06-14-2023

Hangman Game

kavya Borra

# Create and import image and Word file

To bootstrapping hangman code, we need word category dictionary and hangman image to display for incorrect guess.

From hangman\_pkg, created hangman image list and words list dictionary which is further imported into main python file.

Prompt user’s selection from Category:

### Here, it displays Category Dictionary keys for user to select category, it converts category dictionary into list.

### **list(category.keys())**

### Checks if user’s selection is valid else breaks out from loop, prompts user to enter category selection again.

# A screenshot of a computer Description automatically generated

# Using Random () CATEGORY:

User’s selected category is stored in variable, pass that variable as key in category dictionary.

With, random choice function, pass category [input category] as parameter and assign to variable.

Note: Category is dictionary and pass user’s selection as key and that user’s key should exist in category

chosen\_category = random.choice(category[input\_category]

# variables [counts] to keep track to hangman TRIES:

# total\_chances = 7 (hangman tries)

# Wrong\_count = 0 (increments while wrong letter is gussed

# WHEN user’s input\_LETTER, already exists BLANKS:

### If condition to check user’s input letter already exists in blanks, then it must break out of that if loop and display print statement to display status of blanks.

# A screen shot of a computer Description automatically generated with medium confidence

# Replace blanks with gussed\_LETTER, IF gussed right:

### If condition to check user’s input letter (input\_letter) is in randomized variable of category

### Iterate through len of randomized variable(input\_catergory)

### Further check if index of input\_category == input\_letter [User’s guessed]

### Replace, the corresponding blanks with the correct guessed letter.

### Here it divided into three-part, first part retains substring until index of input\_letter + input\_letter+ retains substring until end of index.

### **blanks = blanks[:index] + input\_letter + blanks[index + 1:]**

# Incorrect Guess, increment count of hangman tries:

### If user guessed is incorrect, then total count must decrement (def =7)

### Then for each incorrect guess, hangman\_display [wrong\_count] must be printed.

A screen shot of a computer

Description automatically generated with medium confidence

# Future Enhancement

### Looking to implement Difficulty mode in play, for each mode hangman tries should be decremented.

### Implement HINT WORD for total\_count >= 4