

Mini – Python Project

ON

# A Game Of “Hangman”

**A Mini Project Submitted By:**

Dhanush R.	4NM21AI024
Kartik Sanil	4NM21AI034
Nihal Mohan	4NM21AI045
Prajwal Simha	4NM21AI048

**Under The Guidance Of**

***Mr. Sudesh Rao***

*Assistant Professor*

**Department Of Artificial Intelligence and Machine Learning**

*In partial fulfilment of the requirements for*  
**Object Oriented Programming with Python – 21AM305**

**NMAM Institute of Technology**  
**Nitte - 574110**



**NITTE**  
EDUCATION TRUST

**NMAM INSTITUTE  
OF TECHNOLOGY**

Nitte (DU) established under Section 3 of UGC Act 1956 | Accredited with 'A+' Grade by NAAC



**NITTE**  
EDUCATION TRUST

**N.M.A.M. INSTITUTE OF TECHNOLOGY**  
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)  
Nitte – 574 110, Karnataka, India

ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC

## CERTIFICATE

*Certified that the mini project work entitled*

***"A Game of Hangman"***

*is a bona fide work carried out by*

***Nihal Mohan Shettigar***

***(4NM21AI045)***

***Dhanush Rajashekar***

***(4NM21AI045)***

***Kartik Sanil***

***(4NM21AI034)***

***Prajwal Simha***

***(4NM21AI048)***

*in partial fulfilment of the requirements for the award of*

***Bachelor of Engineering Degree in Artificial Intelligence and Machine Learning Engineering***

*prescribed by Visvesvaraya Technological University, Belgaum*

*during the year 2022-2023.*

*It is certified that all corrections/suggestions indicated for Internal Assessment have been*

*incorporated in the report deposited in the departmental library.*

*The mini project report has been approved as it satisfies the academic requirements in respect of the*

*mini project work prescribed for the Bachelor of Engineering Degree.*

**Signature of Guide**

**Signature of HOD**

**Evaluation**

**Name of the Examiners**

**Signature with Date**

1. \_\_\_\_\_

2. \_\_\_\_\_

## **ACKNOWLEDGEMENT**

We believe that our mini project will be complete only after we thank the people who have contributed to make this mini project successful.

First and foremost, our sincere thanks to our beloved principal, **Dr. Niranjan N. Chiplunkar** for giving us an opportunity to carry out our mini project work at our college and providing us with all the needed facilities.

I acknowledge the support and valuable inputs given by, **Dr. Sharada U Shenoy** the Head of the Department, Artificial Intelligence and Machine Learning Engineering, NMAMIT, Nitte

We express our deep sense of gratitude and indebtedness to our guide **Mr. Sudesh Rao, Assistant Professor** Artificial Intelligence and Machine Learning Engineering, for his inspiring guidance, constant encouragement, support and suggestions for improvement during the course for our mini project.

We also thank all those who have supported us throughout the entire duration of our mini project.

Finally, we thank the staff members of the Department of Artificial Intelligence and Machine Learning Engineering and all our friends for their honest opinions and suggestions throughout the course of our mini project.

**Nihal Mohan Shettigar**

**Dhanush Rajashekar**

**Kartik Sanil**

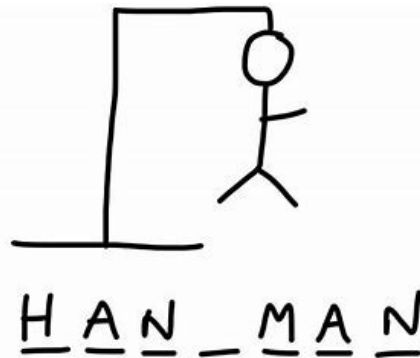
**Prajwal Simha**

## **TABLE OF CONTENTS**

Title .....	1
Certificate .....	2
Acknowledgement .....	3
Table Of Contents .....	4
Abstract .....	5
Introduction .....	6-11
▪ What is Python?	
▪ Why Python?	
▪ Python Examples	
▪ What's it about?	
▪ Packages and Functions Used?	
Implementation Of Code .....	12-14
Output .....	15
System Requirements .....	16
References .....	17

## ABSTRACT

**Hangman** is a guessing game for two or more players. One player thinks of a word, phrase or sentence and the other(s) tries to guess it by suggesting letters within a certain number of guesses. Was originally a Paper-and-pencil game, there are now electronic versions. The word to guess is represented by a row of dashes representing each letter of the word. Rules may permit or forbid proper nouns, such as names, places, movies, brands, or slang. If the guessing player suggests a letter which occurs in the word, the other player writes it in all its correct positions. If the suggested letter does not occur in the word, the other player removes, or either alternatively, adds one element of a hanged stick figure as a tally mark. Generally, the game ends once the word is guessed, or if the stick figure is complete signifying that all guesses have been used. The player guessing the word may, at any time, attempt to guess the whole word. Eventually if the word is correct, the game is Over and the Guesser wins. Otherwise, the player may choose to penalize the Guesser, by adding an element to the Stick Figure.



# INTRODUCTION

## What is Python?

Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming. It was created by **Guido van Rossum**, and first released on February 20, 1991.

While you may know the python as a large snake, the name of the Python programming language comes from an old BBC television comedy sketch series called **Monty Python's Flying Circus**.

One of the amazing features of Python is the fact that it is actually one person's work. Usually, new programming languages are developed and published by large companies employing lots of professionals, and due to copyright rules, it is very hard to name any of the people involved in the project. Python is an exception.

Of course, van Rossum did not develop and evolve all the Python components himself. The speed with which Python has spread around the world is a result of the continuous work of thousands (very often anonymous) programmers, testers, users (many of them aren't IT specialists) and enthusiasts, but it must be said that the very first idea (the seed from which Python sprouted) came to one head - Guido's.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

## Why Python?

What makes Python so special? How does it happen that programmers, young and old, experienced and novice, want to use it? How did it happen that large companies adopted Python and implemented their flagship products using it?

There are many reasons – we’ve listed some of the already, but let’s enumerate them again in a more practical manner:

- It’s **easy to learn** – The time needed to learn Python is shorter than for many other languages, this means that it’s possible to start the actual programming faster.
- It’s **easy to use** for writing new software – It’s often possible to write code faster when using Python.
- It’s **easy to obtain, install and deploy** – Python is free, open and multiplatform, not all languages can boast that.
- It’s **easy to understand** – It’s also often easier to understand someone else’s code faster if it is written in Python.
- It’s **easy to teach** – The teaching workload is smaller than that needed by other languages, this means that the person teaching can put more emphasis on general programming techniques rather than wasting energy and time on various tricks, strange exceptions and incomprehensible rules.

## Python Examples

Python is a great choice for:

- **Scientific and Numeric computing** (e.g. **SciPY** – a collection of packages for the purpose of mathematics, science, and engineering. **Ipython** – an interactive shell that features editing and recording of work sessions)
- **Education** (it’s a brilliant language for teaching programming)
- **Software Development** (build control, management, and testing – Scons, Buildbot, Apache Gump, Roundup, Trac)
- **Desktop GUIs** (e.g., wxWidgets, Kivy, Qt)

- **Web and Internet development** (e.g., Django and Pyramid frameworks, Flask and Bottle micro-frameworks)
- **Games** (e.g., Battlefield series, Sid Meier's Civilization IV), **websites and services** (e.g., Dropbox, UBER, Pinterest, BuzzFeed)
- **Business applications** (ERP and e-commerce systems – Odoo, Tryton)

## **What's it About?**

This is a Hangman game using the Python Programming Language, by implementing a Simple GUI using the Python Package, **'Tkinter'**. We also include a standard python library known, **'random'** which has the in-built function, to select a random word from a pre-built **'txt'** file which contains the list of all the words, which will be the Word to Be Guessed.

The GUI of this Application contains Tkinter widgets like Entry, Label, Canvas, Button, Messagebox.

Here below, is a Basic Methodology used for the Working of the Application:

- The Program randomly selects a Word from the, already predefined Text File which has Varieties of words in it.
- Initially, the word will be represented with Dashes, corresponding to the number of Letters in them. Also, a pictorial representation of the Stick Figure without any Part(s) is placed.
- The User/Player is made to input Random guessed letters one-by-one.
- When the guessed letter is right, the dash corresponding to the letter guessed gets filled with the letter, and the game is continued.
- If the guessed letter is incorrect, the number of Remaining Attempts Decreases by one, and the Stick Figure gets a Part of its body.



## **Packages and Functions Used:**

- **Random Function:**

Python Random module is an in-built module of Python which is used to generate random numbers. These are pseudo-random numbers means these are not truly random. It can be used to perform random actions such as generating random numbers, print random a value for a list or string.

Example: *Printing a random value from a list*

```
#import random  
  
import random  
  
#Prints a random value from the list  
  
list1 = [1, 2, 3, 4, 5, 6]  
  
print(random.choice(list1))
```

- **Class Function:**

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state.

Syntax: *Class Definition*

```
class ClassName:  
  
    #Statement
```

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accesses and used by creating an instance of that class. A class is like a blueprint for an object.

- **Class Function:**

The `list()` function creates a list object. A list object is a collection which is ordered and changeable.

**Syntax:**

```
list(iterable)
```

**Example:**

```
x = list(('apple', 'banana', 'cherry'))
```

- **Structure Function:**

The module **struct** is used to convert the native data types of Python into string of bytes and vice versa. It is a built-in module available in Python3.

1. **Struct.pack()**

The method `struct.pack()` is used to convert the data types into bytes. It takes multiple arguments based on the first string.

2. **Struct.unpack()**

We have another method called `struct.unpack()` that convert bytes to native Python data types. It takes two arguments, the first one is similar to the **pack()** method and the second one is the result of **struct.pack()** method.

- **tkinter (Packages Used):**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps-

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

- **tkinter (Modules):**

- Tk- creates a tkinter object.
- Attributes- modify the attributes or setting of the tkinter root window.
- Geometry- sets the resolution of the tkinter window.
- Font- combines a font face, size and style to create a tkinter font object.
- Label- creates a text slot which cannot be modified.
- Pack- assemble all units of the root window together and start them.
- Mainloop- starts the tkinter window and keeps the window ready to recognize events.
- Config- modifies the property of tkinter elements.

## IMPLEMENTATION OF CODE:

```
from tkinter import *
import random
from tkinter import messagebox

#Initializing Empty List
mywords=[]
file1 = open(r"commonword.txt","r")

#Appending words from file to the list
for x in file1:
    mywords.append(x.replace('\n', ''))

word=random.choice(mywords)
random_word=List(word)
p=[]
s='_ '*len(random_word)
p=s.split(' ')
p.pop(len(random_word))
actual=random_word.copy()

class Hangman:
    def __init__(self,master):
        self.count=0
        self.structure(master)
        self.rn=master

    def structure(self, master):
        """ Instruction Label """

        # Create instruction label for Program
        self.inst_lbl = Label(master, text="Welcome to Hangman Game!")
        self.inst_lbl.grid(row=0, column=0, columnspan=2, sticky=W)

        """ Guess Input """

        # Create label for entering Guess
        self.guess_lbl = Label(master, text="Enter your Guess:")
        self.guess_lbl.grid(row=1, column=0, sticky=W)

        # Create entry widget to accept Guess
        self.guess_ent = Entry(master)
        self.guess_ent.grid(row=1, column=1, sticky=W)

        # Create a space
        self.gap2_lbl1 = Label(master, text=" ")
        self.gap2_lbl1.grid(row=2, column=0, sticky=W)

        # Creating a submit button
        self.submit_btn = Button(master, text="Submit", command=self.submit, height=1, width=20)
        self.submit_btn.grid(row=3, column=1, sticky=W)

        master.bind('<Return>', self.submit)

        # Create a space

        self.gap2_lbl2 = Label(master, text=" ")
        self.gap2_lbl2.grid(row=4, column=0, sticky=W)

        """ RESET """

        # Creating a reset button
        self.reset_btn = Button(master, text="Reset", command=self.reset, height=2, width=20)
        self.reset_btn.grid(row=9, column=2, sticky=W)
```

```

# Create a space
self.gap2_lb13 = Label(master, text=" ")
self.gap2_lb13.grid(row=5, column=0, sticky=W)

self.inst_lb2 = Label(master, text='Life:10')
self.inst_lb2.grid(row=1, column=2, columnspan=2, sticky=W)

# Creating Label to Display Message
self.inst_lb3 = Label(master, text='')
self.inst_lb3.grid(row=6, column=0, columnspan=2, sticky=W)

# Creating label to display current Guessed Status of Word

self.curr_char1 = Label(master, text=p)
self.curr_char1.place(x=100, y=130)
self.curr_char = Label(master, text="Current Status:")
self.curr_char.place(x=0, y=130)

# Create a Hangman's Background

self.c = Canvas(master, height=300, width=200)
self.c.grid(row=9, column=0, sticky=W)
self.l = self.c.create_line(70, 20, 70, 250, width=2)
self.l1 = self.c.create_line(70, 20, 150, 20, width=2)
self.l2 = self.c.create_line(150, 20, 150, 50, width=2)

def current_status(self, char):
    self.curr_char1 = Label(self, text=char)
    self.curr_char1.place(x=100, y=130)

def reset(self):
    self.guess_ent.delete(0, 'end')

def submit(self, *args):
    # Taking Entry From Entry Field
    char = self.guess_ent.get()

    # Checking whether Entry Field is empty or not
    if (len(char) == 0):
        messagebox.showwarning("Warning", "Entry field is Empty!")
    if (len(char) > 1):
        messagebox.showwarning("Warning", "Enter a Single Letter Only!")

    if char in actual and len(char) == 1:
        l = actual.count(char)
        for j in range(l):
            i = actual.index(char)
            p.insert(i, char)
            p.pop(i + 1)
            actual.insert(i, '_')
            actual.pop(i + 1)
            self.inst_lb2.config(text='Life:' + str(10 - self.count))
            self.inst_lb3.config(text='Right Guessed!')
            self.guess_ent.delete(0, 'end')
            self.current_status(p)

        elif (len(char) == 1):
            self.count = self.count + 1
            self.inst_lb2.config(text='Life:' + str(10 - self.count))
            self.inst_lb3.config(text='Wrong Guessed!')
            self.guess_ent.delete(0, 'end')

    # Creating Hangman's parts order wise if wrong character is Guessed
    if (self.count == 1):
        self.cir = self.c.create_oval(125, 100, 175, 50, width=2)
    elif (self.count == 2):

```

```

        self.el = self.c.create_line(135, 65, 145, 65, width=2)
    elif (self.count == 3):
        self.er = self.c.create_line(155, 65, 165, 65, width=2)
    elif (self.count == 4):
        self.no = self.c.create_line(150, 70, 150, 85, width=2)
    elif (self.count == 5):
        self.mo = self.c.create_line(140, 90, 160, 90, width=2)
    elif (self.count == 6):
        self.l3 = self.c.create_line(150, 100, 150, 200, width=2)
    elif (self.count == 7):
        self.hl = self.c.create_line(150, 125, 100, 150, width=2)
    elif (self.count == 8):
        self.hr = self.c.create_line(150, 125, 200, 150, width=2)
    elif (self.count == 9):
        self.fl = self.c.create_line(150, 200, 100, 225, width=2)
    elif (self.count == 10):
        self.fr = self.c.create_line(150, 200, 200, 225, width=2)

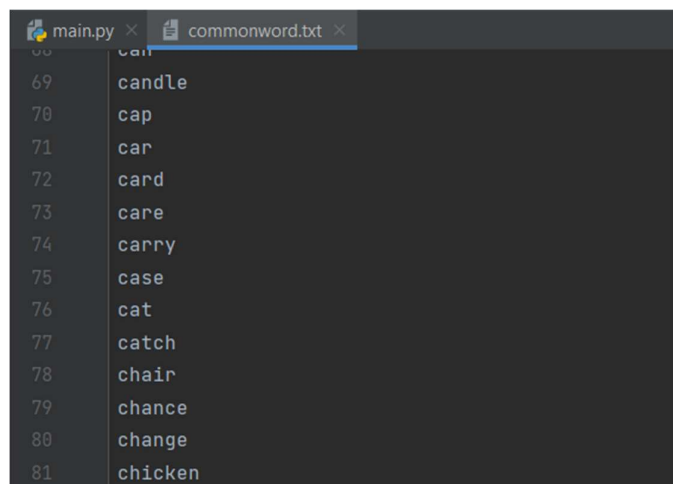
# Condition of Player Won
if (p == random_word):
    self.inst_lb3.config(text='\n\t\t\t\t\tYou perfectly guessed the word!')
    messagebox.showinfo("Hey There :)", "You Won! Congratulations!")
    self.rr.destroy()

# Condition Of player Loose
elif (self.count >= 10):
    self.inst_lb3.config(text='\n\t\t\t\t\tYou lost.... \n\t\t\t\t\tThe word is ' + word)
    messagebox.showinfo("Oops!", "You lost :( Please Try Again!")
    self.rr.destroy()

root = Tk()
root.title("Hangman Game")
root.geometry("580x480")
app = Hangman(root)
root.mainloop()

```

## Sample Words Used:

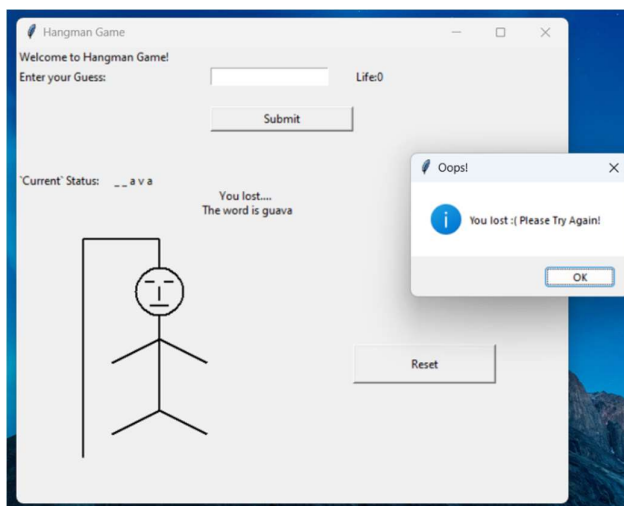
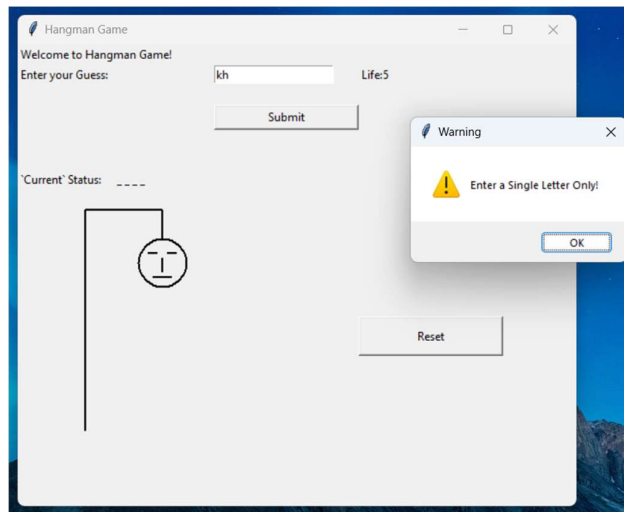
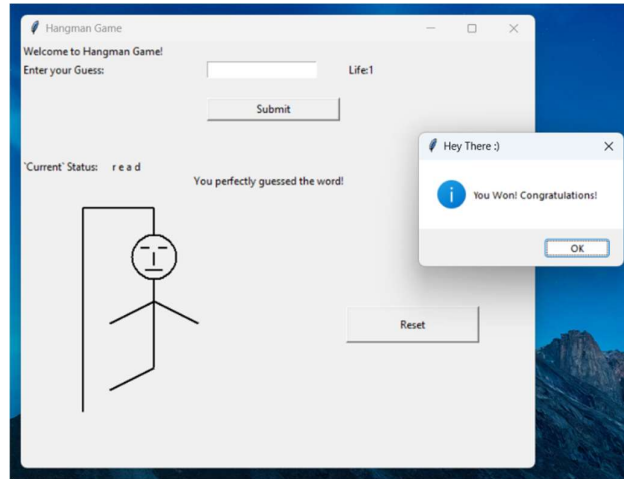


```

68  can
69  candle
70  cap
71  car
72  card
73  care
74  carry
75  case
76  cat
77  catch
78  chair
79  chance
80  change
81  chicken

```

## OUTPUT:



## **SYSTEM REQUIREMENTS**

### **SOFTWARE:**

- Visual Studio Code or PyCharm [JetBrains] or any IDEs which support Python
- Python Interpreter (Version 3.8 or above)
- Necessary Packages / Libraries

### **OS:**

- Windows / Ubuntu / MacOS Based Operating System, whichever's convenient.

### **HARDWARE:**

- 100MB Free Disk space
- RAM: 4GB or above



## References

- [https://en.wikipedia.org/wiki/Hangman\\_\(game\)](https://en.wikipedia.org/wiki/Hangman_(game))
- <https://www.geeksforgeeks.org/python-tkinter-tutorial/>
- <https://www.geeksforgeeks.org/hangman-game-python/>
- <https://www.codespeedy.com/hangman-game-with-gui-in-python-using-tkinter/>
- <https://www.geeksforgeeks.org/python-random-module/>
- <https://www.geeksforgeeks.org/python-classes-and-objects/>
- <https://www.tutorialspoint.com/struct-module-in-python#:~:text=The%20module%20struct%20is%20used,related%20to%20the%20C%20languages>