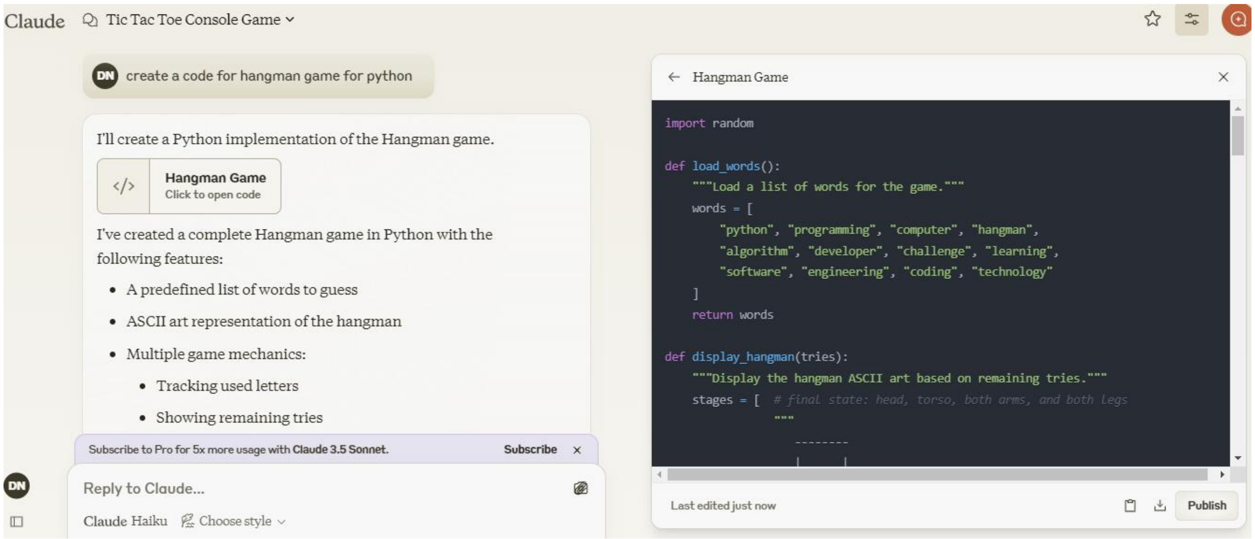
Python Programming using ClaudeAi

Code for Hangman game

05/12/2024

Create code:



create a code for hangman game for python

I'll create a Python implementation of the Hangman game.

Hangman Game
Click to open code

I've created a complete Hangman game in Python with the following features:

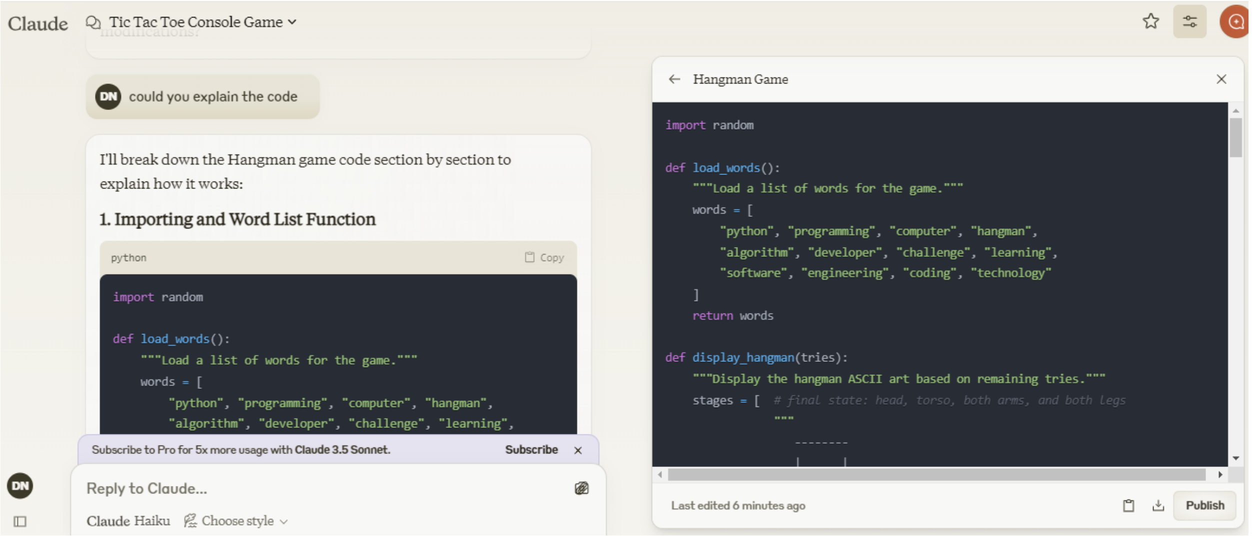
- A predefined list of words to guess
- ASCII art representation of the hangman
- Multiple game mechanics:
  - Tracking used letters
  - Showing remaining tries

```python
import random

def load_words():
    """Load a list of words for the game."""
    words = [
        "python", "programming", "computer", "hangman",
        "algorithm", "developer", "challenge", "learning",
        "software", "engineering", "coding", "technology"
    ]
    return words

def display_hangman(tries):
    """Display the hangman ASCII art based on remaining tries."""
    stages = [  # final state: head, torso, both arms, and both legs
                """
                   --------
                   |      |
```

Explain code:



could you explain the code

I'll break down the Hangman game code section by section to explain how it works:

1. Importing and Word List Function

```python
import random

def load_words():
    """Load a list of words for the game."""
    words = [
        "python", "programming", "computer", "hangman",
        "algorithm", "developer", "challenge", "learning",
```

```python
import random

def load_words():
    """Load a list of words for the game."""
    words = [
        "python", "programming", "computer", "hangman",
        "algorithm", "developer", "challenge", "learning",
        "software", "engineering", "coding", "technology"
    ]
    return words

def display_hangman(tries):
    """Display the hangman ASCII art based on remaining tries."""
    stages = [  # final state: head, torso, both arms, and both legs
                """
                   --------
                   |      |
```

1. Importing and Word List Function

python
Copy

```python
import random

def load_words():
    """Load a list of words for the game."""
    words = [
        "python", "programming", "computer", "hangman",
        "algorithm", "developer", "challenge", "learning",
        "software", "engineering", "coding", "technology"
    ]
    return words
```

- We import the `random` module to randomly select a word
- `load_words()` creates a list of predefined words to choose from
- `random.choice()` will later pick a random word from this list

2. Hangman ASCII Art Function

python
Copy

```python
def display_hangman(tries):
    """Display the hangman ASCII art based on remaining tries."""
```

- This function creates a visual representation of the hangman
- It has 7 different stages (0-6 tries left)
- Each stage shows more of the hangman being "drawn"
- As the player makes wrong guesses, the hangman becomes more complete
- When tries reach 0, the full hangman is displayed, indicating game over

3. Main Game Logic: `play_hangman()` Function

python
Copy

```python
def play_hangman():
    # Load words and choose a random word
    word_list = load_words()
    word = random.choice(word_list)
    word_letters = set(word)  # letters in the word
    alphabet = set('abcdefghijklmnopqrstuvwxyz')
    used_letters = set()  # what the user has guessed

    # Number of tries the player has
    tries = 6
```

- Randomly selects a word from the word list
- Creates sets to track:
    - `word_letters`: unique letters in the word
    - `alphabet`: all possible letters to guess
    - `used_letters`: letters the player has already guessed
- Sets initial number of tries to 6

Game Loop

python
Copy

```python
   # Game loop
    while len(word_letters) > 0 and tries > 0:
        # Letters used
        print("You have", tries, "tries left.")
        print("Used letters:", ' '.join(sorted(used_letters)))

        # What current word is (ie W - R D)
        word_list = [letter if letter in used_letters else '-'
for letter in word]
        print("Current word:", ' '.join(word_list))

        # Player input
        user_letter = input("Guess a letter: ").lower()
```

- The game continues while:

    - There are still letters to guess (`len(word_letters) > 0`)
    - Player has tries remaining (`tries > 0`)
- Displays:
    - Remaining tries
    - Letters already used
    - Current word progress (guessed letters revealed, others as '-')
- Prompts player to guess a letter

Input Handling

python
Copy

```python
       if user_letter in alphabet - used_letters:
            used_letters.add(user_letter)
            if user_letter in word_letters:
                word_letters.remove(user_letter)
            else:
                tries -= 1  # wrong guess, reduce tries
                print("Letter is not in the word.")

        elif user_letter in used_letters:
            print("You have already used that letter. Please try again.")

        else:
            print("Invalid character. Please enter a letter.")

        # Print hangman
        print(display_hangman(tries))
```

- Checks if the guessed letter is:
    - A valid, unused letter
    - If correct, removes it from `word_letters`
    - If incorrect, reduces tries
- Handles cases of:
    - Already used letters
    - Invalid input

- Displays current hangman stage

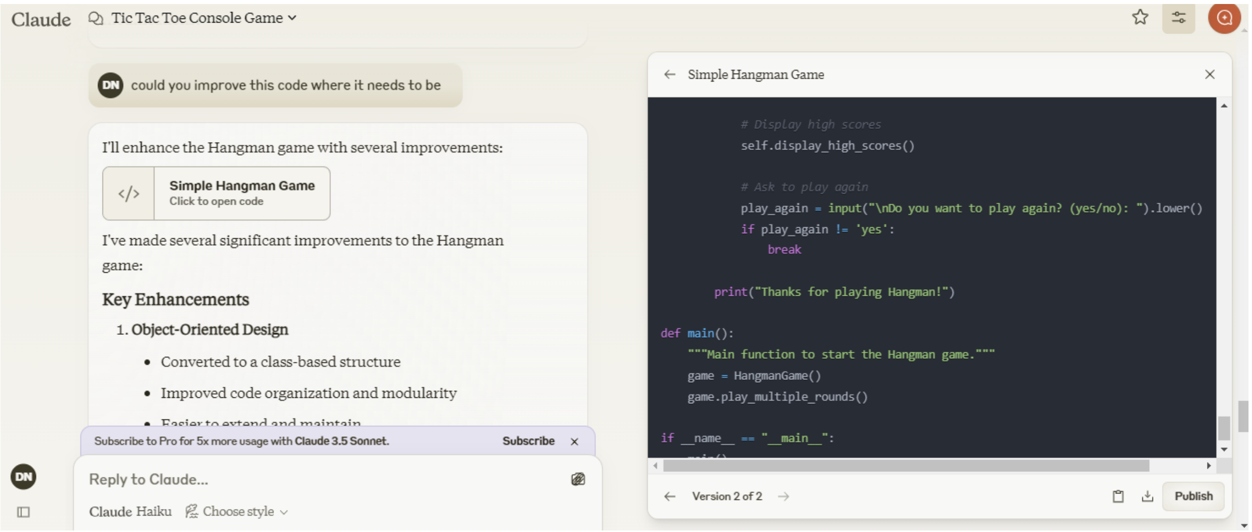
Game Ending

python
Copy

```python
# Game ended
if tries == 0:
    print(f"Sorry, you died. The word was {word}.")
else:
    print(f"Congratulations! You guessed the word {word}!!")
```

- If no tries left, player loses and word is revealed
- If all letters guessed, player wins

Improve Code:



DN could you improve this code where it needs to be

I'll enhance the Hangman game with several improvements:

I've made several significant improvements to the Hangman game:

Key Enhancements

1. **Object-Oriented Design**
   - Converted to a class-based structure
   - Improved code organization and modularity
   - Easier to extend and maintain

Subscribe to Pro for 5x more usage with **Claude 3.5 Sonnet.**

```python
            # Display high scores
            self.display_high_scores()

            # Ask to play again
            play_again = input("\nDo you want to play again? (yes/no): ").lower()
            if play_again != 'yes':
                break

        print("Thanks for playing Hangman!")

def main():
    """Main function to start the Hangman game."""
    game = HangmanGame()
    game.play_multiple_rounds()

if __name__ == "__main__":
```

```python
import random

# List of possible words for the game
words = ['python', 'hangman', 'computer', 'programming', 'developer',
'keyboard', 'challenge']

# Function to choose a random word from the list
def get_word():
    return random.choice(words)
```

```
# Function to display the current state of the word
def display_word(word, guessed_letters):
    return ''.join([letter if letter in guessed_letters else '_' for
letter in word])

# Function to check if the player has won
def check_win(word, guessed_letters):
    return all(letter in guessed_letters for letter in word)

# Main Hangman function
def hangman():
    word = get_word()
    guessed_letters = set()
    attempts = 0
    max_attempts = 6

    print("Welcome to Hangman!")

    while attempts < max_attempts:
        print(display_word(word, guessed_letters))

        # Ask for the player's guess
        guess = input("Guess a letter: ").lower()

        # Input validation
        if len(guess) != 1 or not guess.isalpha():
            print("Invalid input. Please enter a single letter.")
            continue

        if guess in guessed_letters:
            print(f"You already guessed '{guess}'. Try a different
letter.")
            continue

        # Add the guess to guessed_letters
        guessed_letters.add(guess)

        # Check if the guess is correct
        if guess not in word:
            attempts += 1
            print(f"Incorrect guess. You have {max_attempts -
attempts} attempts left.")
        else:
            print(f"Good guess! The letter '{guess}' is in the word.")

        # Check if the player has won
        if check_win(word, guessed_letters):
            print(display_word(word, guessed_letters))
            print("Congratulations! You guessed the word correctly!")
            break
    else:
        print(f"Game Over! The word was '{word}'.")

    # Ask if the player wants to play again
    play_again = input("Do you want to play again? (y/n): ").lower()
    if play_again == 'y':
        hangman()

# Start the game
hangman()
```

Improvements Without the Diagrams:

1. **No Hangman Drawing**: The game still works without the hangman diagram. The player only sees the current word and the number of remaining attempts.
2. **Input Validation**: Ensures only valid single letters are entered and that repeated guesses are prevented.
3. **Replay Option**: After finishing the game (either win or loss), the player can choose to play again.
4. **Win/Loss Logic**: Once the player either guesses all letters or runs out of attempts, the game will tell them the result and ask if they want to play again.