Hangman Game in Python

Introduction :

Hangman is a classic word-guessing game that has been enjoyedby players of all ages for generations. The game is simple yet captivating, requiring players to use their vocabulary and deduction skills to guess a hidden word one letter at a time. It is a perfect blend of fun and learning, making it a popular choice for educational purposes and casual entertainment.

**Objective of the Game :**

The primary goal of Hangman is to guess a hidden word correctly before the player runs out of allowed incorrect guesses. Each incorrect guess brings the player closer to completing a visual representation of a hangman. If the player completes the hangman before guessing the word, they lose the game.

**Basic Rules :**

 **Word Selection**: A word is chosen randomly from a predefined list. The word remains hidden, and its length is represented by a series of underscores (\_), with each underscore representing a letter in the word.

 **Guessing Letters**: Players guess one letter at a time. If the guessed letter is in the word, all occurrences of that letter are revealed in their correct positions. If the guessed letter is not in the word, it counts as an incorrect guess.

 **Incorrect Guesses**: Each incorrect guess brings the player closer to losing the game. Typically, a part of the hangman figure is drawn with each incorrect guess.

 **Winning and Losing**: The game is won if the player correctly guesses all the letters in the word before the hangman figure is fully drawn. The game is lost if the hangman figure is completed before the player can guess the word.

#### Enhancements in the Modern Hangman Game

To make the game more engaging and versatile, several enhancements can be added:

1. **Word Categories**: Players can choose from different categories such as animals, countries, and movies, adding variety and making the game more interesting.
2. **Difficulty Levels**: Players can select difficulty levels like easy, medium, or hard, which adjust the number of allowed incorrect guesses or the complexity of the words.
3. **Hangman Graphics**: Visual representation of the hangman using ASCII art or simple graphics adds a visual element, making the game more engaging.
4. **Hint System**: A hint system can provide clues about the hidden word at the cost of reducing the player's overall score, adding a strategic element to the game.
5. **Multiplayer Mode**: Enabling multiplayer mode allows players to take turns guessing the word, fostering a collaborative and competitive environment.

### 

#### 1. Word Categories

Create a dictionary with categories as keys and lists of words as values.

python

Copy code

word\_categories = {

"animals": ["elephant", "giraffe", "penguin", "kangaroo", "dolphin"],

"countries": ["brazil", "canada", "germany", "italy", "japan"],

"movies": ["inception", "gladiator", "titanic", "avatar", "matrix"]

}

#### 2. Hangman ASCII Art

Create a list of strings representing the hangman stages.

python

hangman\_stages = [

"""

-----

| |

|

|

|

|

=========

""",

"""

-----

| |

O |

|

|

|

=========

""",

"""

-----

| |

O |

| |

|

|

=========

""",

"""

-----

| |

O |

/| |

|

|

=========

""",

"""

-----

| |

O |

/|\\ |

|

|

=========

""",

"""

-----

| |

O |

/|\\ |

/ |

|

=========

""",

"""

-----

| |

O |

/|\\ |

/ \\ |

|

=========

"""

]

#### 3. Difficulty Levels

Define a dictionary with difficulty levels and corresponding maximum number of incorrect guesses.

difficulty\_levels = {

"easy": 10,

"medium": 7,

"hard": 5

}

#### 4. Functions for Game Logic

**Choosing a Category:**

def choose\_category():

print("Choose a category:")

for idx, category in enumerate(word\_categories.keys()):

print(f"{idx + 1}. {category}")

choice = int(choose\_input("\nEnter the number of your choice: ", range(1, len(word\_categories) + 1)))

return list(word\_categories.keys())[choice - 1]

**Getting User Input:**

def choose\_input(prompt, valid\_range):

while True:

try:

choice = int(input(prompt))

if choice in valid\_range:

return choice

else:

print(f"Invalid input! Please enter a number between {valid\_range[0]} and {valid\_range[-1]}.")

except ValueError:

print("Invalid input! Please enter a number.")

**Getting a Word:**

def get\_word(category):

return random.choice(word\_categories[category])

**Displaying Hangman:**

def display\_hangman(tries):

return hangman\_stages[tries]

#### 5. Main Game Logic

**Playing the Game:**

def play\_game():

category = choose\_category()

word = get\_word(category)

word\_letters = set(word)

guessed\_letters = set()

tries = 0

hint\_used = False

difficulty = input("Choose difficulty (easy/medium/hard): ").lower()

max\_tries = difficulty\_levels.get(difficulty, 7)

print(f"\nCategory: {category.capitalize()}\n")

print(display\_hangman(tries))

while tries < max\_tries and not word\_letters.issubset(guessed\_letters):

print(f"Guessed letters: {' '.join(sorted(guessed\_letters))}")

print("Word: " + ' '.join([letter if letter in guessed\_letters else '\_' for letter in word]))

guess = input("Guess a letter: ").lower()

if guess == "hint" and not hint\_used:

hint\_used = True

print(f"Hint: The word is related to {category}.")

continue

if guess in guessed\_letters:

print("You already guessed that letter.")

elif guess in word\_letters:

guessed\_letters.add(guess)

print("Good guess!")

else:

guessed\_letters.add(guess)

tries += 1

print("Incorrect guess.")

print(display\_hangman(tries))

if word\_letters.issubset(guessed\_letters):

print(f"Congratulations! You've guessed the word '{word}'!")

else:

print(f"Game over! The word was '{word}'.")

def main():

print("Welcome to Hangman!")

while True:

play\_game()

replay = input("Do you want to play again? (yes/no): ").lower()

if replay != 'yes':

break

print("Thank you for playing!")

if \_\_name\_\_ == "\_\_main\_\_":

main()

### Optional: Multiplayer Mode

To implement a multiplayer mode, you can allow players to take turns guessing letters and keep track of scores for each player. Here is a brief idea:

**Multiplayer Function:**

def play\_multiplayer\_game():

category = choose\_category()

word = get\_word(category)

word\_letters = set(word)

guessed\_letters = set()

tries = 0

hint\_used = False

difficulty = input("Choose difficulty (easy/medium/hard): ").lower()

max\_tries = difficulty\_levels.get(difficulty, 7)

print(f"\nCategory: {category.capitalize()}\n")

print(display\_hangman(tries))

players = int(input("Enter number of players: "))

player\_scores = {f"Player {i+1}": 0 for i in range(players)}

current\_player = 0

while tries < max\_tries and not word\_letters.issubset(guessed\_letters):

current\_player = current\_player % players

player\_name = f"Player {current\_player + 1}"

print(f"\n{player\_name}'s turn")

print(f"Guessed letters: {' '.join(sorted(guessed\_letters))}")

print("Word: " + ' '.join([letter if letter in guessed\_letters else '\_' for letter in word]))

guess = input("Guess a letter: ").lower()

if guess == "hint" and not hint\_used:

hint\_used = True

print(f"Hint: The word is related to {category}.")

continue

if guess in guessed\_letters:

print("You already guessed that letter.")

elif guess in word\_letters:

guessed\_letters.add(guess)

player\_scores[player\_name] += 1

print("Good guess!")

else:

guessed\_letters.add(guess)

tries += 1

print("Incorrect guess.")

current\_player += 1

print(display\_hangman(tries))

if word\_letters.issubset(guessed\_letters):

print(f"Congratulations! You've guessed the word '{word}'!")

else:

print(f"Game over! The word was '{word}'.")

print("\nFinal Scores:")

for player, score in player\_scores.items():

print(f"{player}: {score}")

def main():

print("Welcome to Hangman!")

while True:

mode = input("Choose mode (single/multiplayer): ").lower()

if mode == "single":

play\_game()

elif mode == "multiplayer":

play\_multiplayer\_game()

else:

print("Invalid mode.")

continue

replay = input("Do you want to play again? (yes/no): ").lower()

if replay != 'yes':

break

print("Thank you for playing!")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Overview:**

The Hangman Game is a classic word-guessing game where players try to discover a hidden word by suggesting letters within a limited number of guesses. This project enhances the traditional game with features such as word categories, ASCII art for visual representation, difficulty levels, and a hint system. Additionally, a multiplayer mode can be added as an optional feature.

**Features:**

1. **Word Categories**: Players can choose from different categories (e.g., animals, countries, movies) before starting the game.
2. **Hangman Graphics**: Visual representation of the hangman using ASCII art.
3. **Difficulty Levels**: Players can choose from easy, medium, or hard difficulty levels.
4. **Hint System**: Provides players with a clue about the hidden word at the cost of reducing their overall score.
5. **Multiplayer Mode (Optional)**: Enables multiple players to take turns guessing the word.

### Conclusion :

This enhanced Hangman game provides a fun and interactive experience for players. It includes multiple features that increase its complexity and enjoyment. The code is structured to be easily understandable and extendable, allowing further enhancements such as networked multiplayer mode or a graphical user interface (GUI) in the future.