Technical description of the problem

Extended requirements implemented in the game:  
1. Output with correct guesses in relative position (e.g. O \_ \_ O \_ \_ ).  
2. User is informed how many occurrences of a guess are there in the word. This  
counts as one guess. (e.g. Correct! The word contains 2 letters O)  
3. User is informed if the entered letter has been guessed before.  
This does not count as a guess.  
4. There are 4 categories of words in this game that the user can choose.

The user selects the word category. The category is a list of words. The random.choice() function is used to select a random word from the list. The user is informed how many letters the selected word has. This is also how many guesses/tries the user has. Valid input is either one character or a string of the same length as the word. Invalid input does not count as a guess. A non-alphabetic character does count as a guess, but user is informed that the word contains only letters. A letter that has been guessed before doesn’t count as a guess and the user is informed.

We check if a one letter guess is correct by first adding this guess to the all\_guessed list. Because a string is iterable, we can loop through the letters in the selected word and check if the letter is in the all\_guessed list. If the letter is in the word we add to a string the letter in the relative position and if not we add an underscore. This string is called word\_after\_guess and is what the user sees after a guess. Word\_after\_guess is then converted into an alphabetic only string,namely status\_word, by using the same method in a for loop as above. This conversion is necessary to check if status\_word == word.

If len(guess) == len(word) then we check if guess=word.

We keep the user informed how many tries are left. We do so by subtracting 1 from the num\_of\_guesses each time a valid guess is made. If a guess occurs more than one time in a word, then it still counts as one guess.

We store the correct and incorrect guesses in a list without repetitions. When the number of guesses == 0 then we present the user with the correct and incorrect guesses. The user has a last attempt to guess the word. He has to input the full, correct word. After that, the user is informed if his final guess was correct and we output the word, as well as the lists with the correct and incorrect guesses. This is the end of the game.

How to play the hangman game

First select a category of words. From this category a word will be randomly selected. You will be informed how many letters the word has. You have as many guesses as there are letters in the word. You can only guess one single letter or the whole word at a time. After you’ve used up all your guesses you will be presented with the correct and incorrect guesses you have made. Based on this information you have to guess the word. If your final guess is false you will see what was the word.

Pseudocode

1. INITIALIZE

1.1 Categories:

animal\_list

fruit\_list

country\_list

city\_list\_UK

1.2 num\_of\_guesses = len(word)

all\_guessed = [ ]

correct\_guessed = [ ]

incorrect\_guessed = [ ]

word = random from category list

word\_after\_guess = “”

num\_of\_guesses = len(word)

guessed = False

2. INPUT

2.1 category = “select category from categories”

2.2 guess = “guess a letter or the whole word”

3. CALCULATE

3.1 if len(guess) == 1

num\_of\_guesses -= 1

add guess to all\_guessed

for letter in word:

if letter in all\_guessed:

word\_after\_guess += letter

else:

word\_after\_guess += “\_”

return word\_after\_guess

4. OUTPUT

4.1 print len(word), num\_of\_guesses

4.2 while guessed is False:

4.1.2 if num\_of\_guesses > 0

if len(guess) == len(word):

num\_guesses -= 1

if guess == word:

print “Congratulations”

guessed = True

else:

print “Wrong”

elif len(guess) == 1:

print word\_after\_guess

else:

print ”Invalid input”

4.1.3 elif num\_of\_guesses = 0:

print “Guess the word based on this information”

print all\_guessed, correct\_guessed, incorrect\_guessed

if guess == word:

print “Congratulations”

else:

print “Wrong”

guessed = True

Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Input** | **Output** | **Comments** |
| 1. | A, E, I, B, R, P, O, L, K  Guess: Liverpool | All guessed letters: A, E, I, B, R, P, O, L, K Correctly guessed letters: E, I, R, P, O, L Incorrectly guessed letters: A, B, K  Congratulations! You guessed the word on your last try | word = LIVERPOOL  As expected. |
| 2. | 12, @, #, E, R, T, Y, U,  Guess: jaguar | All guessed letters: E, R, T, Y, U Correctly guessed letters: E, U Incorrectly guessed letters: R, T, Y  Game over! The word was MOUSE  You have guessed 2 letters correctly: E, U You have guessed 3 letters incorrectly: R, T, Y | word = MOUSE  User was informed that this inputs are not letters: 12, @, #  These inputs weren’t added to the all\_guessed list, because of this.  These did count as a guess.  As expected. |
| 3. | a, n, m, c, h, e, s, t, e, r  Guess: ‘r’ was the last guess | Correct! The word contains the letter R  Congratulations! You guessed the word in 9 tries | word = MANCHESTER  User isn’t informed which letters were guessed correctly and which not when the game ended. This is because he already guessed the word.  As expected. |
| 4. | a, er, e, r, p, l  Guess: ‘l’ was the last guess | Correct! The word contains the letter L  Congratulations! You guessed the word in 5 tries | word = APPLE  User was informed of invalid input on guess: ‘er’  This didn’t count as a guess.  As expected. |
| 5. | A, E, R, T, P  Guess: china | All guessed letters: A, E, R, T, P Correctly guessed letters: A Incorrectly guessed letters: E, R, T, P  Guess the word: china  Congratulations! You guessed the word on your last try | word = CHINA  As expected. |
| 6. | A, D, O  Guess: owl | All guessed letters: A, D, O  Correctly guessed letters: O  Incorrectly guessed letters: A, D  Guess the word: owl  Congratulations! You guessed the word on your last try | word = owl  As expected. |
| 7. | A, E, R, T, N, M  Guess: gepart | All guessed letters: A, E, R, T, N, M Correctly guessed letters: A, R Incorrectly guessed letters: E, T, N, M  Guess the word: gepart  Game over! The word was LIZARD  You have guessed 2 letters correctly: A, R You have guessed 4 letters incorrectly: E, T, N, M | Word = LIZARD  As expected. |
| 8. | Guess: plum | Guess a letter or the whole word: plum  Congratulations! You guessed the word on your first try! | Word = PLUM  As expected. |
| 9. | r, a, antilope, antel0pe, as, p, e, t, c, l  Guess: antelope | All guessed letters: R, A, P, C, E  Correctly guessed letters: A, P, E  Incorrectly guessed letters: R, C  Congratulations! You guessed the word on your last try | word = ANTELOPE  User was informed, that ‘as’ and ‘antil0pe’ is an invalid input.’ |
| 10. | 3, -, ], s, a,  Guess: snake | All guessed letters: S, A  Correctly guessed letters: S  Incorrectly guessed letters: A  Game over! The word was HORSE  You have guessed 1 letter correctly: S  You have guessed 1 letter incorrectly: A | word: horse  As expected |

# 201307211 Szafarczyk\_Robert-CA05.py

# 11/17

# This is a hangman game. The user selects a category and from this category

# (a list) a random word is selected. User is informed how many letters the

# word has and that's how many tries to guess the word the user has.

# Repetitions do not count as a guess, invalid input does not count as a

# guess (e.g. 'ae'). Non-letters do count as a guess, strings with the same

# length as the word do count as a guess. The user is presented with a

# string with of underscores. Each underscore represents each letter from

# word. If a letter is guessed correctly the underscore is changed to this

# letter. If user has used up his guesses, then the program prints out the

# correctly and incorrectly guessed letters. Based on this information the

# user has to guess the whole word. After this guess the game ends.

# Extended requirements in the game:

# 1) output with correct guesses in relative position, e.g. F \_ I \_ \_

# 2) we tell how many occurrences of a guess are there in the word. This

# counts as one guess.

# 3) we tell the user if he entered a letter that has been guessed before.

# This does not count as a guess.

# 4) there are 4 categories of words in this game that the user can choose

def main():

def option\_AB():

print()

print()

print("Only option C available at this time")

def option\_Exit():

print()

print()

print("Goodbye!")

def option\_C():

def hangman():

import random

animal\_list = ['lion', 'bat', 'anaconda', 'salmon', 'whale',

'grasshopper', 'alligator', 'rat', 'bear',

'frog', 'elephant', 'cat', 'dog', 'giraffe',

'eagle', 'lizard', 'earthworm', 'tiger', 'dolphin',

'mouse', 'owl', 'horse', 'donkey', 'antelope',

'cobra', 'duck', 'emu', 'fox', 'jaguar']

fruit\_list = ['apple', 'banana', 'strawberry', 'blackberry',

'orange', 'grapefruit', 'mango', 'peach',

'lime', 'cherry', 'date', 'pineapple', 'papaya',

'raspberry', 'honeyberry', 'tangerine',

'coconut', 'nectarine', 'passionfruit', 'plum',

'watermelon', 'currant']

country\_list = ['England', 'Poland', 'Germany', 'Brazil',

'Portugal', 'France', 'Australia', 'China',

'Japan', 'India', 'Canada', 'Egypt', 'Argentina',

'Russia', 'Austria', 'Spain', 'Columbia',

'Chechia', 'Hungary', 'Denmark', 'Italy',

'Ireland', 'Turkey', 'Afghanistan', 'Iraq',

'Mongolia', 'Thailand', 'Chile', 'Peru', 'Kenya',

'Algeria', 'Ghana']

city\_list\_UK = ['Liverpool', 'Manchester', 'London', 'Plymouth',

'Newcastle', 'Lancaster', 'Birmingham',

'Leeds', 'Sheffield', 'Chester', 'Swansea',

'Oxford', 'Edinburgh', 'Glasgow', 'Colchester',

'Norwich', 'Exeter', 'Brighton', 'York', 'Cardiff',

'Aberdeen']

def random\_word(category):

return random.choice(category).upper()

def check\_letter(guess, word, all\_guessed):

# this string will be the output to the user

word\_with\_underscores = ""

num\_of\_matched\_letters = 0

# EXTENSION

# loops through every letter in word

# checks if letter from word is in list with all our guesses

for letter in word:

# ads letter in place of the letter in word

if letter in all\_guessed:

word\_with\_underscores += " " + letter + " "

# ads underscore in place of letter

else:

word\_with\_underscores += " \_ "

# EXTENSION

# counts how many times our guess occurs in word

if letter == guess:

num\_of\_matched\_letters += 1

if num\_of\_matched\_letters > 1:

print()

print("Correct! The word contains {} letters {}".

format(num\_of\_matched\_letters, guess))

print()

elif num\_of\_matched\_letters == 1:

print()

print("Correct! The word contains the letter {}".

format(guess))

print()

else:

if guess.isalpha():

print()

print("Sorry. The letter {} is not in the word".

format(guess))

print()

else:

# case where user's guess isn't a letter.

# Does count as guess

print()

print("Sorry. {} is not a letter.".

format(guess))

print()

return word\_with\_underscores

def convert\_to\_word(word\_with\_underscores, word):

status\_word = ""

for letter in word:

if letter in word\_with\_underscores:

status\_word += letter

return status\_word

print()

print()

print("Welcome to hangman!")

print()

print("Categories of words:")

print()

print("A. Animals")

print("B. Fruits")

print("C. Countries")

print("D. Cities in the UK")

print()

category = input("Select category A, B, C or D:")

category = category.upper()

# CATEGORY EXTENSION

while category not in ("A", "B", "C", "D"):

print()

print("Wrong input!")

category = input("Select category A, B, C or D: ")

category = category.upper()

# selecting random word from chosen category list

if category == "A":

word = random\_word(animal\_list)

word\_info = "Category: Animals\n" \

"The word has {} letters".format(len(word))

elif category == "B":

word = random\_word(fruit\_list)

word\_info = "Category: Fruits\n" \

"The word has {} letters".format(len(word))

elif category == "C":

word = random\_word(country\_list)

word\_info = "Category: Country\n" \

"The word has {} letters".format(len(word))

elif category == "D":

word = random\_word(city\_list\_UK)

word\_info = "Category: Cities in the UK\n" \

"The word has {} letters".format(len(word))

# initialising variables

num\_of\_guesses = len(word)

all\_guessed = []

correct\_guessed = []

correct\_guessed\_distinct = list(set(correct\_guessed))

incorrect\_guessed = []

incorrect\_guessed\_distinct = list(set(incorrect\_guessed))

guessed = False

print()

print("Let's start!")

print()

print(word\_info)

print("You have {} guesses to guess the word".format(len(word)))

while guessed is False:

if num\_of\_guesses > 0:

print()

guess = input("Guess a letter or the whole word: ")

print()

guess = guess.upper()

# EXTENSION

# informing user if he guessed the letter already.

# Does not count as guess

if guess in all\_guessed:

print("You already guessed {}".format(guess))

print("You still have {} guesses left".

format(num\_of\_guesses))

print()

print(word\_info)

print()

print(word\_after\_guess)

# user tries to guess whole word

elif len(guess) == len(word):

if guess.isalpha():

if guess == word:

if num\_of\_guesses == len(word):

print("Congratulations! You guessed "

"the word on your first try!")

guessed = True

else:

num\_of\_guesses -= 1

print("Congratulations! "

"You guessed the word in {} tries".

format(len(word) - num\_of\_guesses))

guessed = True

else:

num\_of\_guesses -= 1

print()

print("Sorry, this is not the word")

print()

print(word\_info)

print()

print("You have {} guesses left".

format(num\_of\_guesses))

else:

num\_of\_guesses -= 1

print()

print("Wrong input")

print("The word contains only letters")

# user guesses 1 letter

elif len(guess) == 1:

num\_of\_guesses -= 1

# this list contains all alpha guesses

if guess.isalpha():

all\_guessed.append(guess)

# function to check if guess in word

word\_after\_guess = check\_letter(guess, word,

all\_guessed)

# remove underscores and spaces

status\_word = convert\_to\_word(word\_after\_guess, word)

if status\_word == word:

print("Congratulations! "

"You guessed the word in {} tries".

format(len(word) - num\_of\_guesses))

guessed = True

else:

print(word\_after\_guess)

print()

print(word\_info)

print()

if num\_of\_guesses > 1:

print("You have {} guesses left".format

(num\_of\_guesses))

if num\_of\_guesses == 1:

print("You have 1 guess left.")

print()

if guess.isalpha():

if guess in word\_after\_guess:

correct\_guessed\_distinct.append(guess)

else:

incorrect\_guessed\_distinct.append(guess)

# if user's guess is not one single character or the same

# amount of characters as the word

else:

print()

print("Invalid entry")

print()

if len(all\_guessed) > 0:

print(word\_after\_guess)

print()

print(word\_info)

print()

print("This didn't count as a guess")

print("You still have {} guesses left".

format(num\_of\_guesses))

print()

# last guess. We output the correct and incorrect guesses

elif num\_of\_guesses == 0:

print("No guesses left!")

print()

print("Guess the word based on this information:")

print("All guessed letters: {}".

format(", ".join(all\_guessed)))

print("Correctly guessed letters: {}".

format(", ".join(correct\_guessed\_distinct)))

print("Incorrectly guessed letters: {}".

format(", ".join(incorrect\_guessed\_distinct)))

print()

guess = input("Guess the word: ")

guess = guess.upper()

if guess == word:

print()

print("Congratulations! "

"You guessed the word on your last try")

guessed = True

else:

print()

print()

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

print()

if len(correct\_guessed\_distinct) != 1:

print("You have guessed {} letters correctly: {}".

format(len(correct\_guessed\_distinct),

", ".join(correct\_guessed\_distinct)))

else:

print("You have guessed 1 letter correctly: {}".

format(", ".join(correct\_guessed\_distinct)))

if len(incorrect\_guessed\_distinct) != 1:

print("You have guessed {} letters incorrectly: "

"{}".format(len(incorrect\_guessed\_distinct),

", ".

join(

incorrect\_guessed\_distinct)))

else:

print("You have guessed 1 letter incorrectly: {}".

format(", ".

join(incorrect\_guessed\_distinct)))

guessed = True

# user didn't guess the word

else:

print()

print()

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

print()

if len(correct\_guessed\_distinct) != 1:

print("You have guessed {} letters correctly: {}".

format(len(correct\_guessed\_distinct),

", ".join(correct\_guessed\_distinct)))

else:

print("You have guessed 1 letter correctly: {}".

format(", ".join(correct\_guessed\_distinct)))

if len(incorrect\_guessed\_distinct) != 1:

print("You have guessed {} letters incorrectly: {}".

format(len(incorrect\_guessed\_distinct),

", ".join(incorrect\_guessed\_distinct)))

else:

print("You have guessed 1 letter incorrectly: {}".

format(", ".join(incorrect\_guessed\_distinct)))

guessed = True

hangman()

option\_end\_C = ""

while option\_end\_C != "X":

print()

print()

print()

print("You're still in Games")

print()

print("A. Play again")

print("X. Main Menu")

print()

option\_end\_C = input("Enter option A or X: ")

option\_end\_C = option\_end\_C.upper()

while option\_end\_C not in ("A", "X"):

print()

print("Invalid input")

option\_end\_C = input("Enter option A or X: ")

option\_end\_C = option\_end\_C.upper()

if option\_end\_C == "A":

hangman()

option = ""

while option != "X":

print()

print()

print("Main Menu")

print()

print("A. Numbers")

print("B. Strings")

print("C. Games")

print("X. Exit")

print()

option = input("Enter option A,B,C or X: ")

option = option.upper()

while option not in ("A", "B", "C", "X"):

print()

print("Invalid input")

option = input("Enter option A,B,C or X: ")

option = option.upper()

if option == "A" or option == "B":

option\_AB()

elif option == "C":

option\_C()

elif option == "X":

option\_Exit()

main()