

PYTHON MINI PROJECT REPORT

The Second Year Python Mini Project Report Submitted To

Don Bosco Institute of Technology

For The Award of:

DEGREE IN ELECTRONICS AND TELECOMMUNICATION



Department of Electronics and Telecommunication Engineering

Don Bosco Institute of Technology

Kurla, Mumbai

Academic Year 2021-22

ACKNOWLEDGEMENT

We are thankful to the institute “Don Bosco Institute of Technology” for providing necessary facility to carry out learning through the Python Miniproject successfully.

It is our duty to record my sincere thanks and gratitude towards the institute.

Also, I am highly obliged to our faculty Ms. Poonam Chakraborty ma'am who provided such a great opportunity to complete this miniproject by guiding us throughout and clearing our doubts time to time.

Thank You Again,

Jaipreet Singh 11

Shruti Gokhale 40

**Department of Electronics and Telecommunication Engineering
Don Bosco Institute Of Technology, Mumbai Maharashtra, India**

Contents

Purpose

Steps to build game

Code

Screenshot of Code

Screenshot of Output

Conclusion

Purpose

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a generalpurpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. The better way to learn Python is to learn it and then implement it.

Miniprojects are one way to have hands on experience on python and implementation.

Here in our mini project, we have created a Hangman game using Python code to get the hands on experience on it.

Steps on Hangman Game in Python

1. Import the random module.
2. Define a function to choose a word at random.
3. Allot 7 lives for the user.
4. Create a decision-making process, to check that the user only enters alphabets and not numbers as a name.
5. A while loop is created to check the condition that the letters of words are more than zero and lives are not zero.
6. Create visual representation of the 'hangman' as the name suggests i.e.

```
_____
|/   |
|/   ()
|   |
|   /\
|
```

7. For each wrong letter one item is added to the hangman representation.
8. Create a list for words to be chosen from named as words which is as

```
words = [ " ", " ", . . . ].
```

CODE

hangman.py

```
import random
```

```
from words import words
```

```
from hangman_visual import lives_visual_dict
```

```
import string
```

```
def get_valid_word(words):
```

```
    word = random.choice(words) # randomly chooses something from the list
```

```
    while '-' in word or ' ' in word:
```

```
        word = random.choice(words)
```

```
    return word.upper()
```

```
def hangman():
```

```
    word = get_valid_word(words)
```

```
    word_letters = set(word) # letters in the word
```

```
    alphabet = set(string.ascii_uppercase)
```

```
    used_letters = set() # what the user has guessed
```

```
    lives = 7
```

```
    # getting user input
```

```
    while len(word_letters) > 0 and lives > 0:
```

```
        # letters used
```

```
        # ' '.join(['a', 'b', 'cd']) --> 'a b cd'
```

```
        print('You have', lives, 'lives left and you have used these letters: ', ''.join(used_letters))
```

```
        # what current word is (ie W - R D)
```

```
word_list = [letter if letter in used_letters else '-' for letter in word]
print(lives_visual_dict[lives])
print('Current word: ', ' '.join(word_list))
```

```
user_letter = input('Guess a letter: ').upper()
```

```
if user_letter in alphabet - used_letters:
```

```
    used_letters.add(user_letter)
```

```
    if user_letter in word_letters:
```

```
        word_letters.remove(user_letter)
```

```
    print("")
```

```
else:
```

```
    lives = lives - 1 # takes away a life if wrong
```

```
    print("\nYour letter,', user_letter, 'is not in the word.')"
```

```
elif user_letter in used_letters:
```

```
    print("\nYou have already used that letter. Guess another letter.")
```

```
else:
```

```
    print("\nThat is not a valid letter.")
```

```
# gets here when len(word_letters) == 0 OR when lives == 0
```

```
if lives == 0:
```

```
    print(lives_visual_dict[lives])
```

```
    print('You died, sorry. The word was', word)
```

```
else:
```

```
    print('YAY! You guessed the word', word, '!!!')
```

```
if __name__ == '__main__':
```

```
    hangman()
```

hangman_visual.py

```
lives_visual_dict = {
```

```
    0: """
```

```
        _____
        | /      |
        |/      ()
        |      |
        |      /\
        |
        """,
```

```
    1: """
```

```
        _____
        | /      |
        |/      ()
        |      |
        |      /
        |
        """,
```

```
    2: """
```

```
        _____
        | /      |
        |/      ()
        |      |
        |
        |
        """,
```

```
    3: """
```

```
        _____
        | /      |
        |/      ()
        |

```

4: """"

5: ""

6: ""

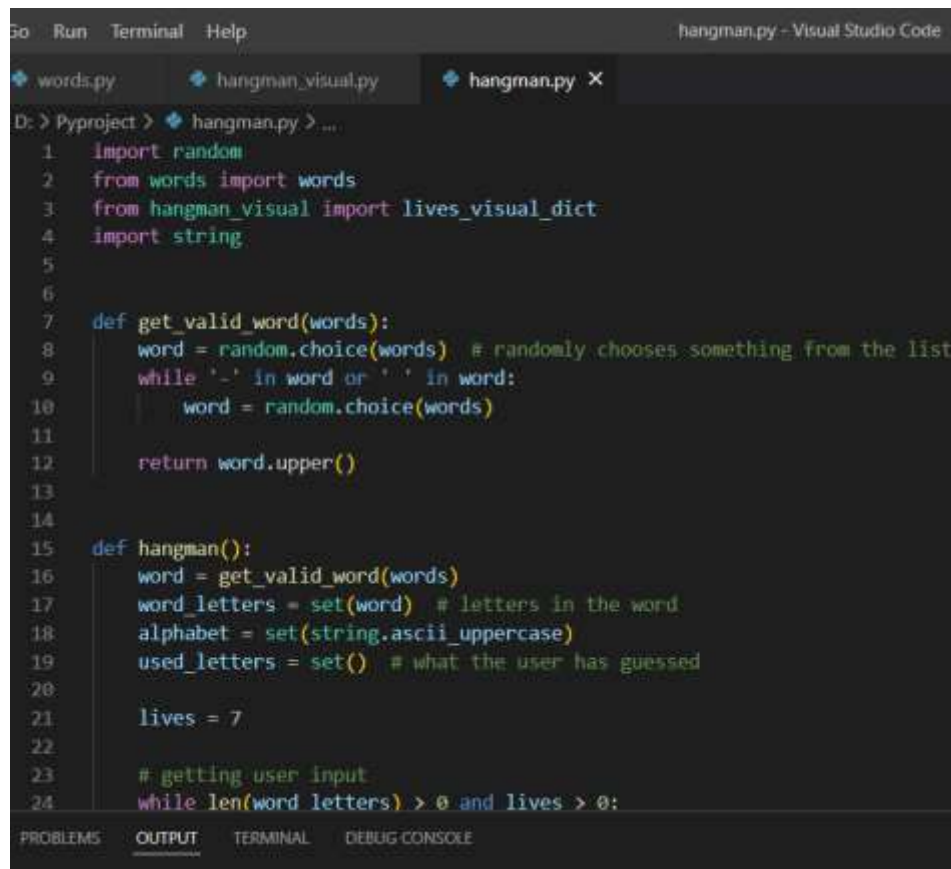
7: "" ,

words.py

```
words = ["aback","abaft","abandoned","abashed","aberrant",.....
,"yawn","yell","zip","zoom"]
```

reference: <https://www.randomlists.com/data/words.json>

Screenshot of Code

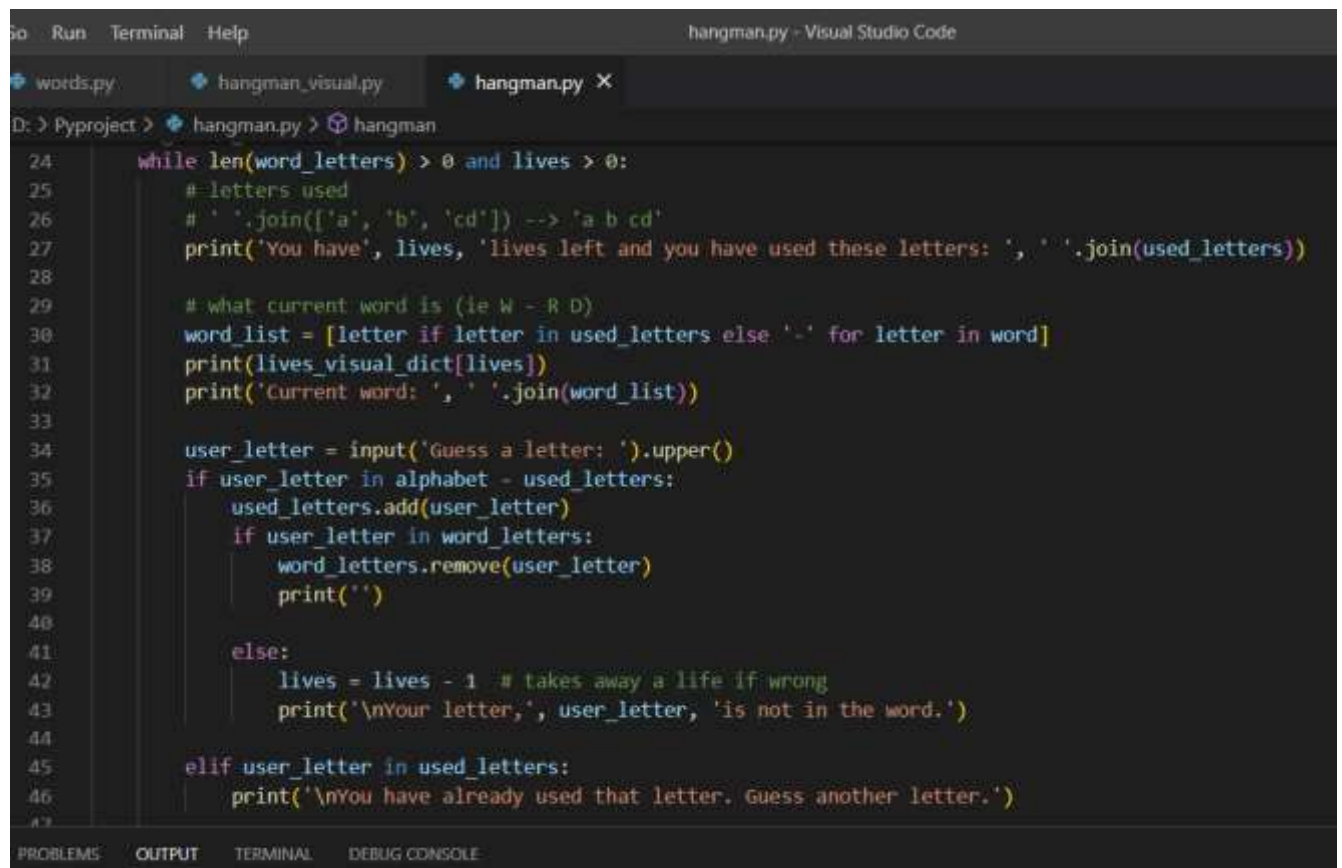


This screenshot shows the first part of the `hangman.py` file in Visual Studio Code. The code includes imports for `random`, `words`, `lives_visual_dict`, and `string`. It defines a `get_valid_word` function that selects a random word from a list, ensuring it contains no hyphens or apostrophes. The `hangman` function is also shown, which initializes the word, letters, alphabet, and lives.

```
Go Run Terminal Help hangman.py - Visual Studio Code

words.py hangman_visual.py hangman.py X

D:\> Pyproject > hangman.py > ...
1 import random
2 from words import words
3 from hangman_visual import lives_visual_dict
4 import string
5
6
7 def get_valid_word(words):
8     word = random.choice(words) # randomly chooses something from the list
9     while '-' in word or ' ' in word:
10         word = random.choice(words)
11
12     return word.upper()
13
14
15 def hangman():
16     word = get_valid_word(words)
17     word_letters = set(word) # letters in the word
18     alphabet = set(string.ascii_uppercase)
19     used_letters = set() # what the user has guessed
20
21     lives = 7
22
23     # getting user input
24     while len(word_letters) > 0 and lives > 0:
```



This screenshot shows the continuation of the `hangman` function in `hangman.py`. It details the logic for displaying the current word state, handling user input, and updating the game state based on whether the letter is correct or incorrect.

```
Go Run Terminal Help hangman.py - Visual Studio Code

words.py hangman_visual.py hangman.py X

D:\> Pyproject > hangman.py > hangman
24 while len(word_letters) > 0 and lives > 0:
25     # letters used
26     # ' '.join(['a', 'b', 'cd']) --> 'a b cd'
27     print('You have', lives, 'lives left and you have used these letters: ', ' '.join(used_letters))
28
29     # what current word is (ie W - R D)
30     word_list = [letter if letter in used_letters else '-' for letter in word]
31     print(lives_visual_dict[lives])
32     print('Current word: ', ' '.join(word_list))
33
34     user_letter = input('Guess a letter: ').upper()
35     if user_letter in alphabet - used_letters:
36         used_letters.add(user_letter)
37         if user_letter in word_letters:
38             word_letters.remove(user_letter)
39             print('')
40
41         else:
42             lives = lives - 1 # takes away a life if wrong
43             print('\nYour letter,', user_letter, 'is not in the word.')
44
45     elif user_letter in used_letters:
46         print('\nYou have already used that letter. Guess another letter.')
47
48     #
```

```

40
41         else:
42             lives = lives - 1 # takes away a life if wrong
43             print('\nYour letter,', user_letter, 'is not in the word.')
44
45         elif user_letter in used_letters:
46             print('\nYou have already used that letter. Guess another letter.')
47
48         else:
49             print('\nThat is not a valid letter.')
50
51     # gets here when len(word_letters) == 0 OR when lives == 0
52     if lives == 0:
53         print(lives_visual_dict[lives])
54         print('You died, sorry. The word was', word)
55     else:
56         print('YAY! You guessed the word', word, '!!!')
57
58
59 if __name__ == '__main__':
60     hangman()

```

```

D:\> Pyproject > hangman_visual.py > ...
1  lives_visual_dict = {}
2      0: """
3          |
4          | / \
5          ||   ()
6          |
7          | / \
8          |
9          +---+
10         1: """
11        |
12        | / \
13        ||   ()
14        |
15        | / \
16        |
17        +---+
18        2: """
19        |
20        | / \
21        ||   ()
22        |
23        | / \
24        |

```

The screenshot shows a PyCharm IDE with a dark theme. The top bar has tabs for 'Go', 'Run', 'Terminal', and 'Help'. Below the tabs, there are three open files: 'words.py', 'hangman_visual.py' (which is the active file), and 'hangman.py'. The main editor area shows the code for 'hangman_visual.py'. The code is a Python script for a Hangman game. It includes a list of words, a function to choose a random word, a function to check if a letter is in the word, a function to check if a letter has been guessed before, a function to update the progress bar, and a function to print the current state of the game. The game is currently in progress, with the word 'SECRET' being guessed. The current state shows the word with underscores for unknown letters and a progress bar. The player has 6 lives remaining. The code includes logic for guessing letters, checking for correct/incorrect guesses, and updating the progress bar.

```

36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```

At the bottom of the IDE, there are four tabs: 'PROBLEMS', 'OUTPUT', 'TERMINAL', and 'DEBUG CONSOLE'. The 'OUTPUT' tab is currently selected, showing the output of the program.

Screenshot of Output

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

D:\Pyproject>python hangman
python: can't open file 'D:\\Pyproject\\hangman': [Errno 2] No such file or directory

D:\Pyproject>python hangman.py
You have 7 lives left and you have used these letters:

Current word: - - - -
Guess a letter: a

Your letter, A is not in the word.
You have 6 lives left and you have used these letters:  A

      |
      |
      |
      |
      |
      |
      |

Current word: - - - -
Guess a letter: s

Your letter, S is not in the word.
You have 5 lives left and you have used these letters:  S A

      / \
     /   \
    /       \
   /         \
  /           \
 /             \
/               \

Current word: - - - -
Guess a letter: d

You have 5 lives left and you have used these letters:  S A D

      / \
     /   \
    /       \
   /         \
  /           \
 /             \
/               \

Current word: D - - -
Guess a letter: f
```

C:\Windows\System32\cmd.exe

Your letter, F is not in the word.

You have 4 lives left and you have used these letters: F S A D

```
| / |
| / |
|  |
|  |
|  |
```

Current word: D _ _ _

Guess a letter: e

You have 4 lives left and you have used these letters: F S D E A

```
| / |
| / |
|  |
|  |
|  |
```

Current word: D E _ _

Guess a letter: c

Your letter, C is not in the word.

You have 3 lives left and you have used these letters: C F S D E A

```
| / |
| / |
|  |
|  |
|  |
```

Current word: D E _ _

Guess a letter: x

Your letter, X is not in the word.

You have 2 lives left and you have used these letters: C F X S D E A

```
| / |
| / |
|  |
|  |
|  |
```

C:\Windows\System32\cmd.exe

Your letter, X is not in the word.

You have 2 lives left and you have used these letters: C F X S D E A

```
| / |
| / |
|  |
|  |
|  |
```

Current word: D E _ _

Guess a letter: z

Your letter, Z is not in the word.

You have 1 lives left and you have used these letters: Z C F X S D E A

```
| / |
| / |
|  |
|  |
|  |
```

Current word: D E _ _

Guess a letter: h

Your letter, H is not in the word.

```
| / |
| / |
|  |
|  |
|  |
```

You died, sorry. The word was DEBT

D:\Pyproject>

CONCLUSION

So here we conclude that hangman is a popular word guessing game where the player attempts to build a missing word by guessing one letter at a time. After a certain number of incorrect guesses, the game ends and the player loses. The game also ends if the player correctly identifies all the letters of the missing word.