QuickQuizzer

Interactive Terminal Quiz Game

By Tatiya Seehatrakul st124875

Presentation Date: 29 November 2024

Github Source Code

https://github.com/werrnnnwerrrnnnnnn/Quick-Quizzer

By Tatiya Seehatrakul st124875

Presentation Date: 29 November 2024

Purpose & Objectives:

01 **Design a Custom Protocol:** Create a clear set of rules for client-server communication, handling game actions, feedback, and errors.

02 **Provide Engaging Gameplay:** Offer two game modes with interactive challenges and user-friendly instructions.

03 **Use TCP for Reliable Communication:**Ensure accurate, ordered, and lossless message delivery.

04 **Real-Time Feedback:** Measure response times, track scores, and provide live updates during gameplay.

05 **Handle Errors Gracefully:** Validate inputs and give meaningful error messages for incorrect entries.

06 **Practical Networking:** Gain experience with socket programming, multithreading, and latency tracking.

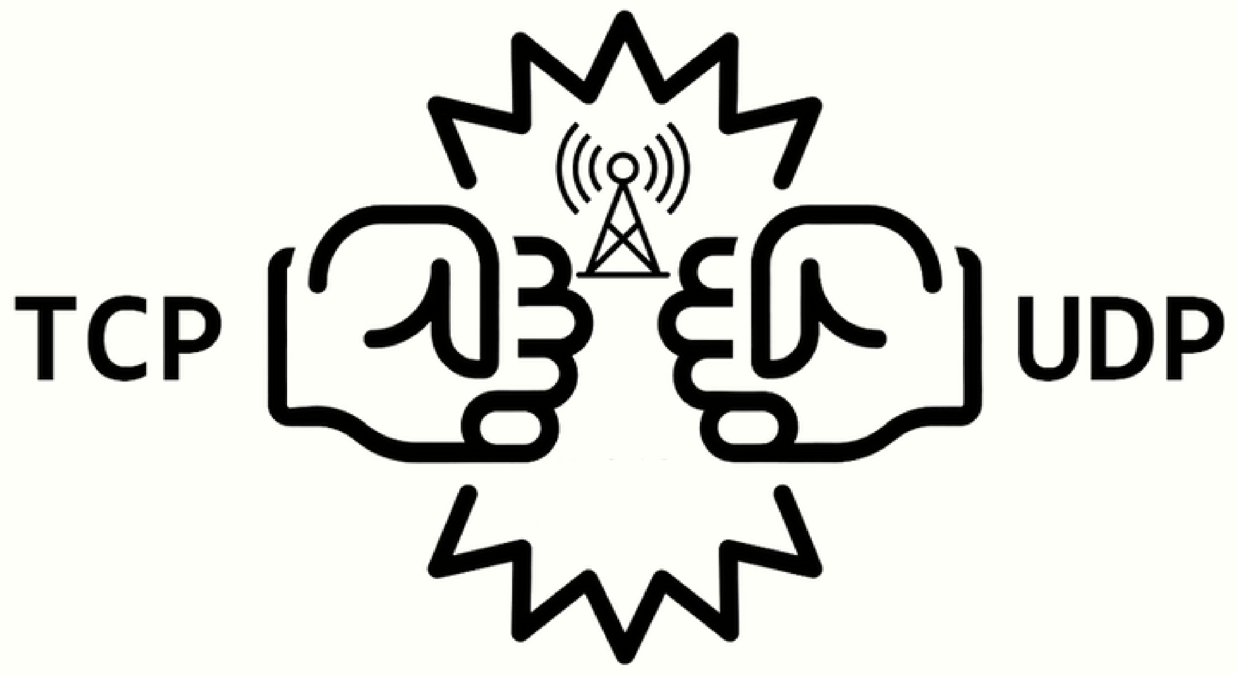
Why TCP?

`socket.socket(socket.AF_INET, socket.SOCK_STREAM)`

01 **Reliability:** Guarantees accurate and sequential message delivery.

02 **Error-checking:** Ensures no data loss, essential for interactive games.

03 **Connection-oriented:** Maintains a stable connection throughout the session.



Game Modes:

Math Quiz

Hangman

Math Quiz

GAME MODE 1

GAME MODE 1

Math Quiz

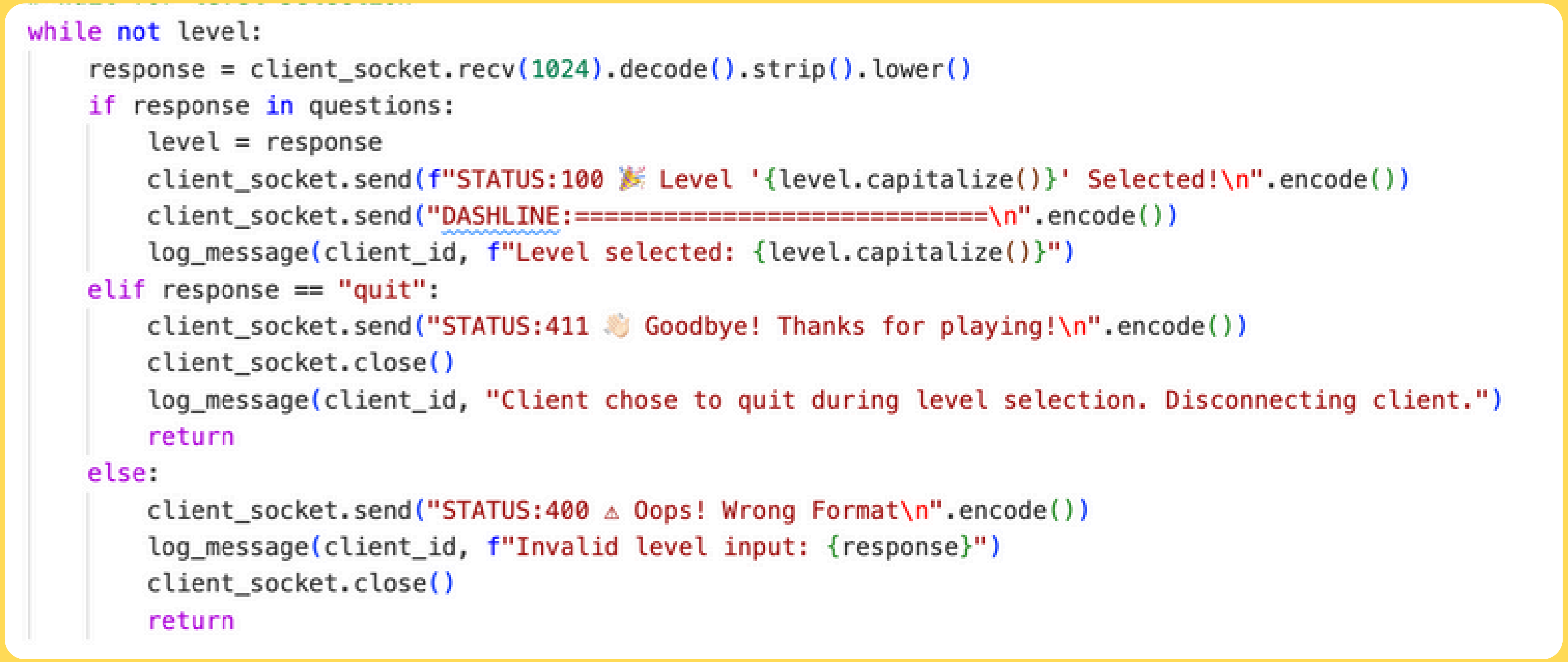
Main Features:

- **Difficulty Levels:** Easy, Medium, Hard
- **Instant Feedback:** Get messages for correct or wrong answers
- **Score Updates:** Track score for each question
- **Time Tracking:** The server records how fast you answer and shows it at the end.
- **Error Handling:** Clear error messages for invalid inputs such as non-numeric answers
- **Completion Summary:** Display final score, average response time, and correctiveness.

GAME MODE 1

Math Quiz

Server - Level Selection:

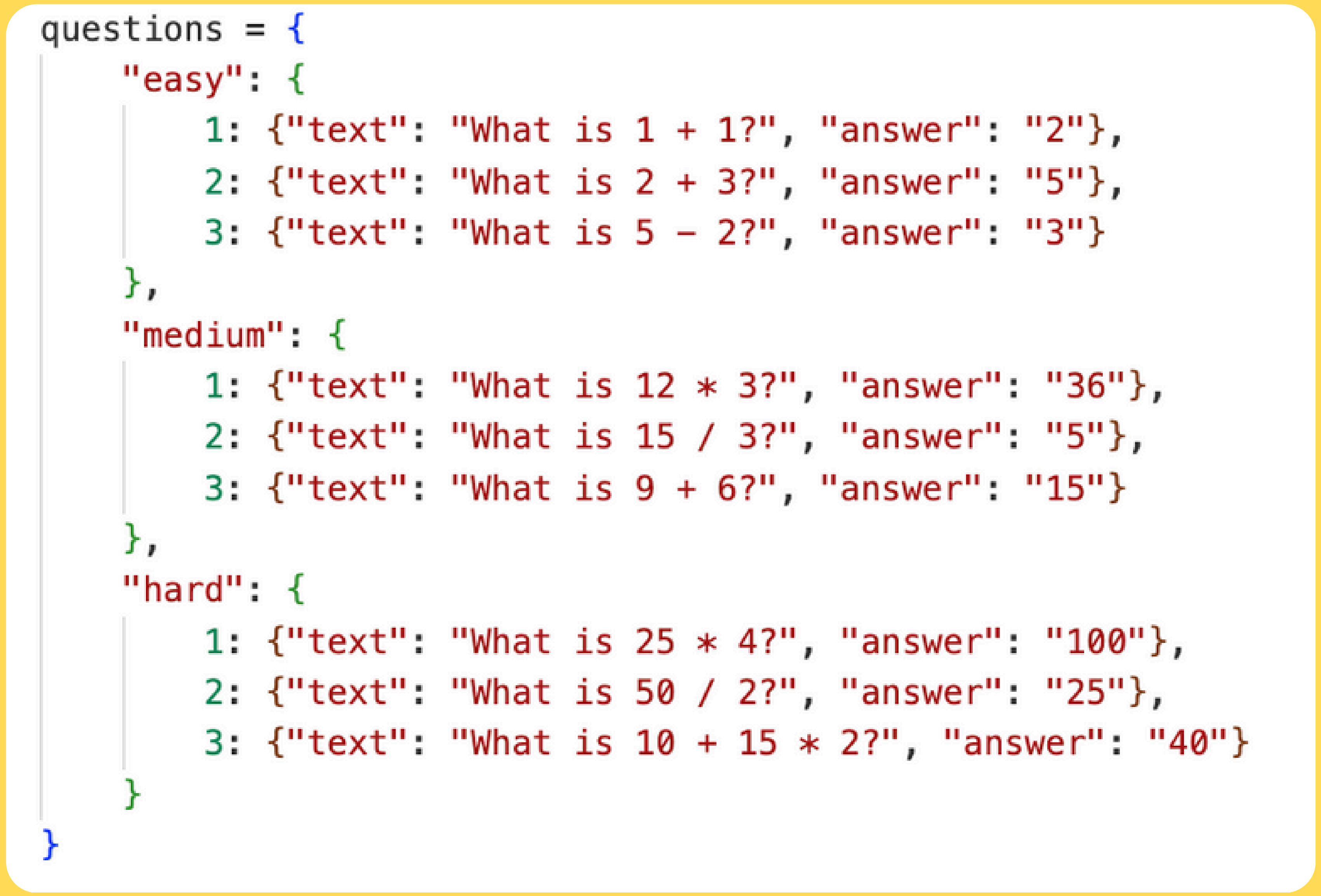


```python
while not level:
    response = client_socket.recv(1024).decode().strip().lower()
    if response in questions:
        level = response
        client_socket.send(f"STATUS:100 🎉 Level '{level.capitalize()}' Selected!\n".encode())
        client_socket.send("DASHLINE:============================\n".encode())
        log_message(client_id, f"Level selected: {level.capitalize()}")
    elif response == "quit":
        client_socket.send("STATUS:411 👋 Goodbye! Thanks for playing!\n".encode())
        client_socket.close()
        log_message(client_id, "Client chose to quit during level selection. Disconnecting client.")
        return
    else:
        client_socket.send("STATUS:400 ⚠ Oops! Wrong Format\n".encode())
        log_message(client_id, f"Invalid level input: {response}")
        client_socket.close()
        return
```

GAME MODE 1

Math Quiz

Server - Question Types:

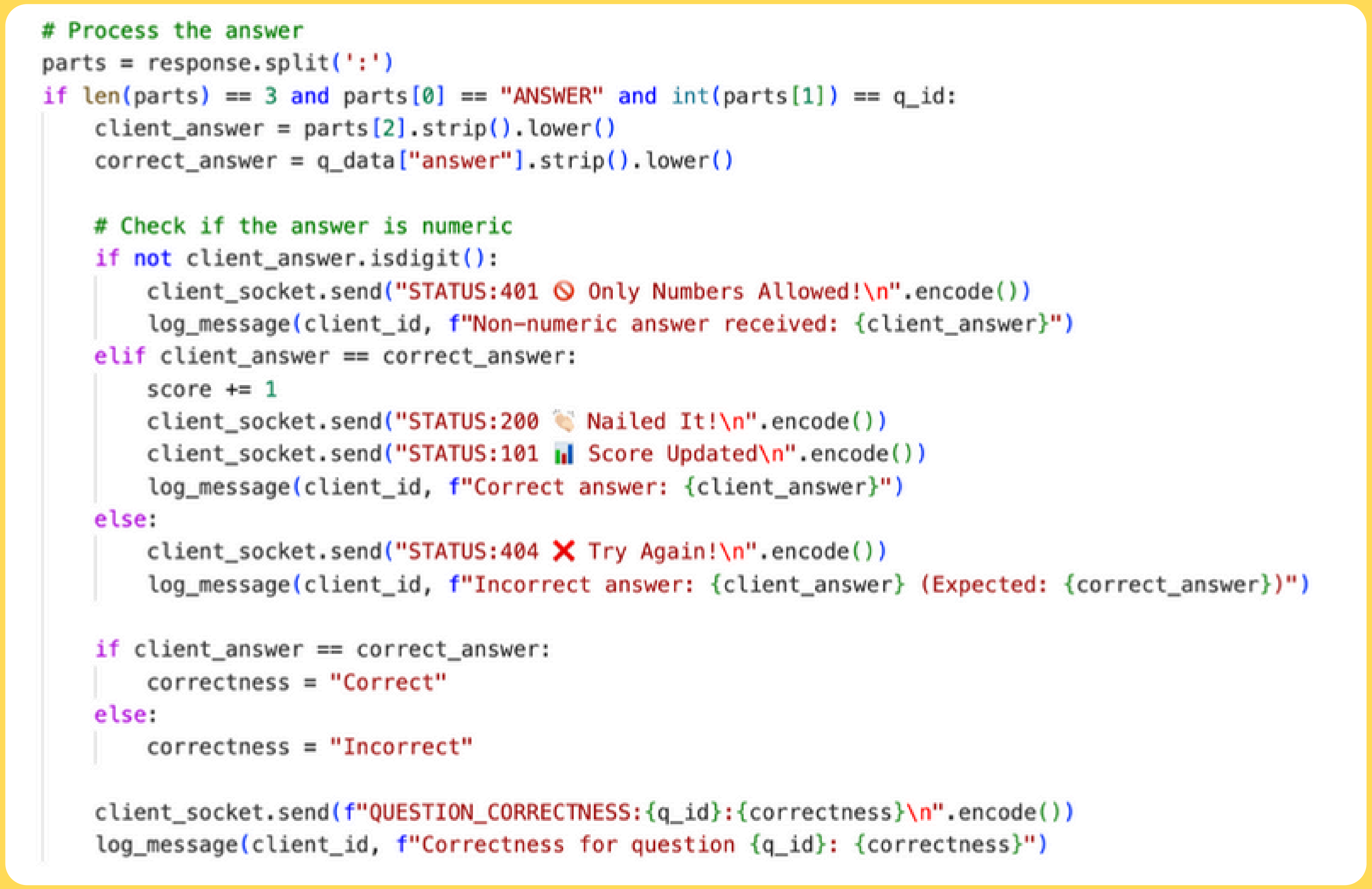


```
questions = {
    "easy": {
        1: {"text": "What is 1 + 1?", "answer": "2"},
        2: {"text": "What is 2 + 3?", "answer": "5"},
        3: {"text": "What is 5 - 2?", "answer": "3"}
    },
    "medium": {
        1: {"text": "What is 12 * 3?", "answer": "36"},
        2: {"text": "What is 15 / 3?", "answer": "5"},
        3: {"text": "What is 9 + 6?", "answer": "15"}
    },
    "hard": {
        1: {"text": "What is 25 * 4?", "answer": "100"},
        2: {"text": "What is 50 / 2?", "answer": "25"},
        3: {"text": "What is 10 + 15 * 2?", "answer": "40"}
    }
}
```

GAME MODE 1

Math Quiz

Server - Answer Checking:

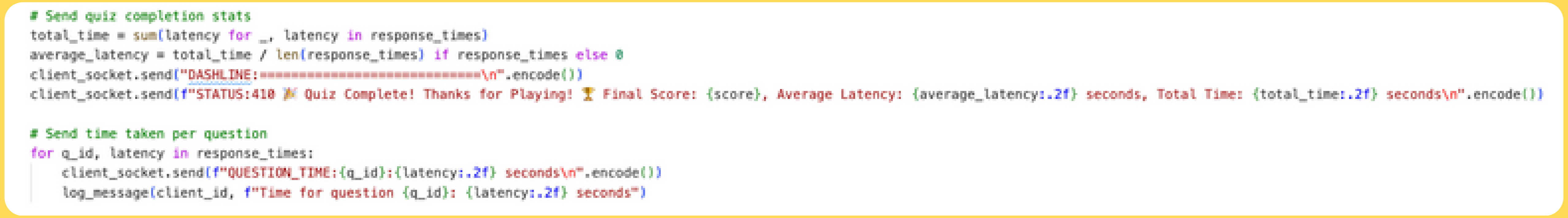


```python
# Process the answer
parts = response.split(':')
if len(parts) == 3 and parts[0] == "ANSWER" and int(parts[1]) == q_id:
    client_answer = parts[2].strip().lower()
    correct_answer = q_data["answer"].strip().lower()

    # Check if the answer is numeric
    if not client_answer.isdigit():
        client_socket.send("STATUS:401 🚫 Only Numbers Allowed!\n".encode())
        log_message(client_id, f"Non-numeric answer received: {client_answer}")
    elif client_answer == correct_answer:
        score += 1
        client_socket.send("STATUS:200 👏 Nailed It!\n".encode())
        client_socket.send("STATUS:101 📊 Score Updated\n".encode())
        log_message(client_id, f"Correct answer: {client_answer}")
    else:
        client_socket.send("STATUS:404 ❌ Try Again!\n".encode())
        log_message(client_id, f"Incorrect answer: {client_answer} (Expected: {correct_answer})")

    if client_answer == correct_answer:
        correctness = "Correct"
    else:
        correctness = "Incorrect"

    client_socket.send(f"QUESTION_CORRECTNESS:{q_id}:{correctness}\n".encode())
    log_message(client_id, f"Correctness for question {q_id}: {correctness}")
```

GAME MODE 1

Math Quiz

Server - Completion Stats:

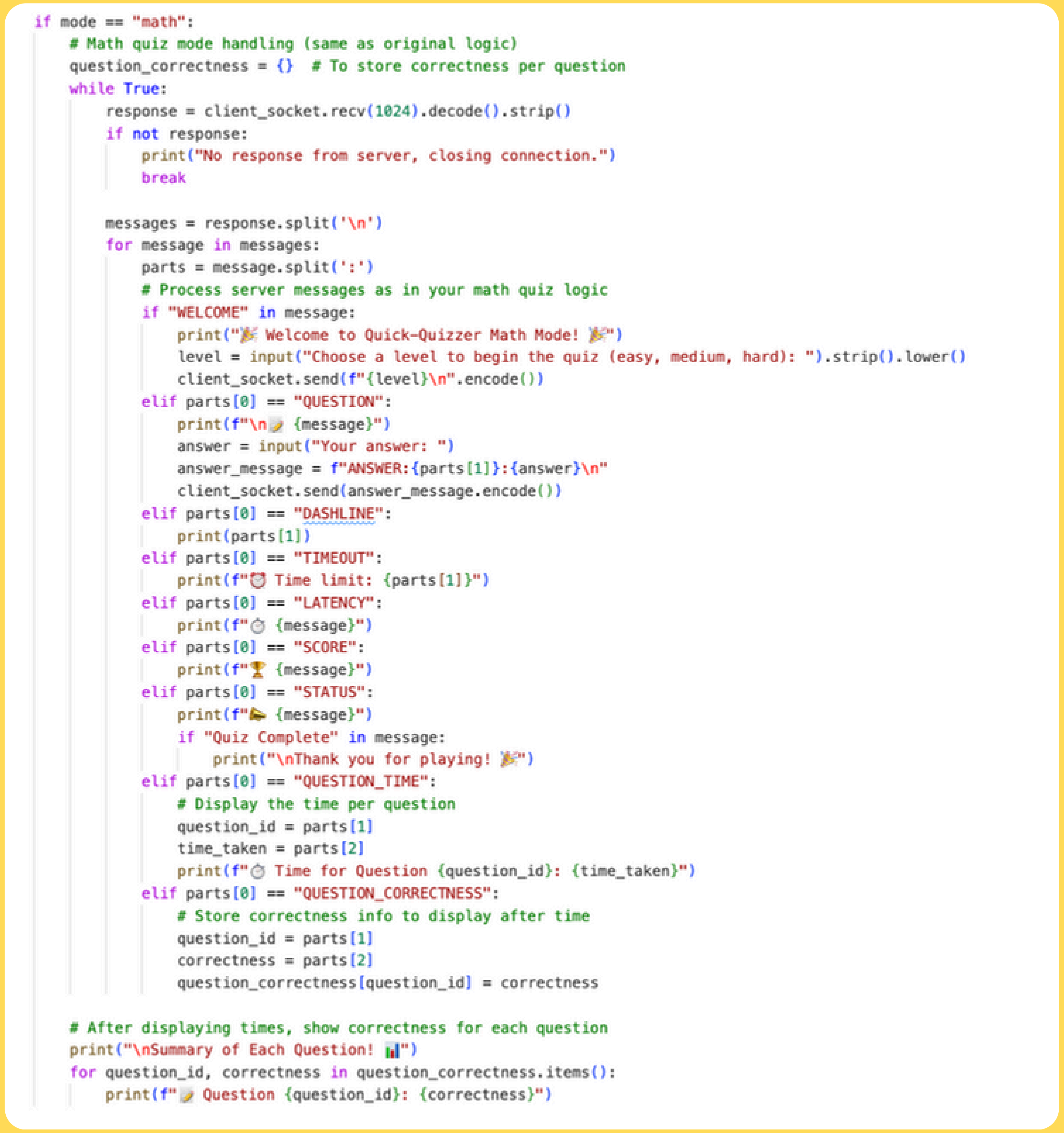


```python
# Send quiz completion stats
total_time = sum(latency for _, latency in response_times)
average_latency = total_time / len(response_times) if response_times else 0
client_socket.send("DASHLINE:==============================\n".encode())
client_socket.send(f"STATUS:410 🎉 Quiz Complete! Thanks for Playing! 🏆 Final Score: {score}, Average Latency: {average_latency:.2f} seconds, Total Time: {total_time:.2f} seconds\n".encode())

# Send time taken per question
for q_id, latency in response_times:
    client_socket.send(f"QUESTION_TIME:{q_id}:{latency:.2f} seconds\n".encode())
    log_message(client_id, f"Time for question {q_id}: {latency:.2f} seconds")
```

GAME MODE 1

Math Quiz

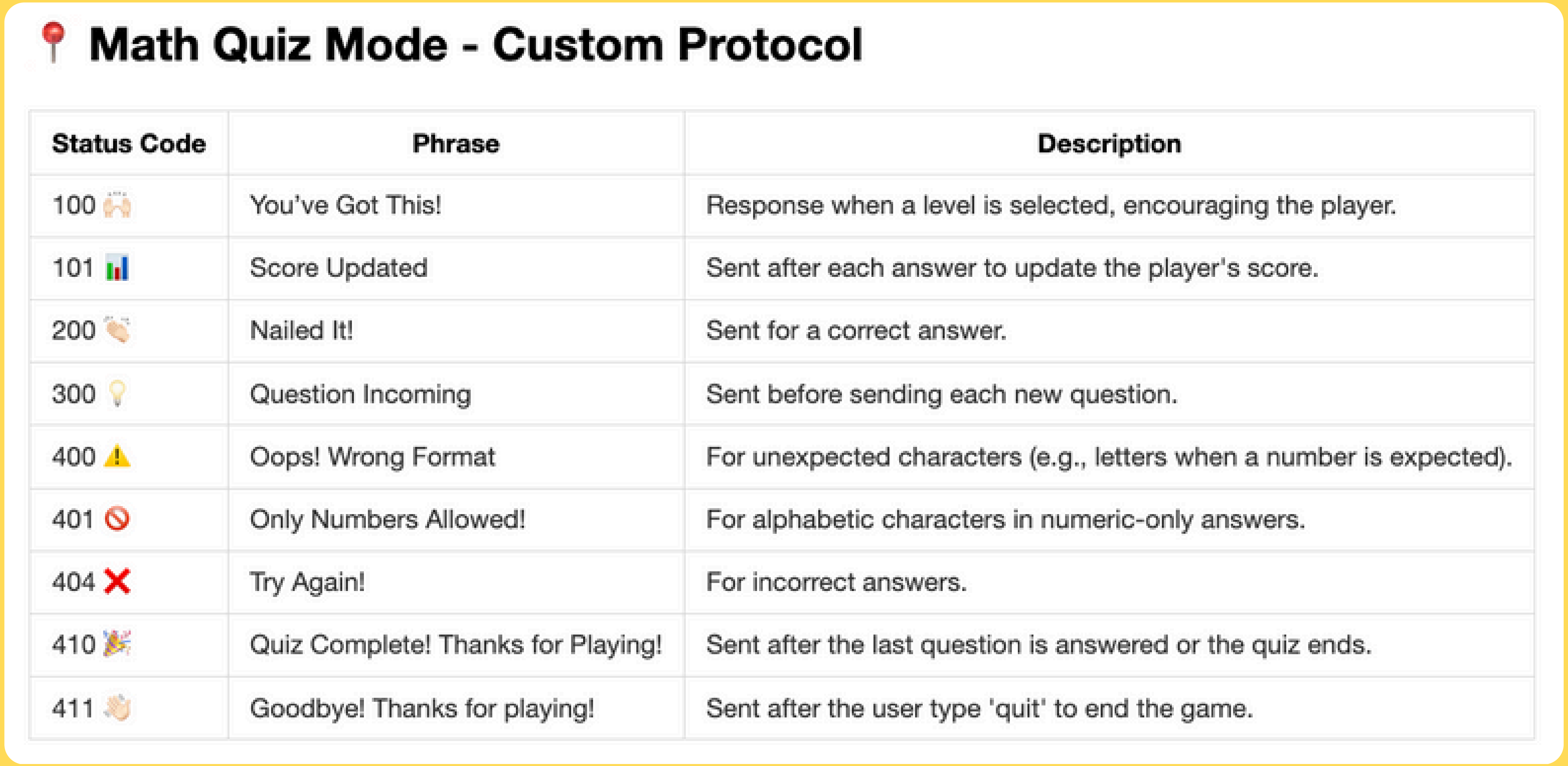
Client - Math Quiz Mode:



```python
if mode == "math":
    # Math quiz mode handling (same as original logic)
    question_correctness = {}  # To store correctness per question
    while True:
        response = client_socket.recv(1024).decode().strip()
        if not response:
            print("No response from server, closing connection.")
            break

        messages = response.split('\n')
        for message in messages:
            parts = message.split(':')
            # Process server messages as in your math quiz logic
            if "WELCOME" in message:
                print("🎉 Welcome to Quick-Quizzer Math Mode! 🎉")
                level = input("Choose a level to begin the quiz (easy, medium, hard): ").strip().lower()
                client_socket.send(f"{level}\n".encode())
            elif parts[0] == "QUESTION":
                print(f"\n📝 {message}")
                answer = input("Your answer: ")
                answer_message = f"ANSWER:{parts[1]}:{answer}\n"
                client_socket.send(answer_message.encode())
            elif parts[0] == "DASHLINE":
                print(parts[1])
            elif parts[0] == "TIMEOUT":
                print(f"⏰ Time limit: {parts[1]}")
            elif parts[0] == "LATENCY":
                print(f"⏱ {message}")
            elif parts[0] == "SCORE":
                print(f"🏆 {message}")
            elif parts[0] == "STATUS":
                print(f"📣 {message}")
                if "Quiz Complete" in message:
                    print("\nThank you for playing! 🎉")
            elif parts[0] == "QUESTION_TIME":
                # Display the time per question
                question_id = parts[1]
                time_taken = parts[2]
                print(f"⏱ Time for Question {question_id}: {time_taken}")
            elif parts[0] == "QUESTION_CORRECTNESS":
                # Store correctness info to display after time
                question_id = parts[1]
                correctness = parts[2]
                question_correctness[question_id] = correctness

    # After displaying times, show correctness for each question
    print("\nSummary of Each Question! 📊")
    for question_id, correctness in question_correctness.items():
        print(f"📝 Question {question_id}: {correctness}")
```

GAME MODE 1

Math Quiz



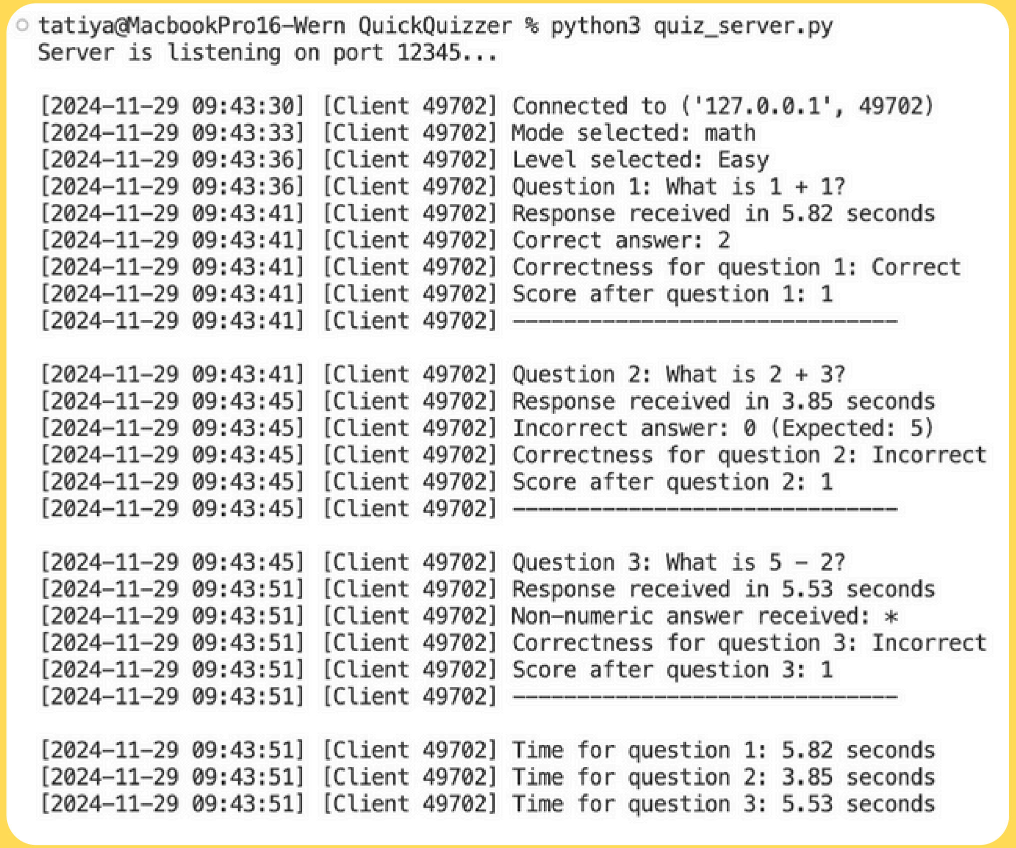
📍 Math Quiz Mode - Custom Protocol

Status Code	Phrase	Description
100 🙌	You've Got This!	Response when a level is selected, encouraging the player.
101 📊	Score Updated	Sent after each answer to update the player's score.
200 👏	Nailed It!	Sent for a correct answer.
300 💡	Question Incoming	Sent before sending each new question.
400 ⚠️	Oops! Wrong Format	For unexpected characters (e.g., letters when a number is expected).
401 🚫	Only Numbers Allowed!	For alphabetic characters in numeric-only answers.
404 ❌	Try Again!	For incorrect answers.
410 🎉	Quiz Complete! Thanks for Playing!	Sent after the last question is answered or the quiz ends.
411 👋	Goodbye! Thanks for playing!	Sent after the user type 'quit' to end the game.

GAME MODE 1

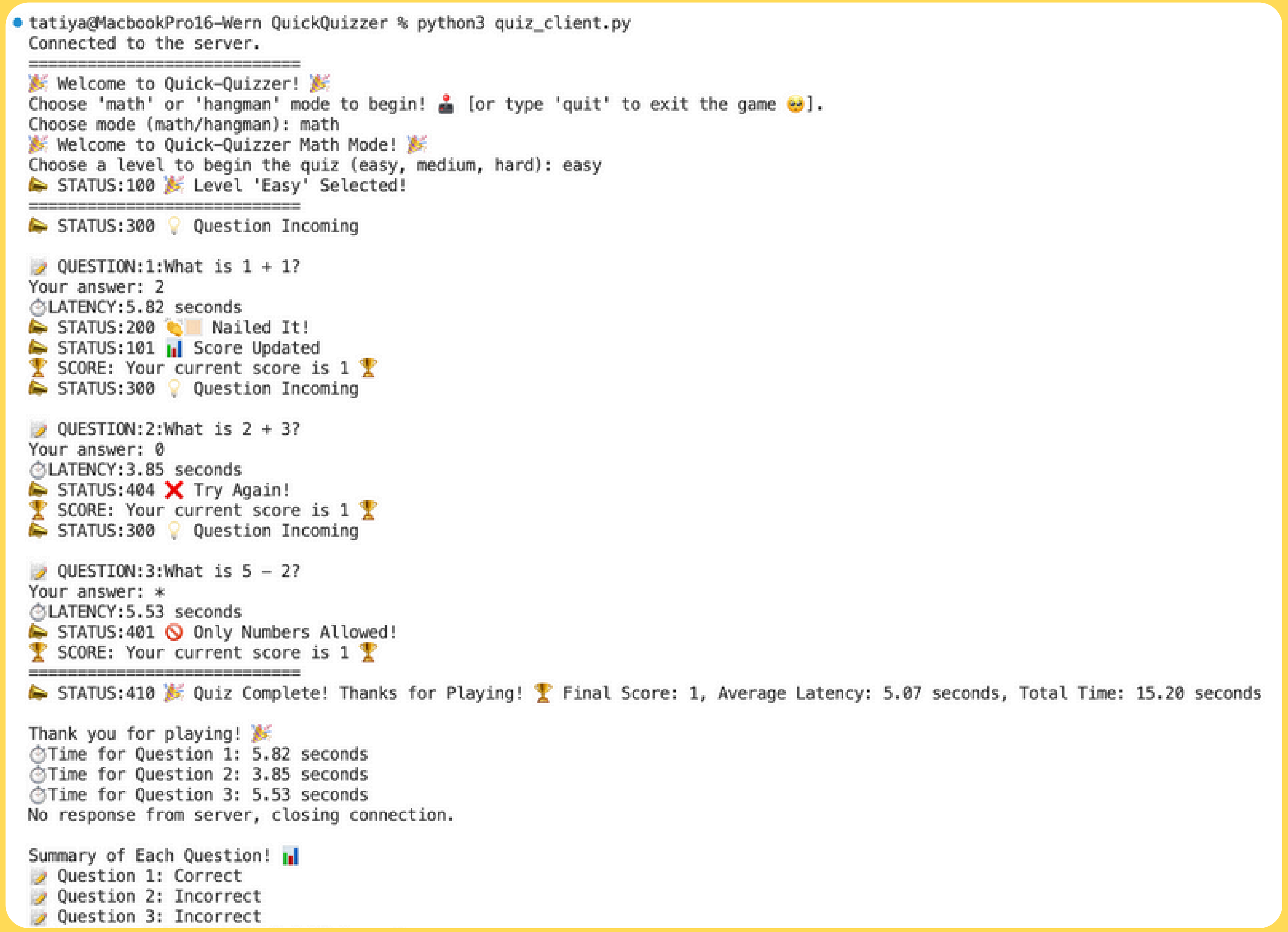
Math Quiz

Server



```
○ tatiya@MacbookPro16-Wern QuickQuizzer % python3 quiz_server.py
Server is listening on port 12345...

[2024-11-29 09:43:30] [Client 49702] Connected to ('127.0.0.1', 49702)
[2024-11-29 09:43:33] [Client 49702] Mode selected: math
[2024-11-29 09:43:36] [Client 49702] Level selected: Easy
[2024-11-29 09:43:36] [Client 49702] Question 1: What is 1 + 1?
[2024-11-29 09:43:41] [Client 49702] Response received in 5.82 seconds
[2024-11-29 09:43:41] [Client 49702] Correct answer: 2
[2024-11-29 09:43:41] [Client 49702] Correctness for question 1: Correct
[2024-11-29 09:43:41] [Client 49702] Score after question 1: 1
[2024-11-29 09:43:41] [Client 49702] ------------------------------

[2024-11-29 09:43:41] [Client 49702] Question 2: What is 2 + 3?
[2024-11-29 09:43:45] [Client 49702] Response received in 3.85 seconds
[2024-11-29 09:43:45] [Client 49702] Incorrect answer: 0 (Expected: 5)
[2024-11-29 09:43:45] [Client 49702] Correctness for question 2: Incorrect
[2024-11-29 09:43:45] [Client 49702] Score after question 2: 1
[2024-11-29 09:43:45] [Client 49702] ------------------------------

[2024-11-29 09:43:45] [Client 49702] Question 3: What is 5 - 2?
[2024-11-29 09:43:51] [Client 49702] Response received in 5.53 seconds
[2024-11-29 09:43:51] [Client 49702] Non-numeric answer received: *
[2024-11-29 09:43:51] [Client 49702] Correctness for question 3: Incorrect
[2024-11-29 09:43:51] [Client 49702] Score after question 3: 1
[2024-11-29 09:43:51] [Client 49702] ------------------------------

[2024-11-29 09:43:51] [Client 49702] Time for question 1: 5.82 seconds
[2024-11-29 09:43:51] [Client 49702] Time for question 2: 3.85 seconds
[2024-11-29 09:43:51] [Client 49702] Time for question 3: 5.53 seconds
```

Client



```
● tatiya@MacbookPro16-Wern QuickQuizzer % python3 quiz_client.py
Connected to the server.
==============================
🎉 Welcome to Quick-Quizzer! 🎉
Choose 'math' or 'hangman' mode to begin! 🕹️ [or type 'quit' to exit the game 🥺].
Choose mode (math/hangman): math
🎉 Welcome to Quick-Quizzer Math Mode! 🎉
Choose a level to begin the quiz (easy, medium, hard): easy
📣 STATUS:100 🎉 Level 'Easy' Selected!
==============================
📣 STATUS:300 💡 Question Incoming

📝 QUESTION:1:What is 1 + 1?
Your answer: 2
⏱️LATENCY:5.82 seconds
📣 STATUS:200 👏🏻 Nailed It!
📣 STATUS:101 📊 Score Updated
🏆 SCORE: Your current score is 1 🏆
📣 STATUS:300 💡 Question Incoming

📝 QUESTION:2:What is 2 + 3?
Your answer: 0
⏱️LATENCY:3.85 seconds
📣 STATUS:404 ❌ Try Again!
🏆 SCORE: Your current score is 1 🏆
📣 STATUS:300 💡 Question Incoming

📝 QUESTION:3:What is 5 - 2?
Your answer: *
⏱️LATENCY:5.53 seconds
📣 STATUS:401 🚫 Only Numbers Allowed!
🏆 SCORE: Your current score is 1 🏆
==============================
📣 STATUS:410 🎉 Quiz Complete! Thanks for Playing! 🏆 Final Score: 1, Average Latency: 5.07 seconds, Total Time: 15.20 seconds

Thank you for playing! 🎉
⏱️Time for Question 1: 5.82 seconds
⏱️Time for Question 2: 3.85 seconds
⏱️Time for Question 3: 5.53 seconds
No response from server, closing connection.

Summary of Each Question! 📊
📝 Question 1: Correct
📝 Question 2: Incorrect
📝 Question 3: Incorrect
```

Hangman

GAME MODE 2

GAME MODE 2

Hangman

_ _ _ _ _

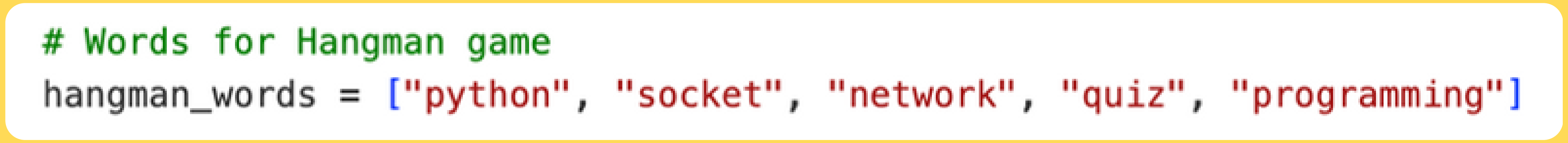
Main Features:

- **Word Guessing:** Guess letters to figure out a hidden word, shown as underscores (_).
- **Limited Attempts:** Only 6 attempts to guess the word correctly.
- **Feedback:** Correct, incorrect, Invalid guesses (numbers or repeated letters)
- **Game State:** `Win` - guessing the word correctly. `Lose` - attempts run out.
- **Progress Updates:** Show current word state and remaining attempts after every guess.

GAME MODE 2

Hangman

Server - Preset Words:

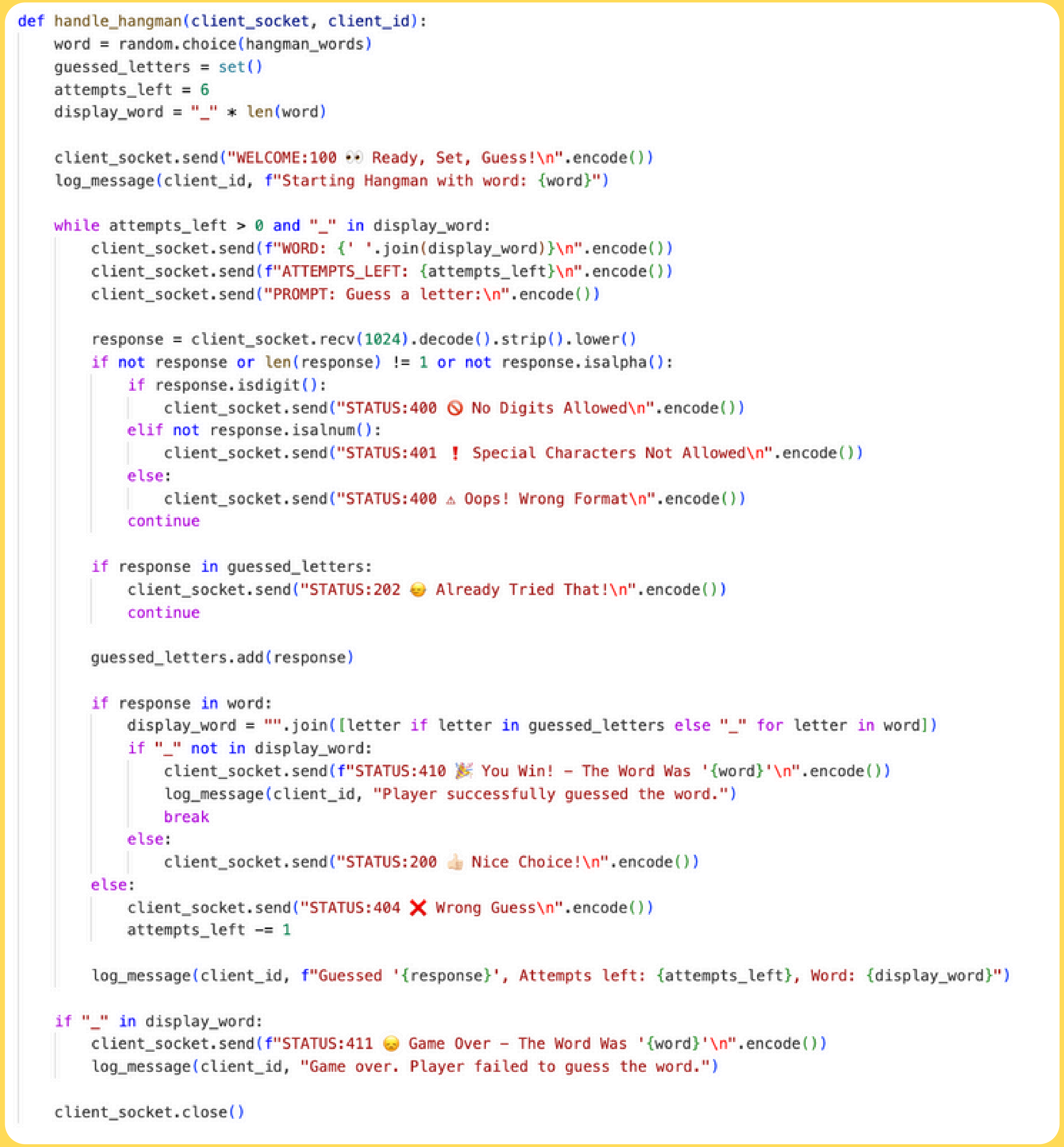


```
# Words for Hangman game
hangman_words = ["python", "socket", "network", "quiz", "programming"]
```

GAME MODE 2

Hangman

Server - Hangman Mode:

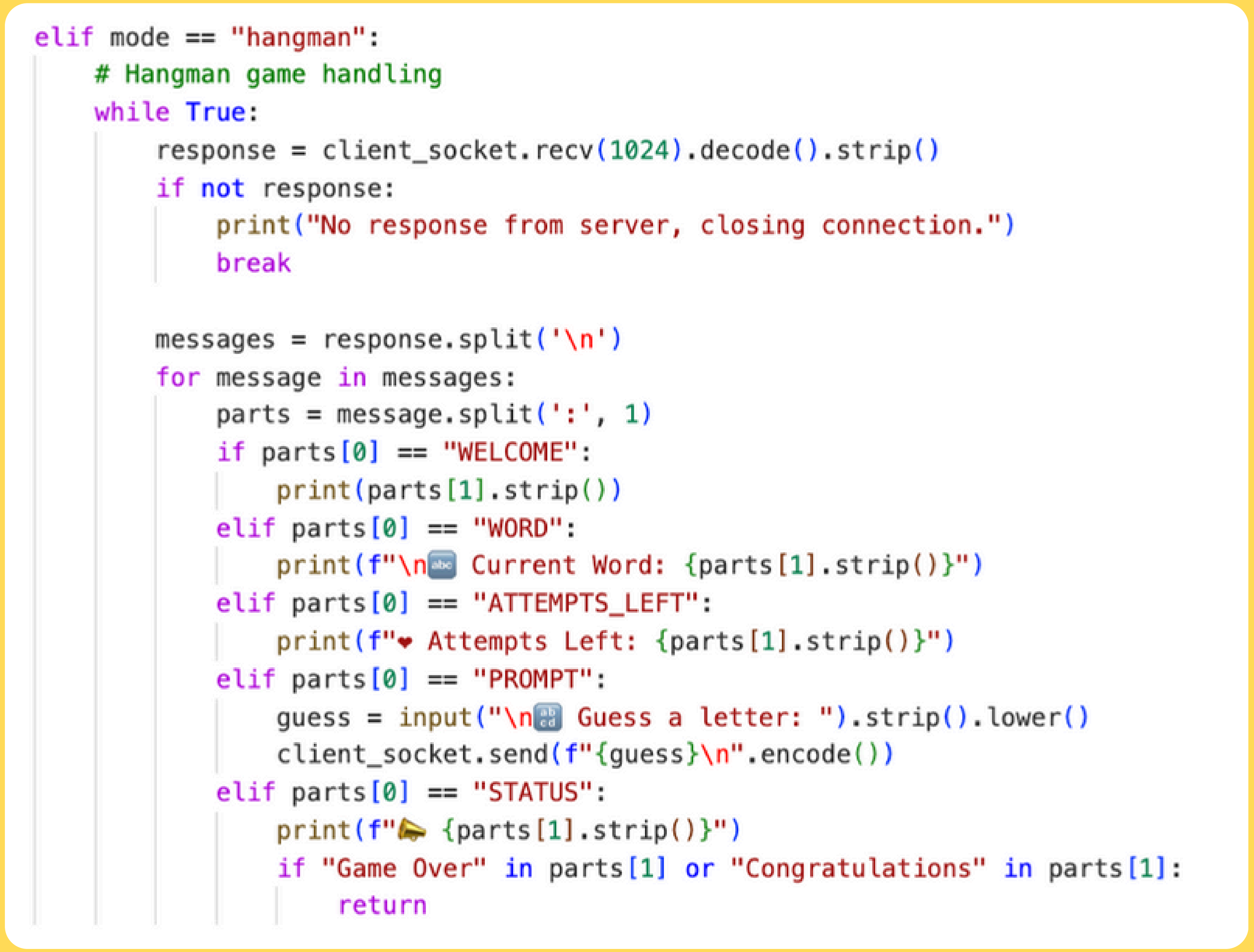


```python
def handle_hangman(client_socket, client_id):
    word = random.choice(hangman_words)
    guessed_letters = set()
    attempts_left = 6
    display_word = "_" * len(word)

    client_socket.send("WELCOME:100 👀 Ready, Set, Guess!\n".encode())
    log_message(client_id, f"Starting Hangman with word: {word}")

    while attempts_left > 0 and "_" in display_word:
        client_socket.send(f"WORD: {' '.join(display_word)}\n".encode())
        client_socket.send(f"ATTEMPTS_LEFT: {attempts_left}\n".encode())
        client_socket.send("PROMPT: Guess a letter:\n".encode())

        response = client_socket.recv(1024).decode().strip().lower()
        if not response or len(response) != 1 or not response.isalpha():
            if response.isdigit():
                client_socket.send("STATUS:400 🚫 No Digits Allowed\n".encode())
            elif not response.isalnum():
                client_socket.send("STATUS:401 ❗ Special Characters Not Allowed\n".encode())
            else:
                client_socket.send("STATUS:400 ⚠ Oops! Wrong Format\n".encode())
            continue

        if response in guessed_letters:
            client_socket.send("STATUS:202 😅 Already Tried That!\n".encode())
            continue

        guessed_letters.add(response)

        if response in word:
            display_word = "".join([letter if letter in guessed_letters else "_" for letter in word])
            if "_" not in display_word:
                client_socket.send(f"STATUS:410 🎉 You Win! - The Word Was '{word}'\n".encode())
                log_message(client_id, "Player successfully guessed the word.")
                break
            else:
                client_socket.send("STATUS:200 👍 Nice Choice!\n".encode())
        else:
            client_socket.send("STATUS:404 ❌ Wrong Guess\n".encode())
            attempts_left -= 1

        log_message(client_id, f"Guessed '{response}', Attempts left: {attempts_left}, Word: {display_word}")

    if "_" in display_word:
        client_socket.send(f"STATUS:411 😞 Game Over - The Word Was '{word}'\n".encode())
        log_message(client_id, "Game over. Player failed to guess the word.")

    client_socket.close()
```

GAME MODE 2

Hangman

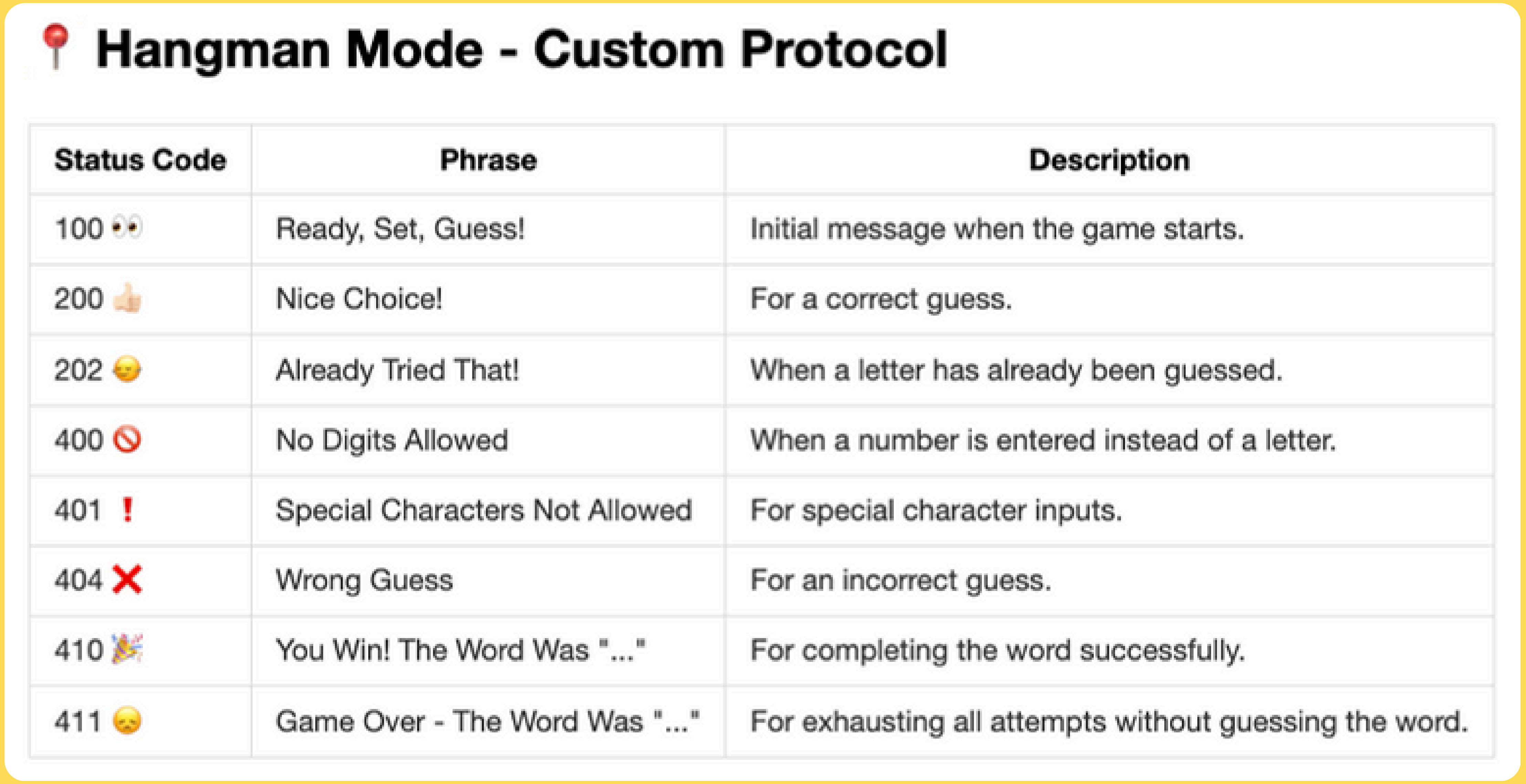
Client - Hangman Mode:



```python
elif mode == "hangman":
    # Hangman game handling
    while True:
        response = client_socket.recv(1024).decode().strip()
        if not response:
            print("No response from server, closing connection.")
            break

        messages = response.split('\n')
        for message in messages:
            parts = message.split(':', 1)
            if parts[0] == "WELCOME":
                print(parts[1].strip())
            elif parts[0] == "WORD":
                print(f"\n🔤 Current Word: {parts[1].strip()}")
            elif parts[0] == "ATTEMPTS_LEFT":
                print(f"❤ Attempts Left: {parts[1].strip()}")
            elif parts[0] == "PROMPT":
                guess = input("\n🔡 Guess a letter: ").strip().lower()
                client_socket.send(f"{guess}\n".encode())
            elif parts[0] == "STATUS":
                print(f"📢 {parts[1].strip()}")
                if "Game Over" in parts[1] or "Congratulations" in parts[1]:
                    return
```

GAME MODE 2

Hangman



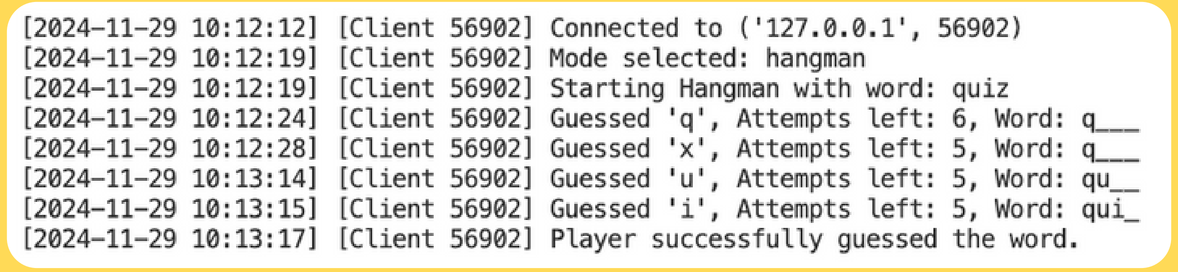
📍 Hangman Mode - Custom Protocol

Status Code	Phrase	Description
100 👀	Ready, Set, Guess!	Initial message when the game starts.
200 👍🏻	Nice Choice!	For a correct guess.
202 😏	Already Tried That!	When a letter has already been guessed.
400 🚫	No Digits Allowed	When a number is entered instead of a letter.
401 ❗	Special Characters Not Allowed	For special character inputs.
404 ❌	Wrong Guess	For an incorrect guess.
410 🎉	You Win! The Word Was "..."	For completing the word successfully.
411 😞	Game Over - The Word Was "..."	For exhausting all attempts without guessing the word.

GAME MODE 2

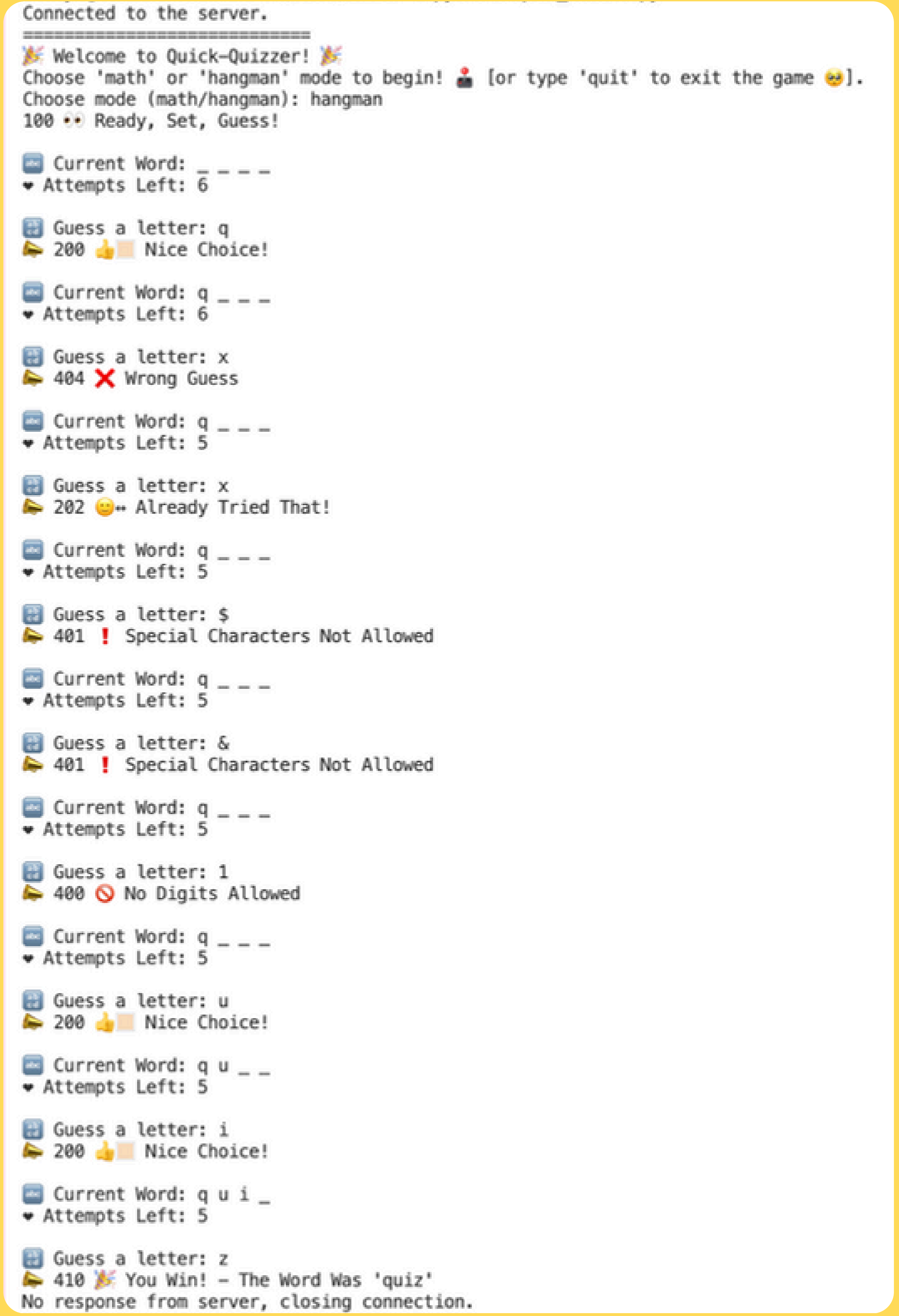
Hangman

Server



```
[2024-11-29 10:12:12] [Client 56902] Connected to ('127.0.0.1', 56902)
[2024-11-29 10:12:19] [Client 56902] Mode selected: hangman
[2024-11-29 10:12:19] [Client 56902] Starting Hangman with word: quiz
[2024-11-29 10:12:24] [Client 56902] Guessed 'q', Attempts left: 6, Word: q___
[2024-11-29 10:12:28] [Client 56902] Guessed 'x', Attempts left: 5, Word: q___
[2024-11-29 10:13:14] [Client 56902] Guessed 'u', Attempts left: 5, Word: qu__
[2024-11-29 10:13:15] [Client 56902] Guessed 'i', Attempts left: 5, Word: qui_
[2024-11-29 10:13:17] [Client 56902] Player successfully guessed the word.
```

Client – Win

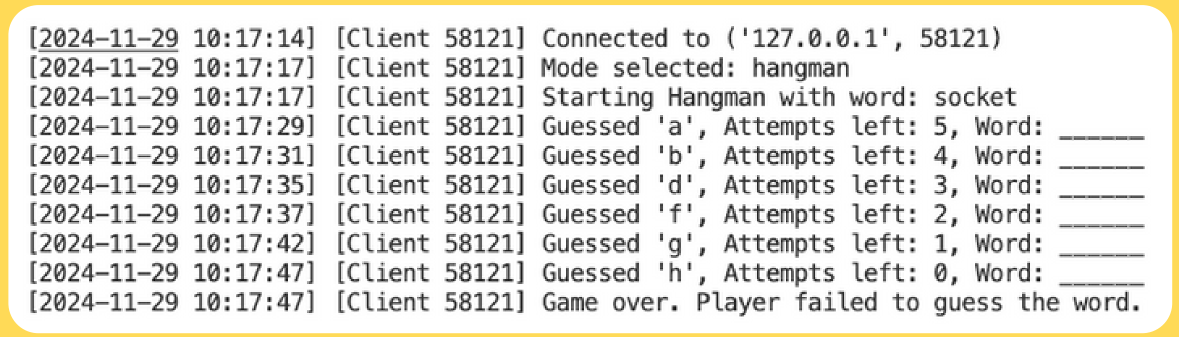


```
Connected to the server.
==============================
🎉 Welcome to Quick-Quizzer! 🎉
Choose 'math' or 'hangman' mode to begin! 🕹️ [or type 'quit' to exit the game 🥺].
Choose mode (math/hangman): hangman
100 👀 Ready, Set, Guess!

🔤 Current Word: _ _ _ _
❤ Attempts Left: 6

🔡 Guess a letter: q
📣 200 👍🏻 Nice Choice!

🔤 Current Word: q _ _ _
❤ Attempts Left: 6

🔡 Guess a letter: x
📣 404 ❌ Wrong Guess

🔤 Current Word: q _ _ _
❤ Attempts Left: 5

🔡 Guess a letter: x
📣 202 😶‍🌫️ Already Tried That!

🔤 Current Word: q _ _ _
❤ Attempts Left: 5

🔡 Guess a letter: $
📣 401 ❗ Special Characters Not Allowed

🔤 Current Word: q _ _ _
❤ Attempts Left: 5

🔡 Guess a letter: &
📣 401 ❗ Special Characters Not Allowed

🔤 Current Word: q _ _ _
❤ Attempts Left: 5

🔡 Guess a letter: 1
📣 400 🚫 No Digits Allowed

🔤 Current Word: q _ _ _
❤ Attempts Left: 5

🔡 Guess a letter: u
📣 200 👍🏻 Nice Choice!

🔤 Current Word: q u _ _
❤ Attempts Left: 5

🔡 Guess a letter: i
📣 200 👍🏻 Nice Choice!

🔤 Current Word: q u i _
❤ Attempts Left: 5

🔡 Guess a letter: z
📣 410 🎉 You Win! – The Word Was 'quiz'
No response from server, closing connection.
```

GAME MODE 2

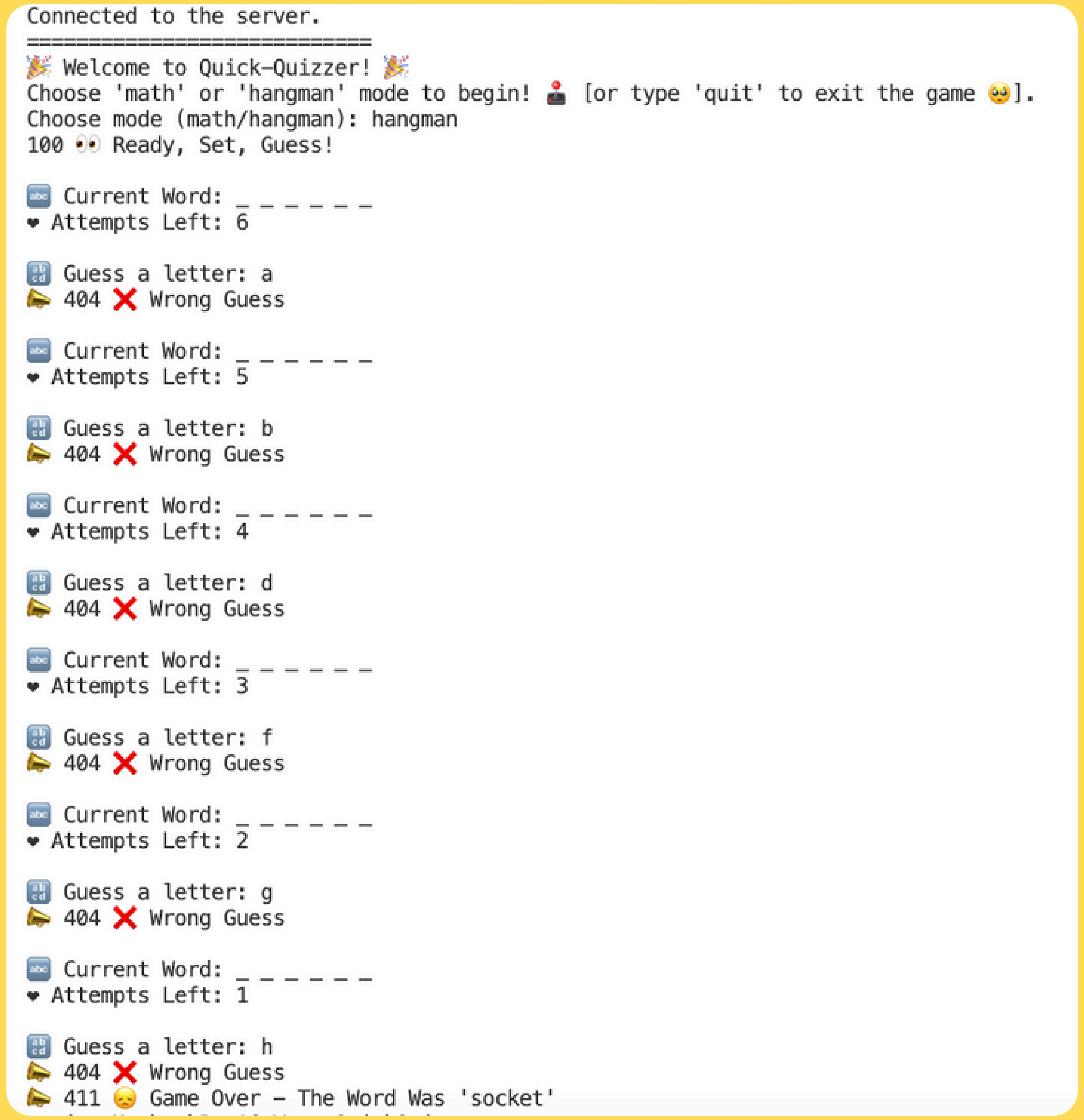
Hangman

Server



```
[2024-11-29 10:17:14] [Client 58121] Connected to ('127.0.0.1', 58121)
[2024-11-29 10:17:17] [Client 58121] Mode selected: hangman
[2024-11-29 10:17:17] [Client 58121] Starting Hangman with word: socket
[2024-11-29 10:17:29] [Client 58121] Guessed 'a', Attempts left: 5, Word: ______
[2024-11-29 10:17:31] [Client 58121] Guessed 'b', Attempts left: 4, Word: ______
[2024-11-29 10:17:35] [Client 58121] Guessed 'd', Attempts left: 3, Word: ______
[2024-11-29 10:17:37] [Client 58121] Guessed 'f', Attempts left: 2, Word: ______
[2024-11-29 10:17:42] [Client 58121] Guessed 'g', Attempts left: 1, Word: ______
[2024-11-29 10:17:47] [Client 58121] Guessed 'h', Attempts left: 0, Word: ______
[2024-11-29 10:17:47] [Client 58121] Game over. Player failed to guess the word.
```

Client - Lose



```
Connected to the server.
============================
🎉 Welcome to Quick-Quizzer! 🎉
Choose 'math' or 'hangman' mode to begin! 🕹 [or type 'quit' to exit the game 🥺].
Choose mode (math/hangman): hangman
100 👀 Ready, Set, Guess!

🔤 Current Word: _ _ _ _ _ _
❤ Attempts Left: 6

🔡 Guess a letter: a
📣 404 ❌ Wrong Guess

🔤 Current Word: _ _ _ _ _ _
❤ Attempts Left: 5

🔡 Guess a letter: b
📣 404 ❌ Wrong Guess

🔤 Current Word: _ _ _ _ _ _
❤ Attempts Left: 4

🔡 Guess a letter: d
📣 404 ❌ Wrong Guess

🔤 Current Word: _ _ _ _ _ _
❤ Attempts Left: 3

🔡 Guess a letter: f
📣 404 ❌ Wrong Guess

🔤 Current Word: _ _ _ _ _ _
❤ Attempts Left: 2

🔡 Guess a letter: g
📣 404 ❌ Wrong Guess

🔤 Current Word: _ _ _ _ _ _
❤ Attempts Left: 1

🔡 Guess a letter: h
📣 404 ❌ Wrong Guess
📣 411 😞 Game Over - The Word Was 'socket'
```

Challenges:

1. Designing an intuitive protocol. – TCP
2. Handling invalid inputs and edge cases.
3. Ensure smooth client-server interaction.
4. Ensure error handling for all cases.

GAME MODE 1

Conclusion

Math Quiz Mode

- 3 Levels
- Instant Feedback
- Score Updates
- Time Tracking
- Error Handling
- Completion Summary

GAME MODE 2

Conclusion

Hangman Mode

- Word Guessing
- Limited Attempts
- Instant Feedback
- Game State
- Progress Updates

Thank You

For your attention