NEW HORIZON COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

"HANGMAN GAME"

A MINI PROJECT REPORT

Submitted by

CHRISEL FERNANDES [1NH18IS026]

Under the guidance of,

Mrs.Rafega

Sr. Assistant Professor, ISE, NHCE

In the partial fulfillment of the requirements in the 4th Semester of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

FOR

COURSE NAME :MINI PROJECT

COURSE CODE :19ISE49

NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

CERTIFICATE

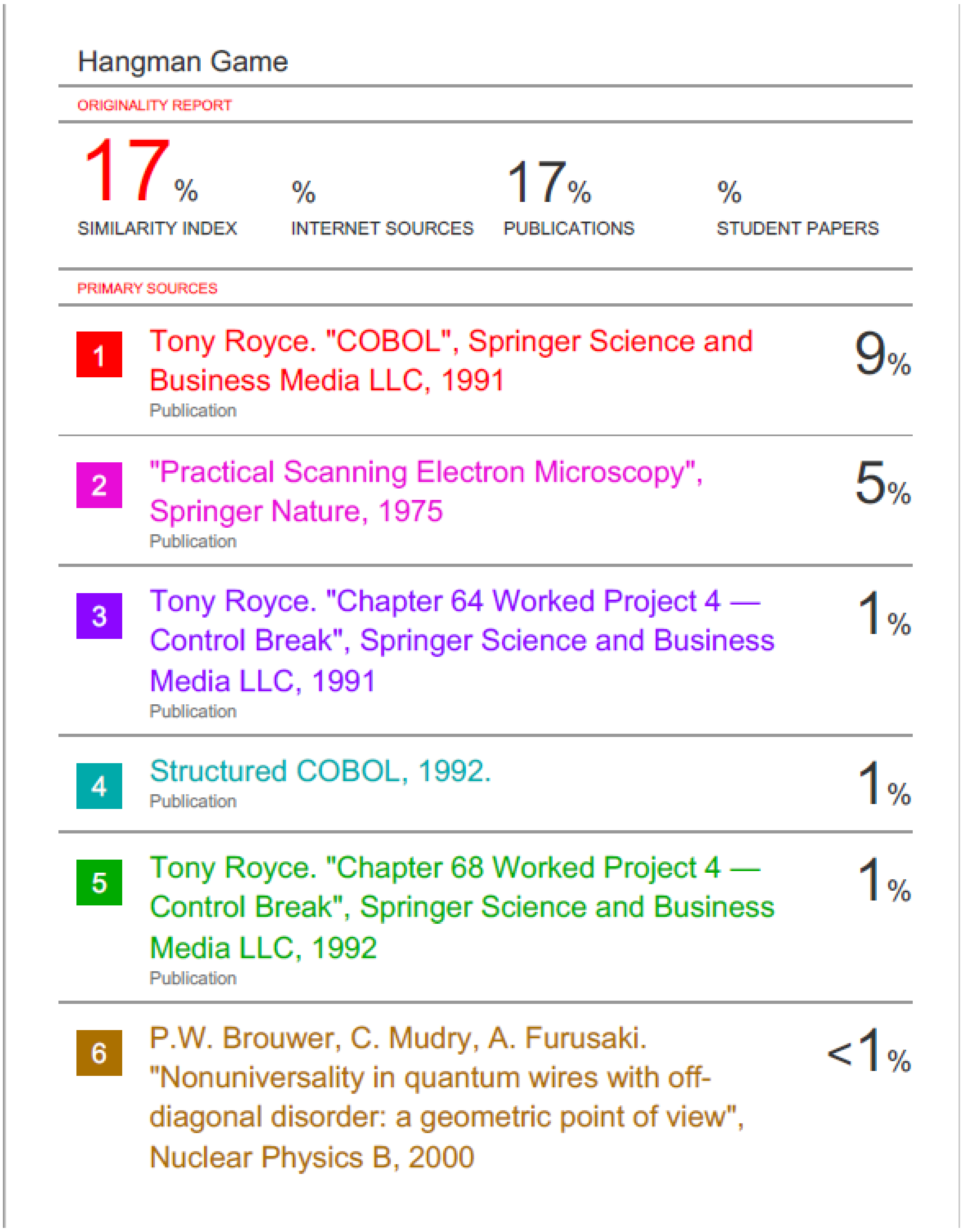
Certified that the project work entitled Hangman Game carried out by Mr. CHRISEL FERNANDES, bearing USN 1NH18IS026, a bonafide student of 4th semester in partial fulfillment for the award of Bachelor of Engineering in Information Science & Engineering of the Visveswaraiah Technological University, Belagavi during the year 2019-20. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated. The project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the said Degree.

Name & Signature of Guide	**Name & Signature of HOD**	**Name & Signature of Principal**
Mrs.Rafega	Dr.Anandhii R J	Dr. Manjunatha

Examiners :

	Name	**Signature**
1.	……………………………………………….	………………………………..
2.	…………………………………………………	…………………………………

PLAGARISM CERTIFICATE



Hangman Game

ORIGINALITY REPORT

17%	%	17%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Tony Royce. "COBOL", Springer Science and Business Media LLC, 1991 Publication	9%
2	"Practical Scanning Electron Microscopy", Springer Nature, 1975 Publication	5%
3	Tony Royce. "Chapter 64 Worked Project 4 — Control Break", Springer Science and Business Media LLC, 1991 Publication	1%
4	Structured COBOL, 1992. Publication	1%
5	Tony Royce. "Chapter 68 Worked Project 4 — Control Break", Springer Science and Business Media LLC, 1992 Publication	1%
6	P.W. Brouwer, C. Mudry, A. Furusaki. "Nonuniversality in quantum wires with off-diagonal disorder: a geometric point of view", Nuclear Physics B, 2000	<1%

ABSTRACT

Word games are spoken or board games often designed to test ability with language or to explore its properties. Hangman is a paper and pencil guessing game for two or more players. One player thinks of a word and the other tries to guess it by suggesting the letters. If the guessing player suggests a letter which occurs in the word, the program writes it in all its correct positions. If the suggested letter does not occur in the word, the other player draws one element of the hangman diagram as a tally mark.
The game is over when: the guessing player completes the word, or guesses the whole word correctly. This project aims to implement a computer-based version of the classic 2-player Hangman Game using python 3 and will use Object Oriented Programming concept where the code will choose a random word from a list of words which have been already fed into it. The word has to be guessed by the player(i.e., the user).The benefit of this program is primarily that it allows someone to play without requiring another person to participate.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, who's constant guidance and encouragement crowned our efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman of New Horizon Educational Institutions for providing necessary infrastructure and creating good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha**, Principal NHCE, for his constant support and encouragement.

I am grateful to **Dr. Prashanth C.S.R**, Dean Academics, for his unfailing encouragement and suggestions, given to me in the course of my project work.

I would also like to thank **Dr. R.J. Anandhi,** Professor and Head, Department of Information Science and Engineering, for her constant support.

I express my gratitude to **Mrs. Rafega,** Senior Assistant Professor, my project guide, for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

CHRISEL FERNANDES

1NH18IS026

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: - INTRODUCTION

- Word games (also called word game puzzles) are spoken or board games often designed to test ability with language or to explore its properties
- Word games are generally used as a source of entertainment, but can additionally serve an educational purpose. Young children can enjoy playing games such as Hangman, while naturally developing important language skills like spelling. Researchers have found that adults who regularly solved crossword puzzles, which require familiarity with a larger vocabulary, had better brain function later in life
- Students can enjoy playing Hangman game in the leisure time and can relax and enjoy the game while at the same time increasing their vocabulary and keep their minds active
- Hangman is a paper and pencil guessing game for two or more players. One player thinks of a word and the other tries to guess it by suggesting the letters.
- Here's how it works:
  1. Player One picks a secret word and draws a line for each letter in it (you will use an underscore to represent each line).
  2. Player Two tries to guess the word one letter at a time.
  3. If Player Two guesses a letter correctly, Player One replaces the corresponding underscore with the correct letter. In this version of the game, if a letter appears twice in a word, you have to guess it twice.

     OR

     If Player Two guesses incorrectly, Player One draws a body part of a hanged stick figure (starting with the head).
  4. If Player Two completes the word before the drawing of the hangman is complete, they win. If not, they lose.

1.1 Objective

- To develop a computer-based version of the classic 2-player Hangman Game using Python 3.

1.2 Brief Description

- Hangman is a quick and easy game for at least two people that requires nothing more than paper, a pencil, and the ability to spell. One player, the "host," makes up a secret word, while the other player tries to guess the word by asking what letters it contains. However, every wrong guess brings them one step closer to losing.

- One player thinks of a word or phrase; the others try to guess what it is one letter at a time. The player draws a number of dashes equivalent to the number of letters in the word. If a guessing player suggests a letter that occurs in the word, the other player fills in the blanks with that letter in the right places. If the word does not contain the suggested letter, the other player draws one element of a hangman's gallows. As the game progresses, a segment of the gallows and of a victim is added for every suggested letter not in the word. The number of incorrect guesses before the game ends is up to the players, but completing a character in a noose provides a minimum of six wrong answers until the game ends. The first player to guess the correct answer thinks of the word for the next game.
- This project implements a computer-based version of the 2-player Hangman Game. The human player competes against the computer to guess a word the program has selected at random.
- The benefit of this program is primarily that it allows someone to play without requiring another person to participate.

1.3 Project Features

The application's purpose is to simulate the word game "Hangman." The major features are:

- Obtain a hidden word from the program.
- Manage the game play: getting the player guess, evaluating it, and updating the characters in the word(phrase).
- Determining if the game is won or lost and display the necessary output.

CHAPTER 2: - PROJECT PLAN AND DESIGN METHOD

2.1 Resources Planned

- Hardware Requirements: -
    - A system based on 64-bit architecture that supports Intel® Streaming SIMD Extensions 4.2 (Intel® SSE 4.2) instructions or compatible non-Intel® processors
    - 2 GB of free disk space for all product features and architectures

- Software Requirements: -
    - Windows* 7, 8.x, Windows® 10 (SP1 for Windows 7 is required for Intel® Advanced Vector Extensions use)
    - Windows Server* 2008 R2 SP1 and SP2, 2012, 2016
    - Python 3

2.2 Software Used

- Python 3:

  Python was developed by Guido van Rossum in early 1990's and its latest version is 3.7.1, we can simply call it as Python3. Python 3.0 was released in 2008. and is interpreted language i.e. it's not compiled and the interpreter will check the code line by line. is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a must for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

  Python is a multi-paradigm programming language. Meaning, it supports different programming approach. One of the popular approaches to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP). An object has two characteristics: attributes and behaviour.

The concept of OOP in Python focuses on creating reusable code. This concept is also known as DRY (Don't Repeat Yourself). In Python, the concept of OOP follows some basic principles:

- Inheritance: A process of using details from a new class without modifying existing class.
- Encapsulation: Hiding the private details of a class from other objects.
- Polymorphism: A concept of using common operation in different ways for different data input.

- <u>IDLE (Integrated Development and Learning Environment):</u>

  IDLE is an integrated development environment (IDE) for Python. The Python installer for Windows contains the IDLE module by default. IDLE can be used to execute a single statement just like Python Shell and also to create, modify and execute Python scripts. IDLE provides a fully-featured text editor to create Python scripts that includes features like syntax highlighting, autocompletion and smart indent. It also has a debugger with stepping and breakpoints features.

  Its main features are:

  - Multi-window text editor with syntax highlighting, autocompletion, smart indent and other.
  - Python shell with syntax highlighting.
  - Integrated debugger with stepping, persistent breakpoints, and call stack visibility.

CHAPTER 3: -CONCEPT USED

This project uses OOPs (Object Oriented Programming) concept in Python for its implementation.

It also uses sequence datatypes such as lists and dictionaries.

Also, the random

3.1 OOPs Concept

- **Class:** A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- **Data member:** A class variable or instance variable that holds data associated with a class and its objects.
- **Method:** A special kind of function that is defined in a class definition.
- **Instantiation:** The creation of an instance of a class (object)
- **Object:** A unique instance of a class. An object comprises both data members (class variables and instance variables) and methods. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.
- **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class.
- **Class variable:** A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- **Creation of Class:** A class is created with the keyword *class* and then writing the classname. The simplest form of class definition looks like this:

```
class ClassName:
                                    <statement-1>
                                      .
                                      .
                                      .
                                    <statement-N>
```

  Class definitions, like function definitions must be executed before they have any effect. A class creates a new local namespace where all its attributes are defined. Attributes may be data or functions. There are also special attributes in it that begins with double underscores. For example, __doc__ gives us the docstring of that class. As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

- **Constructors in Python:** Class functions that begins with double underscore are called special functions as they have special meaning. Of one particular interest is the __init__() function. This special function gets called whenever a new object of that class is instantiated.

  This type of function is also called constructors in Object Oriented Programming (OOP). We normally use it to initialize all the variables. A constructor is a special method that is used to initialize the instance variables of a class. In the constructor, we create the instance variables and initialize them with some starting values. The first parameter of the constructor will be "self‟ variable that contains the memory address of the instance. The method __init__() name has two underscores before and after. This indicates that this method is internally defined and we cannot call this method explicitly.

  The methods that act on instances (or objects) of a class are called instance methods. Instance methods use "self‟ as the first parameter that refers to the location of the instance in the memory.

  To create an instance, the following syntax is used:

  instancename = Classname( )

- **Self variable:** "self" is a default variable that contains the memory address of the instance of the current class. When an instance to the class is created, the instance name contains the memory location of the instance. This memory location is internally passed to "self‟. Since "self‟ knows the memory address of the instance, it can refer to all the members of the instance.

  We use "self‟ in two ways:

  - ✓ **"self" is used as first parameter in the constructor:** In this case, "self‟ can be used to refer to the instance variables inside the constructor.
  - ✓ **"self‟ is used as first parameter in the instance methods:** If the instance method wants to act on the instance variables, it should know the memory location of the instance variables. That memory location is by default available to the instance method through "self‟.

- **Types of Constructor:**

  1. **Non-Parameterized Constructor**: In this case no other arguments will be provided except of the default 'self' argument. The first parameter of the constructor will be "self‟ variable that contains the memory address of the instance.

2. **Parameterized Constructor**: In this case, the constructor will have some parameters in addition to “self‟.

- **Types of Variables:**

  The variables which are written inside a class are of 2 types:

  1. **Instance Variables:** Instance variables are the variables whose separate copy is created in every instance.

     Instance variables are defined and initialized using a constructor with “self‟ parameter. Also, to access instance variables, we need instance methods with “self‟ as first parameter. It is possible that the instance methods may have other parameters in addition to the “self‟ parameter.
  2. **Class Variables or Static Variables:** Class variables are the variables whose single copy is available to all the instances of the class. If we modify the copy of class variable in an instance, it will modify all the copies in the other instances.

3.2 Sequence Datatypes Used

In the project code lists and dictionaries are used.

1. **Lists:** A list is similar to an array that consists of a group of elements or items. Just like an array, a list can store elements. But there is one major difference between an array and a list. An array can store only one type of elements whereas a list can store different types of elements. Hence lists are more versatile and useful than an array.
   - **Creating a List:** Creating a list is as simple as putting different comma-separated values between square brackets.

     We can create empty list without any elements by simply writing empty square brackets as: var_name = [ ]

     We can create a list by embedding the elements inside a pair of square braces []. The elements in the list should be separated by a comma (,).
   - **Accessing Values in list:** To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. To view the elements of a list as a whole, we can simply pass the list name to print function.

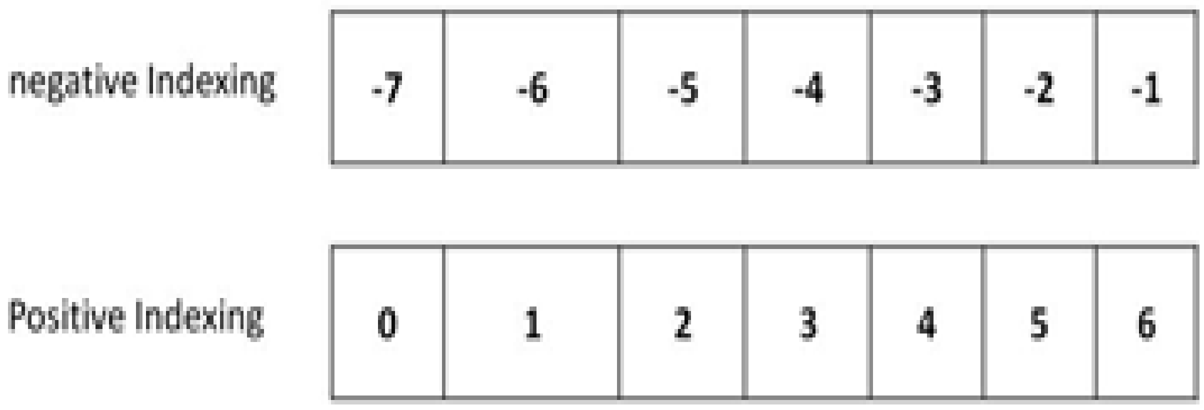


Fig. 3.2.1

- **Looping on lists:** We can also display list by using for loop or while loop. The len( ) function can be used to know the numbers of elements in the list.
- **Updating and deleting lists:** Lists are mutable. It means we can modify the contents of a list.
  We can append, update or delete the elements of a list depending upon our requirements.
  Appending an element means adding an element at the end of the list. To, append a new element to the list, we should use the append() method. Updating an element means changing the value of the element in the list. This can be done by accessing the specific element using indexing or slicing and assigning a new value.
  Deleting an element from the list can be done using *'del'* statement. The *del* statement takes the position number of the element to be deleted.
  If we want to delete entire list, we can give the statement: del lst
- **Membership in Lists:** We can check if an element is a member of a list by using "in" and "not in" operator. If the element is a member of the list, then "in" operator returns **True** otherwise returns **False**. If the element is not in the list, then "not in" operator returns **True** otherwise returns **False**.
- **List Method:**

Method	**Description**
lst.index(x)	Returns the first occurrence of x in the list.
lst.append(x)	Appends x at the end of the list.
lst.insert(i,x)	Inserts x to the list in the position specified by i.
lst.copy()	Copies all the list elements into a new list and returns it.

lst.extend(lst2)	Appends lst2 to list.
lst.count(x)	Returns number of occurrences of x in the list.
lst.remove(x)	Removes x from the list.
lst.pop()	Removes the ending element from the list.
lst.sort()	Sorts the elements of list into ascending order.
lst.reverse()	Reverses the sequence of elements in the list.
lst.clear()	Deletes all elements from the list.
max(lst)	Returns biggest element in the list.
min(lst)	Returns smallest element in the list.

2. **Dictionary:** A dictionary represents a group of elements arranged in the form of key-value pairs. The first element is considered as "key" and the immediate next element is taken as its "value". The key and its value are separated by a colon (:).
   All the key-value pairs in a dictionary are inserted in curly braces { }.
   - To access the elements of a dictionary, indexing or slicing should not be used.
   - To access the value associated with a key, we can mention the key name inside the square braces.
   - To know how many key-value pairs are there in a dictionary, the len( ) function can be used.
   - To insert a new key-value pair into an existing dictionary, mention the key and assign a value to it.
   - To delete a key-value pair from the dictionary, the *del* statement can be used.
   - To Test whether a "key" is available in a dictionary or not, the "in" and "not in" operators can be used. These operators return either True or False.

   - We can use any datatypes for value. But keys should obey the rules:
     - ✓ Keys should be unique. It means, duplicate keys are not allowd. If we enter same key again, the old key will be overwritten and only the new key will be available.
     - ✓ Keys should be immutable type. For example, we can use a number, string or tuples as keys since they are immutable. We cannot use lists or dictionaries as keys. If they are used as keys, we will get "TypeError".

➢ Dictionary Methods:

Method	Description
d.clear()	Removes all key-value pairs from dictionary “d”.
d2=d.copy()	Copies all elements from “d” into a new dictionary d2.
d.fromkeys(s [,v])	Create a new dictionary with keys from sequence “s” and values all set to “v”.
d.get(k [,v])	Returns the value associated with key “k”. If key is not found, it returns “v”.
d.items()	Returns an object that contains key-value pairs of “d”. The pairs are stored as tuples in the object.
d.keys()	Returns a sequence of keys from the dictionary “d”.
d.values()	Returns a sequence of values from the dictionary “d”.
d.update(x)	Adds all elements from dictionary “x” to “d”.
d.pop(k [,v])	Removes the key „k‟ and its value from “d” and returns the value. If key is not found, then the value „v‟ is returned. If key is not found and „v‟ is not mentioned then “KeyError” is raised.
d.setdefault(k[,v])	If key ‘k” is found, its value is returned. If key is not found, then the k, v pair is stored into the dictionary “d”.

3.3 Random Module

The random module gives access to various useful functions and one of them being able to generate random numbers, which is randint().

CHAPTER 4: -DESIGN METHOD

4.1 Algorithm

Step 1: Start.
Step 2: Select a random word.
Step 3: Show appropriate number of blanks for the word selected.
Step 4: Ask user to guess a letter. If the word contains the letter entered, remove the blanks and enter the letter at the appropriate indexes.
Step 5: If the letter entered by the user is not in the word, draw one element of the hangman stick figure.
Step 6: Repeat steps 4,5 and 6 till the user guesses the word completely OR till the hangman figure is completed.
Step 7: Display appropriate message whether the user has guessed the word or not.
Step 8: Ask user if he/she wants to play again or not.
Step 8: End

4.2 Flowchart

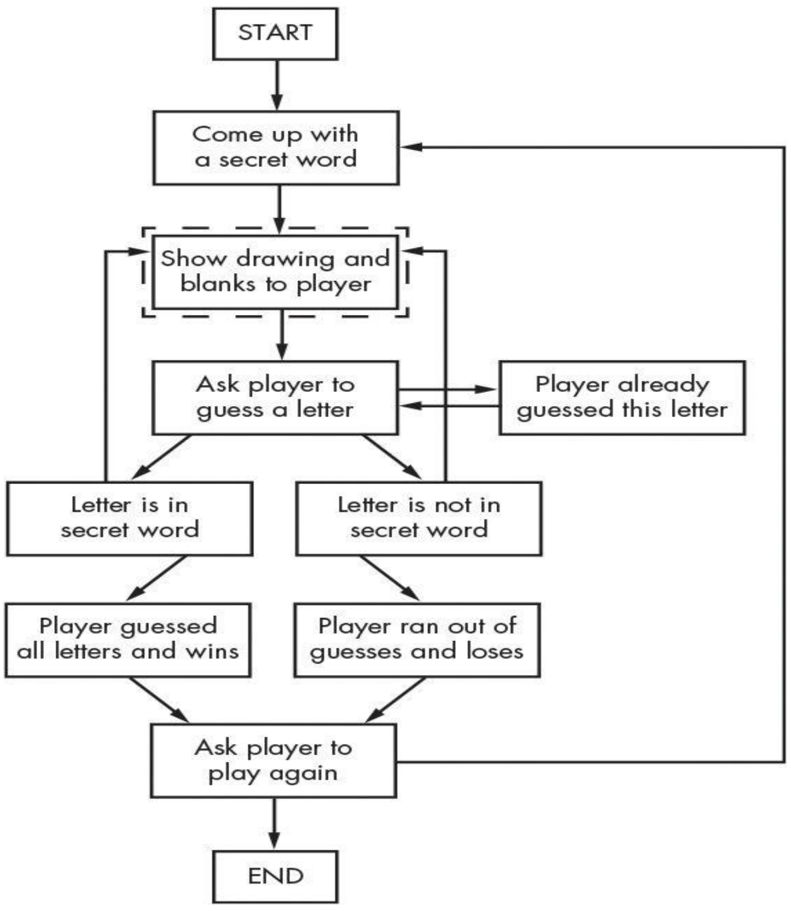

START
Come up with a secret word
Show drawing and blanks to player
Ask player to guess a letter
Player already guessed this letter
Letter is in secret word
Letter is not in secret word
Player guessed all letters and wins
Player ran out of guesses and loses
Ask player to play again
END

Fig 4.2.1 Flowchart

4.3 Code

```
import random
HANGMAN_PICS = ['''
   +---+
       |
       |
       |
      ===''', '''
   +---+
   O   |
       |
       |
      ===''', '''
   +---+
   O   |
   |   |
       |
      ===''', '''
   +---+
   O   |
  /|   |
       |
      ===''', '''
   +---+
   O   |
  /|\  |
       |
      ===''', '''
   +---+
   O   |
  /|\  |
  /    |
      ===''', '''
   +---+
   O   |
  /|\  |
  / \  |
      ===''']

words ='ant baboon badger bat bear beaver camel cat clam cobra cougar coyote crow
deer dog donkey duck eagle ferret fox frog goat goose hawk lion lizard llama mole
monkey moose mouse mule newt otter owl panda parrot pigeon python rabbit ram rat
```

```
raven rhino salmon seal shark sheep skunk sloth snake spider stork swan tiger toad
trout turkey turtle weasel whale wolf wombat zebra'.split()

def getRandomWord(wordList):
  # This function returns a random string from the passed list of strings.
  wordIndex = random.randint(0, len(wordList) - 1)
  return wordList[wordIndex]

def displayBoard(missedLetters, correctLetters, secretWord):
  print(HANGMAN_PICS[len(missedLetters)])
  print()

  print('Missed letters:', end=' ')
  for letter in missedLetters:
    print(letter, end=' ')
  print()

  blanks = '_' * len(secretWord)

  for i in range(len(secretWord)): # Replace blanks with correctly guessed letters.
    if secretWord[i] in correctLetters:
      blanks = blanks[:i] + secretWord[i] + blanks[i+1:]

  for letter in blanks: # Show the secret word with spaces in between each letter.
    print(letter, end=' ')
  print()

def getGuess(alreadyGuessed):
   # Returns the letter the player entered. This function makes sure the player entered
a single letter and not something else.
   while True:
     print('Guess a letter.')
     guess = input()
     guess = guess.lower()
     if len(guess) != 1:
       print('Please enter a single letter.')
     elif guess in alreadyGuessed:
       print('You have already guessed that letter. Choose again.')
     elif guess not in 'abcdefghijklmnopqrstuvwxyz':
       print('Please enter a LETTER.')
     else:
       return guess
```

```
def playAgain():
    # This function returns True if the player wants to play again; otherwise, it returns
False.
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')


print('H A N G M A N')
missedLetters = ''
correctLetters = ''
secretWord = getRandomWord(words)
gameIsDone = False

while True:
    displayBoard(missedLetters, correctLetters, secretWord)

    # Let the player enter a letter.
    guess = getGuess(missedLetters + correctLetters)

    if guess in secretWord:
        correctLetters = correctLetters + guess

        # Check if the player has won.
        foundAllLetters = True
        for i in range(len(secretWord)):
            if secretWord[i] not in correctLetters:
                foundAllLetters = False
                break
        if foundAllLetters:
            print('Yes! The secret word is "' + secretWord +'"! You have won!')
            gameIsDone = True
    else:
        missedLetters = missedLetters + guess

        # Check if player has guessed too many times and lost.
        if len(missedLetters) == len(HANGMAN_PICS) - 1:
            displayBoard(missedLetters, correctLetters, secretWord)
            print('You have run out of guesses!\nAfter ' +str(len(missedLetters)) + ' missed
guesses and ' +str(len(correctLetters)) + ' correct guesses,the word was "' + secretWord
+ '"')
            gameIsDone = True

    # Ask the player if they want to play again (but only if the game is done).
```

```
if gameIsDone:
    if playAgain():
        missedLetters = ''
        correctLetters = ''
        gameIsDone = False
        secretWord = getRandomWord(words)
    else:
        break
```

CHAPTER 5: - TEST SCENARIOS AND RESULT

5.1 Test Scenarios

The program can have varying outputs depending on how the user plays the game.

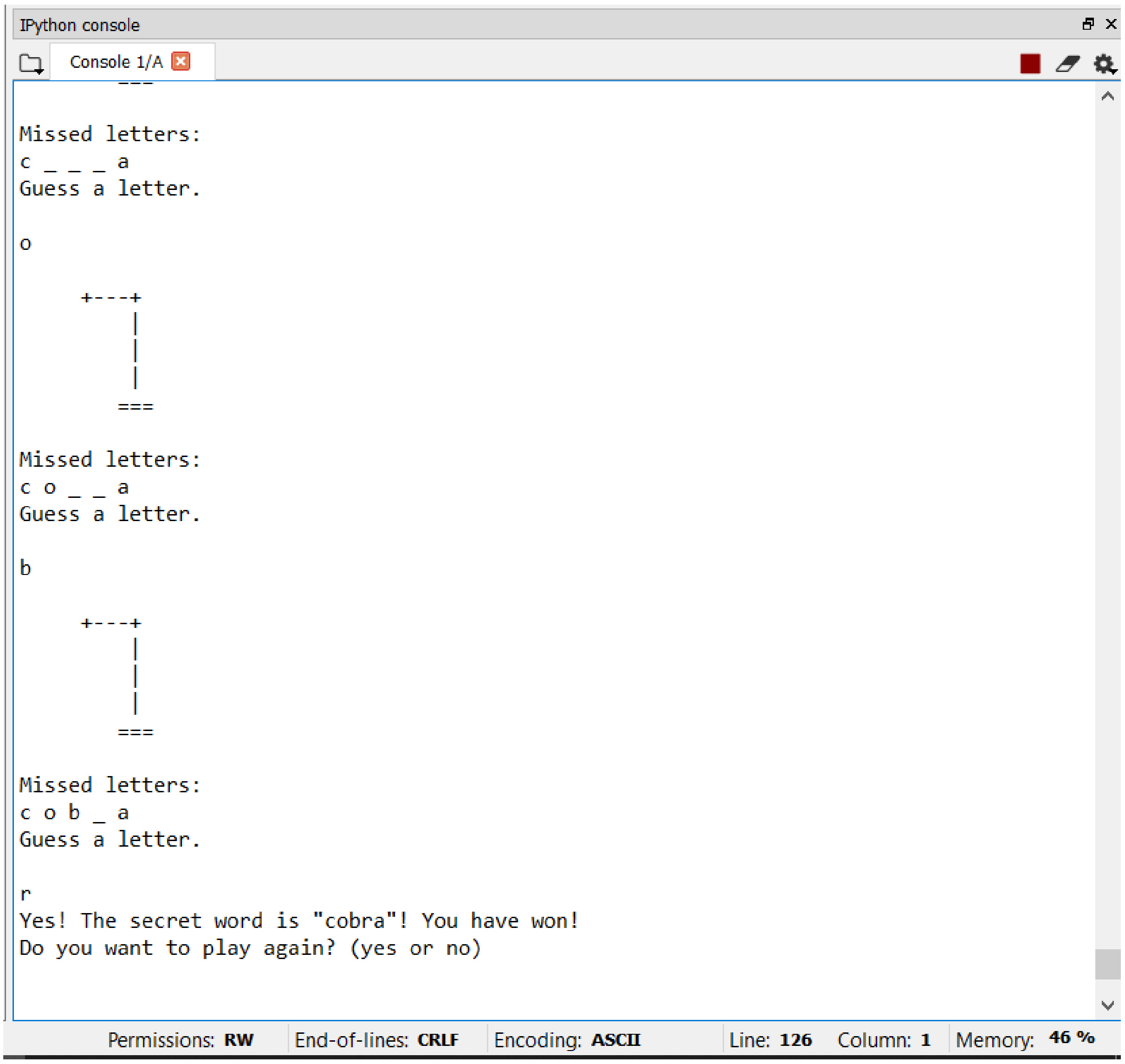
There are primarily two main scenarios:

- The player wins the game
    - The player makes all correct guesses and wins the game.
    - The player makes a combination of correct and wrong guesses and wins the game.
- The player loses the game
    - The player makes all wrong guesses and loses the game.
    - The player makes a combination of correct and wrong guesses and yet loses the game.

5.2 Result

The program was tested for the above test case scenarios and the result expected as per the test case scenarios was successfully achieved. The result for the various test case scenarios is given as follows:

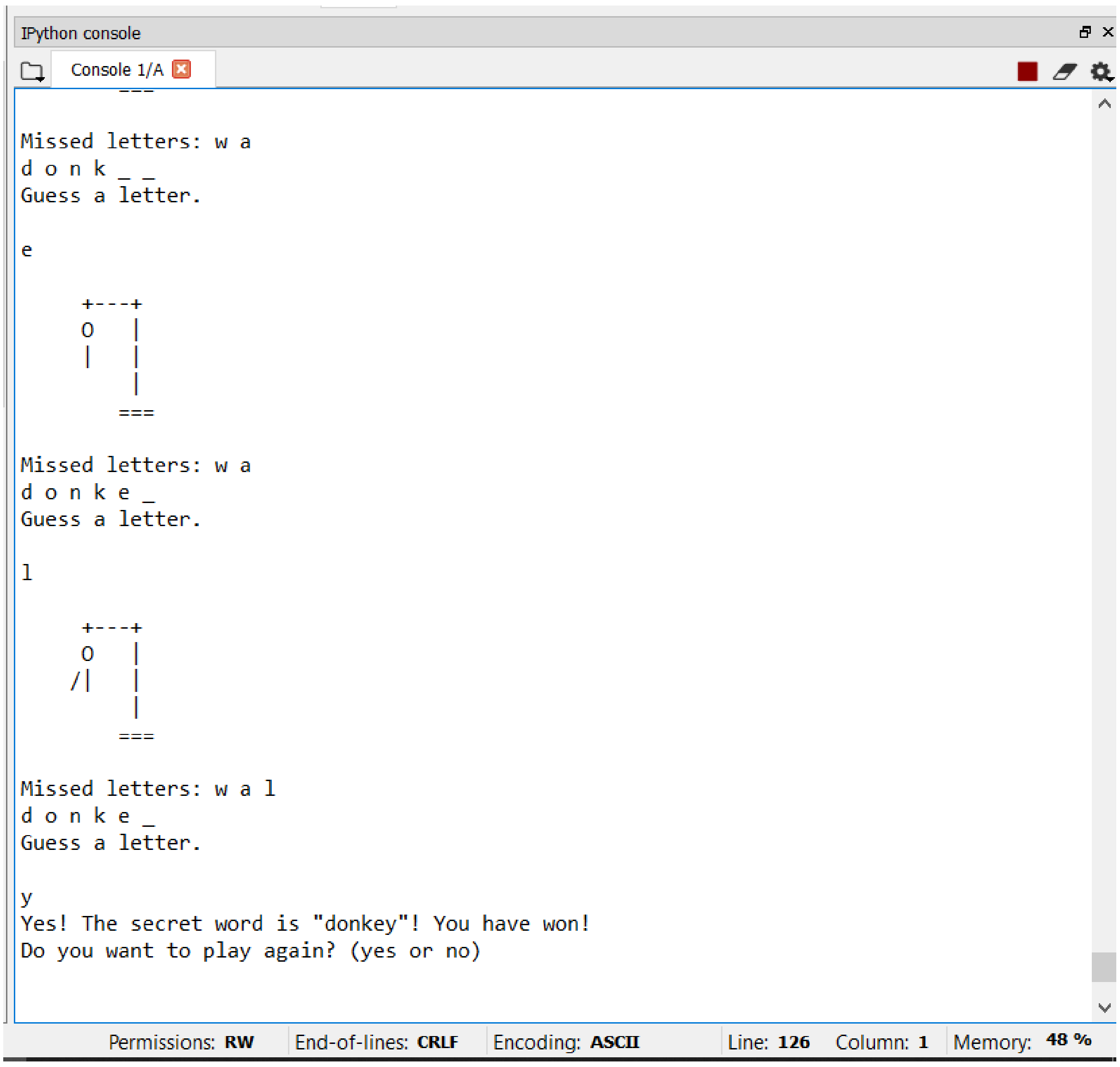
- The player wins the game.



```
        ===

Missed letters:
c _ _ _ a
Guess a letter.

o

    +---+
        |
        |
        |
       ===

Missed letters:
c o _ _ a
Guess a letter.

b

    +---+
        |
        |
        |
       ===

Missed letters:
c o b _ a
Guess a letter.

r
Yes! The secret word is "cobra"! You have won!
Do you want to play again? (yes or no)
```

Permissions: **RW** End-of-lines: **CRLF** Encoding: **ASCII** Line: **126** Column: **1** Memory: **46 %**

Fig 5.2.1. The player wins the game making all correct choices

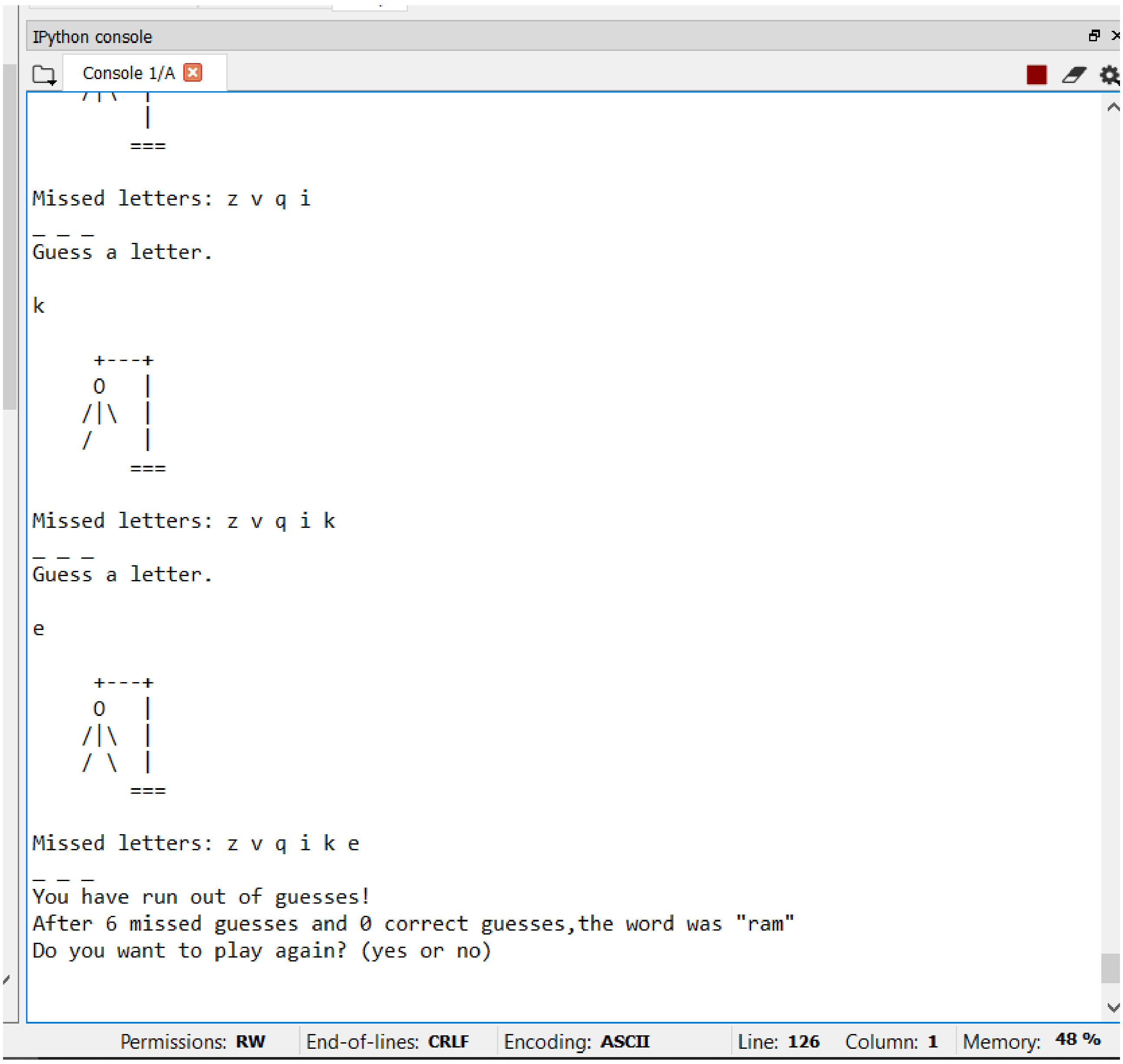


```
      ===

Missed letters: w a
d o n k _ _
Guess a letter.

e


    +---+
    O   |
    |   |
        |
       ===

Missed letters: w a
d o n k e _
Guess a letter.

l


    +---+
    O   |
   /|   |
        |
       ===

Missed letters: w a l
d o n k e _
Guess a letter.

y
Yes! The secret word is "donkey"! You have won!
Do you want to play again? (yes or no)
```

Permissions: RW   End-of-lines: CRLF   Encoding: ASCII   Line: 126   Column: 1   Memory: 48 %

Fig 5.2.2 The player wins the game making a combination of correct and wrong guesses.

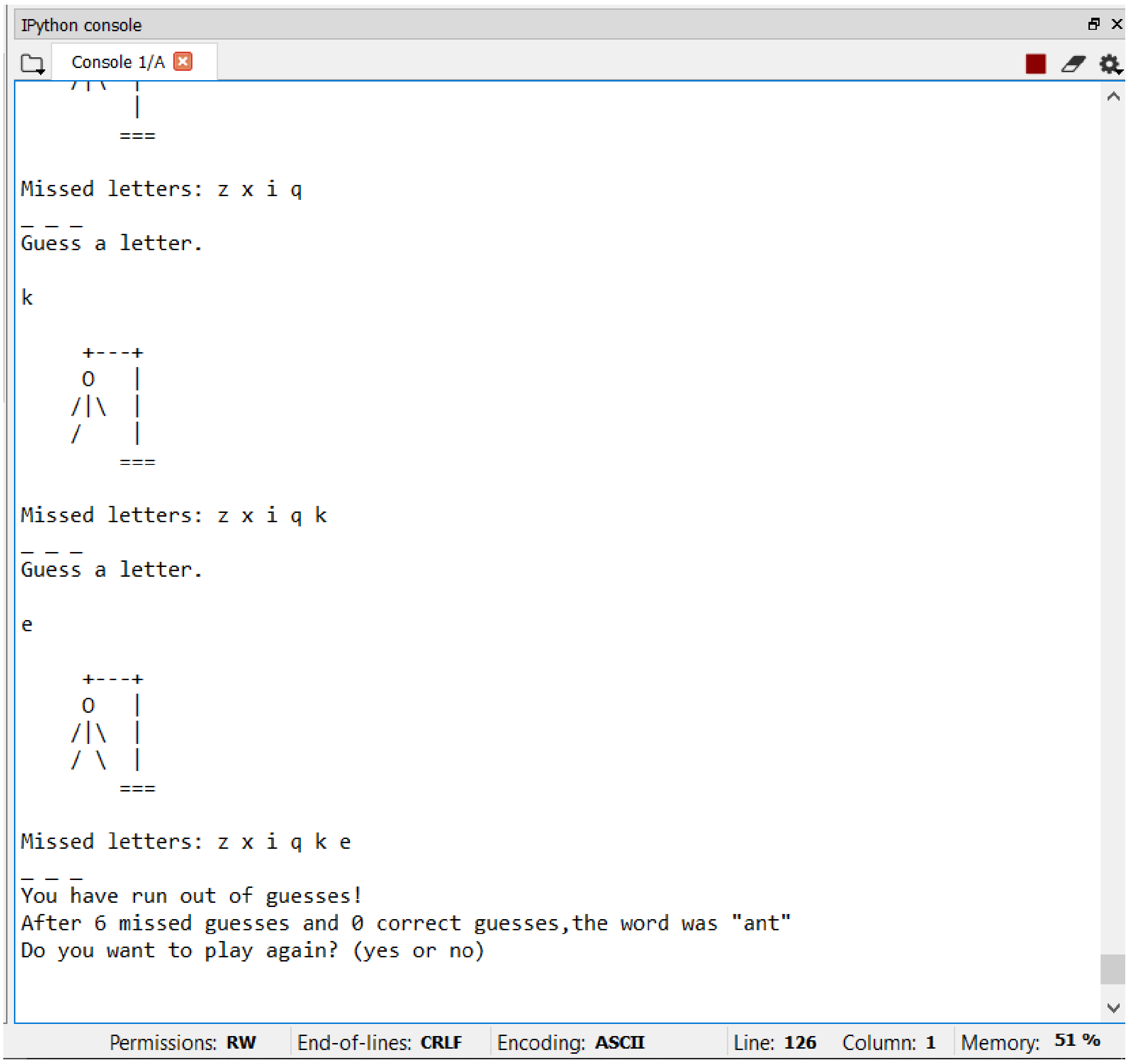
- The player loses the game.



```
              |
             ===

Missed letters: z v q i
_ _ _
Guess a letter.

k

      +---+
      O   |
     /|\  |
     /    |
         ===

Missed letters: z v q i k
_ _ _
Guess a letter.

e

      +---+
      O   |
     /|\  |
     / \  |
         ===

Missed letters: z v q i k e
_ _ _
You have run out of guesses!
After 6 missed guesses and 0 correct guesses,the word was "ram"
Do you want to play again? (yes or no)
```

Permissions: RW   End-of-lines: CRLF   Encoding: ASCII   Line: 126   Column: 1   Memory: 48 %

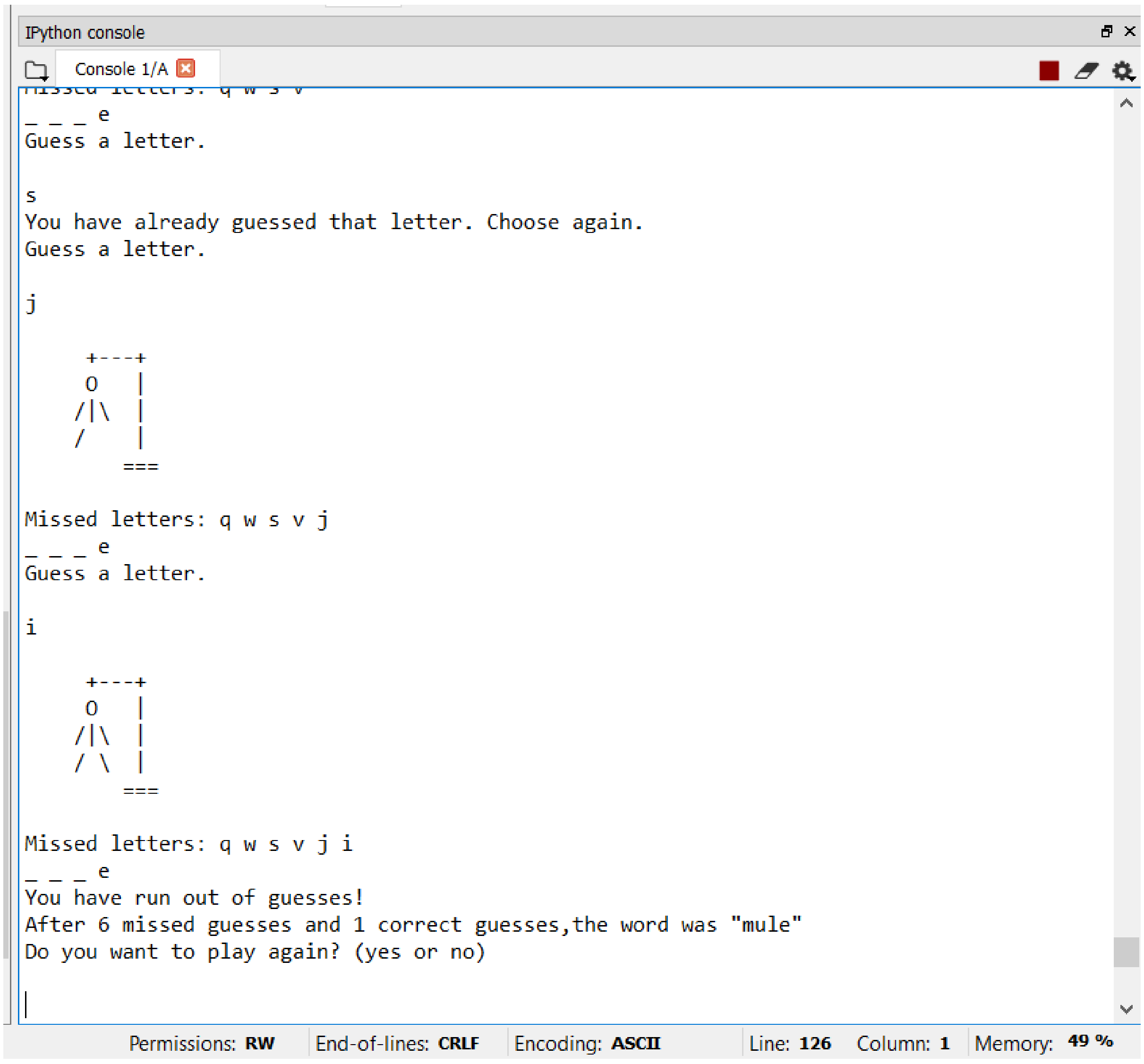
Fig 5.2.3 The player loses the game making all wrong choices.



```
         |
        ===

Missed letters: z x i q

_ _ _
Guess a letter.

k

     +---+
     O   |
    /|\  |
    /    |
        ===

Missed letters: z x i q k

_ _ _
Guess a letter.

e

     +---+
     O   |
    /|\  |
    / \  |
        ===

Missed letters: z x i q k e

_ _ _
You have run out of guesses!
After 6 missed guesses and 0 correct guesses,the word was "ant"
Do you want to play again? (yes or no)
```

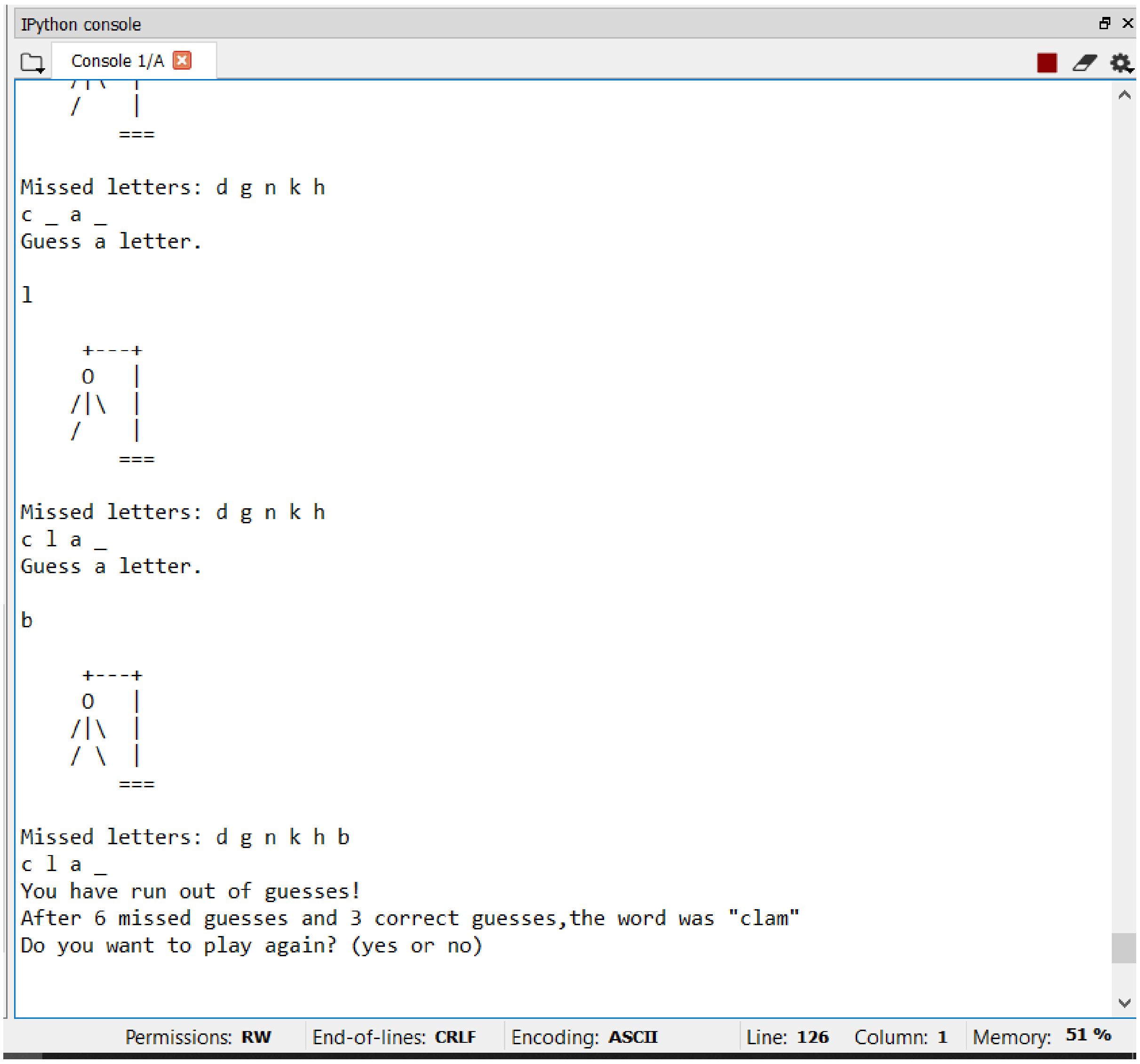
Permissions: **RW** End-of-lines: **CRLF** Encoding: **ASCII** Line: **126** Column: **1** Memory: **51 %**

Fig 5.2.4. The player loses the game making all wrong choices



```
_ _ _ e
Guess a letter.

s
You have already guessed that letter. Choose again.
Guess a letter.

j


     +---+
     O   |
    /|\  |
    /    |
        ===

Missed letters: q w s v j
_ _ _ e
Guess a letter.

i


     +---+
     O   |
    /|\  |
    / \  |
        ===

Missed letters: q w s v j i
_ _ _ e
You have run out of guesses!
After 6 missed guesses and 1 correct guesses,the word was "mule"
Do you want to play again? (yes or no)
```

Fig 5.2.5. The player loses the game making combination of correct and wrong



```
   /    |
       ===

Missed letters: d g n k h
c _ a _
Guess a letter.

l

     +---+
     O   |
    /|\  |
    /    |
        ===

Missed letters: d g n k h
c l a _
Guess a letter.

b

     +---+
     O   |
    /|\  |
    / \  |
        ===

Missed letters: d g n k h b
c l a _
You have run out of guesses!
After 6 missed guesses and 3 correct guesses,the word was "clam"
Do you want to play again? (yes or no)
```

Fig 5.2.6. The player loses the game making combination of correct and wrong choices (continued).

CHAPTER 6: - CONCLUSION

This project aimed at implementing a simple and easy to play version of the classic 2-player Hangman game so that the user can enjoy it without a second player and also so that he enjoys and has fun.

The project at this stage implements a simple user interface by which the code takes input as letters of the word and checks for its presence. The next steps to this project would be to enhance the GUI front end by adding some more buttons, message box, and labels. For this the Tkinter toolkit can be used for the GUI package.

CHAPTER 7: -BIBLIOGRAPHY

- https://inventwithpython.com › chapter9
- https://trinket.io › python
- https://selftaught.blog › python-tutorial-build-hangman
- Python program for word guessing game - GeeksforGeeks