending base selection
    if not a: #finishing the quit
        break
    while True: #asking for difficulty selection and validating selection
        b=input("What difficulty?\n1 - 16 maximum\n2 - 32 maximum\n3 - 64 maximum\n")
        try:
            b=int(b)
        except:
            print("Must be a valid option. 1-3")
            continue
        if b<1 or b>3:
            print("Must be a valid option. 1-3")
            continue
        break
    client.send(str(b).encode()) #sending difficulty selection
    print(client.recv(1024).decode()) #receiving leaderboard
    while True: #main game loop
```

Here is a snippet of the code. The left is some of the code for the server side and the right is some of the code for the client side.

Appendix

projectServer.py below is the server side code.

```
import socket
import time
import random

#initializing server socket
server = socket.socket()
server.bind(('localhost', 5000))
server.listen()

def decimalToRep(i, b): #function to convert decimal numbers to other
bases
    table =
{0:'0',1:'1',2:'2',3:'3',4:'4',5:'5',6:'6',7:'7',8:'8',9:'9',10:'A',11:'B'
,12:'C',13:'D',14:'E',15:'F'}
    answ=""
    while i>0:
        rem = i%b
        i=i//b
        answ=table[rem]+answ
    return answ

def run(base,diff): #function to run the game instance
    score=0
    try: #looking for leaderboard file
        with open(r"C:\Python\Saves\Base{}_Diff{}".format(base,diff),'r')
as leaderboard:
            places=leaderboard.read()
            client.send("--Leaderboard--\n{}".format(places).encode())
#sending leaderboard to player
            places.strip()
            places=places.split()
    except: #making new leaderboard file with default values
        with open(r"C:\Python\Saves\Base{}_Diff{}".format(base,diff),'w')
as leaderboard:
            leaderboard.write("Null 0\nNull 0\nNull 0\nNull 0\nNull 0")
            client.send("--Leaderboard--\nNull 0\nNull 0\nNull 0\nNull
0\nNull 0\n".encode()) #sending leaderboard to player
```

```
places=["Null","0","Null","0","Null","0","Null","0","Null","0"]
    while True: #main game loop
        num=random.randint(1,diff) #random number for guessing
        startTime=time.perf_counter() #starting timer
        client.send("What is {} in base
10?".format(decimalToRep(num,base)).encode()) #sending question to player
        ans=int(client.recv(1024).decode()) #receiving their answer
        endTime=time.perf_counter() #ending timer
        if startTime+10<endTime: #checking if they took too long
            client.send("Too slow!".encode())
            break
        elif num==ans: #checking if they got it right
            score+=100
            client.send("Correct!".encode())
        else: #otherwise they are wrong
            client.send("Incorrect, the answer was
{}.".format(num).encode())
            break
    print(f"Player has ended with a score of {score}.") #logging score on
server side
    if score>int(places[-1]): #checking if their score qualifies for the
leaderboard
        client.send("New High Score of {}! Enter a name for the
leaderboard entry (20 characters or less): ".format(score).encode())
#asking player for a name for leaderboard entry
        name=client.recv(1024).decode() #their name
        for i in range(1,10,2): #placing their entry correctly while
shifting others
            if score>int(places[i]):
                for j in range(7,i-2,-1):
                    places[j+2]=places[j]
                places[i-1]=name
                places[i]=str(score)
                break
        with open(r"C:\Python\Saves\Base{}_Diff{}".format(base,diff),'w')
as leaderboard: #updating leaderboard file
            leaderboard.write(f"{places[0]} {places[1]}\n{places[2]}
{places[3]}\n{places[4]} {places[5]}\n{places[6]} {places[7]}\n{places[8]}
{places[9]}")
```

```
    else:
        client.send("Score: {}".format(score).encode()) #if they didn't
place on the leaderboard telling them their score

while True: #main server loop
    difficulties = {1:16,2:32,3:64}
    print("Waiting for a connection...")
    (client,address)=server.accept() #accepting their connection
    print("connection from: ",address)
    client.send("Here is a game, enter with nothing to quit".encode())
#telling player how to quit
    while True:
        a = client.recv(1024).decode() #receiving their base selection
        if not a: #if they just pressed enter they quit
            client.close()
            print("Player disconnected")
            break
        a=int(a)
        print("Player has selected",a,"as the base.") #logging player base
selection
        b = int(client.recv(1024).decode()) #receiving their difficulty
selection
        print("Player has selected",b,"as the difficulty.") #logging
player difficulty selection
        run(a,difficulties[b]) #starting game instance
```

projectClient.py below is the client side code.

```
import socket

#initializing client socket and connecting
client = socket.socket()
client.connect(('localhost',5000))
print(client.recv(1024).decode()) #receiving introductory message

while True: #main client loop
    while True: #asking for base selection and validating selection
        a=input("\nWhat base? 2-9, 11-16 or enter to quit\n")
        if not a: #leaving selection blank allows them to quit
            break
        try:
```

```
            a=int(a)
        except:
            print("Must be a valid option. 2-9 11-16")
            continue
        if a==10 or a<2 or a>16:
            print("Must be a valid option. 2-9 11-16")
            continue
        break
    client.send(str(a).encode()) #sending base selection
    if not a: #finishing the quit
        break
    while True: #asking for difficulty selection and validating selection
        b=input("What difficulty?\n1 - 16 maximum\n2 - 32 maximum\n3 - 64
maximum\n")
        try:
            b=int(b)
        except:
            print("Must be a valid option. 1-3")
            continue
        if b<1 or b>3:
            print("Must be a valid option. 1-3")
            continue
        break
    client.send(str(b).encode()) #sending difficulty selection
    print(client.recv(1024).decode()) #receiving leaderboard
    while True: #main game loop
        print(client.recv(1024).decode()) #receiving question
        while True: #asking for answer and validating
            ans=input()
            try:
                ans=int(ans)
            except:
                continue
            break
        client.send(str(ans).encode()) #sending answer
        result=client.recv(1024).decode() #receiving result of answer
        print(result) #letting player know
        if result[0]=="T" or result[0]=="I": #checking game is over
            finalResult=client.recv(1024).decode() #receiving final score
and whether leaderboard entry
```

```
            print(finalResult) #letting player know
            if finalResult[0]=="N": #checking whether player got new high
score
                while True: #asking for name for leaderboard entry and
validating
                    name=input()
                    if len(name)>20:
                        print("Must be 20 characters or less.")
                        continue
                    if not name:
                        print("Give a name for your entry to the
leaderboard.")
                        continue
                    break
                client.send(str(name).encode()) #sending name for entry
            break
client.close()
```