**Title:** Python programming

**Final Project:** Hangman Game

**Name**: Angirekula sivarani

**Roll No:**213H1A0503

**Branch:** Computer Science Engineering

Let's develop a Hangman game in Python with the suggested enhancements:

1. **\*Word Categories\*:** We'll provide categories such as animals, countries, and movies, and let the player choose one.

**2. \*Hangman Graphics\*:** We'll use ASCII art to represent the hangman.

**3. \*Difficulty Levels**\*: We'll have different difficulty levels that control the number of allowed incorrect guesses.

**4. \*Multiplayer Mode\* (optional):** Players can take turns guessing the word.

**5. \*Hint System\*:** Players can use a hint at the cost of reducing their score.

**Here's the complete implementation:**

**```python**

**import random**

# Define word categories

categories = {

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

"countries": ["argentina", "belgium", "canada", "denmark", "egypt"],

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

}

# Hangman stages

hangman\_ stages = [

"""

-----

| |

|

|

|

|

=========

""",

"""

-----

| |

O |

|

|

|

=========

""",

"""

-----

| |

O |

| |

|

|

=========

""",

"""

-----

| |

O |

/| |

|

|

=========

""",

"""

-----

| |

O |

/|\\ |

|

|

=========

""",

"""

-----

| |

O |

/|\\ |

/ |

|

=========

""",

"""

-----

| |

O |

/|\\ |

/ \\ |

|

=========

"""

]

# Difficulty levels

difficulty\_ levels = {

"easy": 10,

"medium": 7,

"hard": 5

}

# Hint system

def provide\_ hint(word):

hints = {

"elephant": "It's the largest land animal.",

"giraffe": "It has a very long neck.",

"kangaroo": "It hops and carries its baby in a pouch.",

"dolphin": "A very intelligent marine mammal.",

"penguin": "A flightless bird found in Antarctica.",

"argentina": "A country famous for tango dance.",

"belgium": "Known for chocolates and waffles.",

"canada": "The second-largest country in the world.",

"denmark": "A Scandinavian country with many islands.",

"egypt": "Home to the ancient pyramids.",

"inception": "A movie about dreams within dreams.",

"gladiator": "A movie about a Roman general turned gladiator.",

"titanic": "A movie about a famous shipwreck.",

"avatar": "A movie set on the planet Pandora.",

"casablanca": "A classic romance set during WWII."

}

return hints. Get (word, "No hint available.")

def get\_ random\_ word(category):

return random. Choice (categories[category])

def display\_ hangman(stage):

print (hangman\_ stages[stage])

def play\_ game ():

print ("Welcome to Hangman!")

category = input (f" Choose a category ({', '.join (categories. Keys ())}): "). Lower ()

if category not in categories:

print ("Invalid category. Exiting game.")

return

difficulty = input ("Choose difficulty level (easy, medium, hard): "). Lower ()

if difficulty not in difficulty\_ levels:

print ("Invalid difficulty level. Exiting game.")

return

word = get\_ random\_ word(category)

guessed\_ word = ["\_"] \* len (word)

attempts = difficulty\_ levels[difficulty]

used\_ hints = False

wrong\_ guesses = 0

while wrong\_ guesses < attempts:

print ("\n Current word: ", " ". join (guessed\_ word))

display\_ hangman (wrong\_ guesses)

guess = input ("Guess a letter (or type 'hint' for a hint): "). Lower ()

if guess == "hint":

if used\_ hints:

print ("You have already used your hint!")

else:

hint = provide\_ hint(word)

print (f" Hint: {hint}")

used\_ hints = True

attempts -= 1 # Using a hint costs an attempt

continue

if len (guess) != 1 or not guess. Isalpha():

print ("Please guess a single letter.")

continue

**Conclusion and Explanation:**

Here's the continuation and conclusion of the Hangman game implementation and explanation:

python

used\_ hints = True

attempts -= 1 # Using a hint costs an attempt

continue

if len(guess)! = 1 or not guess. Isalpha ():

print ("Please guess a single letter.")

continue

if guess in guessed\_ word:

print ("You've already guessed that letter.")

continue

if guess in word:

for i, letter in enumerate(word):

if letter == guess:

guessed\_ word[i] = guess

if "\_" not in guessed\_ word:

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

break

else:

wrong\_ guesses += 1

print (f" Incorrect guess. You have {attempts - wrong\_ guesses} attempts left.")

if wrong\_ guesses == attempts:

display \_hangman (wrong\_ guesses)

print (f" Sorry, you've been hanged! The word was: {word}")

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

play\_ game ()

### Conclusion and Full Explanation

1. \*Word and Guess Initialization\*:

python

word = get\_ random\_ word(category)

guessed \_word = ["\_"] \*len(word)

attempts = difficulty \_levels[difficulty]

used \_hints = False

wrong\_ guesses = 0

The chosen word is fetched from the selected category. Guessed \_word is initialized as a list of underscores, representing the hidden letters. attempts are set based on the selected difficulty level. used\_ hints tracks if a hint has been used, and wrong \_guesses keeps count of incorrect guesses.

2. \*Game Loop\*:

python

while wrong\_ guesses < attempts:

print ("\n Current word: ", " ".join(guessed\_ word))

display\_ hangman (wrong\_ guesses)

guess = input ("Guess a letter (or type 'hint' for a hint): "). lower()

The loop runs until the player either guesses the word correctly or exhausts their attempts.

3. \*Hint Handling\*:

python

if guess == "hint":

if used\_ hints:

print ("You have already used your hint!")

else:

hint = provide\_ hint(word)

print (f" Hint: {hint}")

used\_ hints = True

attempts -= 1 # Using a hint costs an attempt

continue

If the player requests a hint, it checks if they have already used one. If not, it provides the hint and deducts one attempt.

4. \*Invalid Guess Handling\*:

python

if len(guess) != 1 or not guess. Isalpha():

print ("Please guess a single letter.")

continue

if guess in guessed\_ word:

print("You've already guessed that letter.")

continue

The game checks if the guess is a single letter and if it has been guessed before.

5. \*Correct Guess Handling\*:

python

if guess in word:

for i, letter in enumerate(word):

if letter == guess:

guessed\_ word[i] = guess

if "\_" not in guessed\_ word:

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

break

If the guessed letter is in the word, it updates guessed\_ word. If the player guesses all letters, they win.

6. \*Incorrect Guess Handling\*:

python

else:

wrong\_ guesses += 1

print (f" Incorrect guess. You have {attempts - wrong\_ guesses} attempts left.")

If the guessed letter is incorrect, it increments wrong\_ guesses.

7. \*End Game\*:

python

if wrong\_ guesses == attempts:

display\_ hangman (wrong\_ guesses)

print (f "Sorry, you've been hanged! The word was: {word}")

If the player runs out of attempts, the game ends, displaying the full hangman and revealing the word.

8. \*Main Function\*:

python

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

play\_ game ()

This ensures the game starts when the script is run.

This enhanced Hangman game now includes word categories, ASCII art for the hangman stages, difficulty levels, a hint system, and proper handling of user input, provide