

1. A program for Hangman in python

The code satisfies the requirements

- i. One word generated randomly
- ii. Player will be presented with a number of blank spaces representing the missing letters the player needs to find.
- iii. If the player's chosen letter exists in the answer, then all places in the answer where that letter appear will be revealed.
- iv. Every time the player guesses a letter wrong, the player's life will be deducted.
- v. The player must find the missing word before the player's life becomes zero.
- vi. Player lives is represented using number instead of hangman picture

```
2. import random
3. from wordslist import words
4.
5. def get_word():
6.     word = random.choice(words)
7.     return word.upper()
8.
9. def hangman():
10.    word = get_word()
11.    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
12.    guessed_letter=[]
13.    lives =6
14.    guessed = False
15.    print(len(word),'letters in the word')
16.    print(len(word)*'_')
17.
18.    while guessed == False and lives > 0:
19.        print(lives,'lives left')
20.        guess = input('guess a letter:').upper()
21.        if len (guess) == 1:
22.            if guess not in alphabet:
23.                print('Invalid character')
24.                lives -=1
25.            elif guess in guessed_letter:
26.                print('you have already guessed the letter')
27.            elif guess not in word:
28.                print('Guessed letter not present in the word')
29.                guessed_letter.append(guess)
30.                lives -=1
31.            elif guess in word:
32.                print('your guess is present in the word')
33.                guessed_letter.append(guess)
34.
35.        status = ''
36.        if guessed == False:
```

```

37.         for letter in word:
38.             if letter in guessed_letter:
39.                 status += letter
40.                 if status == word:
41.                     lives = 0
42.                     print('you win the game')
43.             elif letter not in guessed_letter:
44.                 status += '_'
45.         print(status)
46.     print('You run out of guesses')
47. hangman()

```

Output

```

PS C:\Users\jilug> & C:/Users/jilug/AppData/Local/Programs/Python/Python37/pyth
7 letters in the word

_ _ _ _ _
6 lives left
guess a letter:e
Guessed letter not present in the word

_ _ _ _ _
5 lives left
guess a letter:a
Guessed letter not present in the word

_ _ _ _ _
4 lives left
guess a letter:i
your guess is present in the word
_
_ _ _ _ _
4 lives left
guess a letter:g
Guessed letter not present in the word
_
_ _ _ _ _
3 lives left
guess a letter:d
Guessed letter not present in the word
_
_ _ _ _ _
2 lives left
guess a letter:k
Guessed letter not present in the word
_
_ _ _ _ _
1 lives left
guess a letter:l
Guessed letter not present in the word
_
You run out of guesses
PS C:\Users\jilug>

```

2. Refactoring

Code smells

Duplicated code: The function `gets_word` is used only once, so it can be directly used in the main function

```

3. import random
4. from wordslist import words

```

```
5.
6. def hangman():
7.     word = random.choice(words).upper()
8.     alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
9.     guessed_letter=[]
10.    lives =6
11.    guessed = False
12.    print(len(word),'letters in the word')
13.    print(len(word)*'_')
14.
15.    while guessed == False and lives > 0:
16.        print(lives,'lives left')
17.        guess = input('guess a letter:').upper()
18.        if len (guess) == 1:
19.            if guess not in alphabet:
20.                print('Invalid character')
21.                lives -=1
22.            elif guess in guessed_letter:
23.                print('you have already guessed the letter')
24.            elif guess not in word:
25.                print('Guessed letter not present in the word')
26.                guessed_letter.append(guess)
27.                lives -=1
28.            elif guess in word:
29.                print('your guess is present in the word')
30.                guessed_letter.append(guess)
31.
32.        status = ''
33.        if guessed == False:
34.            for letter in word:
35.                if letter in guessed_letter:
36.                    status += letter
37.                    if status == word:
38.                        lives = 0
39.                        print('you win the game')
40.                elif letter not in guessed_letter:
41.                    status += '_'
42.            print(status)
43.        print('You run out of guesses')
44.hangman()
```

3. Create a Git directory for your assignment

Link: https://github.com/s331717/Jilu_hangman

Step 1: To create a new repository fill the columns owner and repository name. If you are selecting public domain you must ensure to select any license. Then click create repository. Then you will get the link for the project.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 s331717 ▾

Repository name *

/ Jilu_hangman ✓

Great repository names are short and memorable. Need inspiration? How about **reimagined-waffle?**

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.



Add a README file

This is where you can write a long description for your project. [Learn more.](#)



Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)



Choose a license

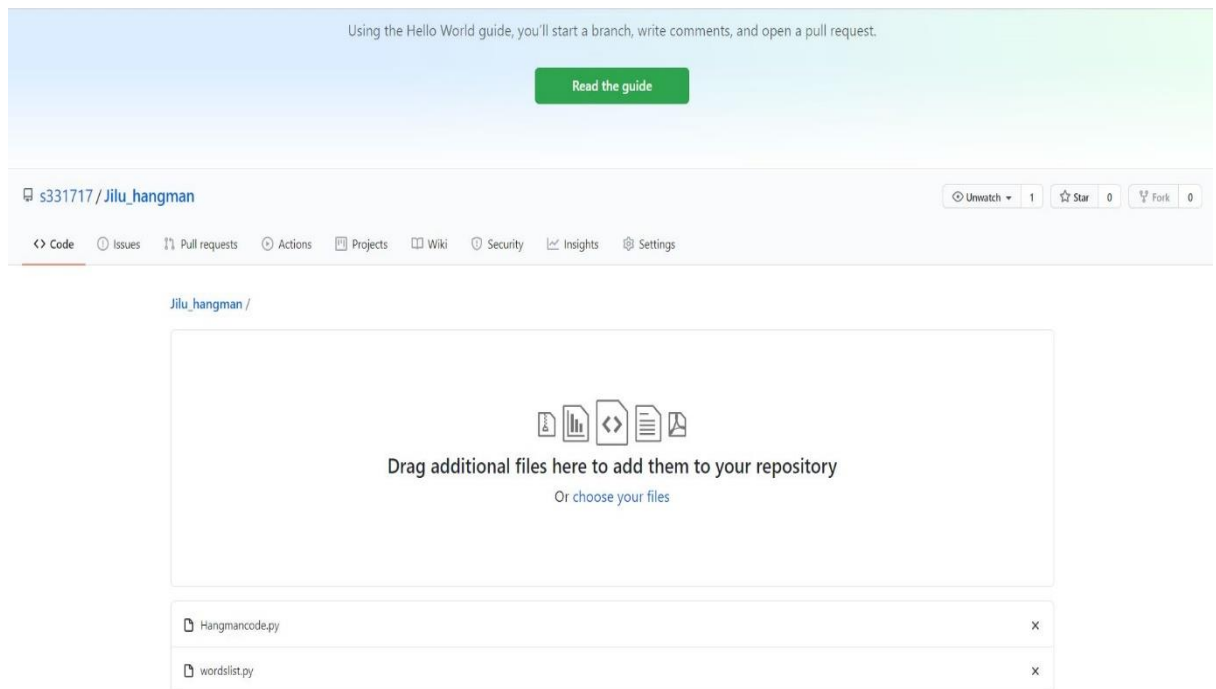
A license tells others what they can and can't do with your code. [Learn more.](#)

License: **None** ▾

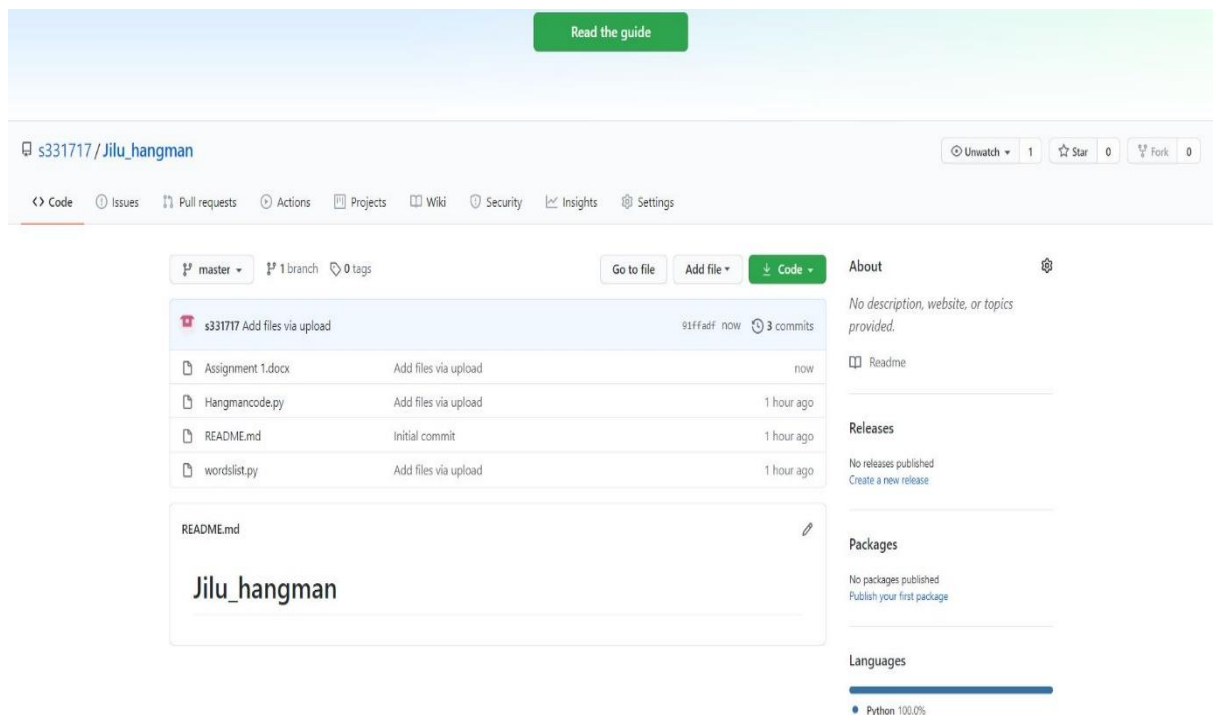
This will set  **master** as the default branch. Change the default name in your [settings](#).

Create repository

Step 2: Choose your files related to the project



Step 3: uploaded files can be seen in the link .



4. *How TDD has been implemented to create your program*

Test driven development is an approach program development in which the testing and coding are interleaved. Once the coding for a function is finished, then it will be tested and meet the requirements. After that coding for next requirement will do and after that its testing should finish before identifying next functionality

First requirement: check the random selection of word from wordlist

```
C: > Users > jilug > OneDrive > Desktop > Python > hangtdd.py > ...
1  import random
2  from wordslist import words
3
4  def get_word():
5      word =random.choice(words)
6      print(word)
7      return word.upper()
8
9  get_word()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\jilug> & C:/Users/jilug/AppData/Local/Programs/Python/Python37/python.exe
creature
PS C:\Users\jilug>

Second Requirement: To print the length of the word and ensure the blank spaces in the place of letters

```
C: > Users > jilug > OneDrive > Desktop > Python > hangtdd.py > ...
1  import random
2  from wordslist import words
3
4  def get_word():
5      word = random.choice(words)
6      print(word)
7      return word.upper()
8  def hangman():
9      word = get_word()
10     guessed_letter=[]
11     guessed = False
12     print(len(word), 'letters in the word')
13     print(len(word)*'_')
14     hangman()
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/powershell>

PS C:\Users\jilug> & C:/Users/jilug/AppData/Local/Programs/Python/Python37/python
cactus
6 letters in the word
PS C:\Users\jilug>

Third requirement: Main loop for getting input and check the condition such as whether the input letter is present in the word or the input is an invalid character, or it is an already enter letter

```
C: > Users > jilug > OneDrive > Desktop > Python > hangtdd.py > ...
1  import random
2  from wordslist import words
3
4  def get_word():
5      word =random.choice(words)
6      print(word)
7      return word.upper()
8  def hangman():
9      word = get_word()
10     alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
11     guessed_letter=[]
12     lives =6
13     guessed = False
14     print(len(word),'letters in the word')
15     print(len(word)*'_ ')
16
17     while guessed == False and lives > 0:
18         print(lives,'lives left')
19         guess = input('guess a letter:').upper()
20         if len (guess) == 1:
21             if guess not in alphabet:
22                 print('Invalid character')
23                 lives -=1
24             elif guess in guessed_letter:
25                 print('you have already guessed the letter')
26             elif guess not in word:
27                 print('Guessed letter not present in the word')
28                 guessed_letter.append(guess)
29                 lives -=1
30             elif guess in word:
31                 print('your guess is present in the word')
32                 guessed_letter.append(guess)
33     hangman()
```



```
PS C:\Users\jilug> & C:/Users/jilug/AppData/Local/Programs/Python/Python37/python.exe c
box
3 letters in the word
_
6 lives left
guess a letter:b
your guess is present in the word
6 lives left
guess a letter:g
Guessed letter not present in the word
5 lives left
guess a letter:5
Invalid character
4 lives left
guess a letter:d
Guessed letter not present in the word
3 lives left
guess a letter:x
your guess is present in the word
3 lives left
guess a letter:█
```

Fourth requirement: Combine the entire requirements and do the user testing

```
import random
from wordslist import words

def get_word():
    word = random.choice(words)
    return word.upper()

def hangman():
    word = get_word()
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    guessed_letter=[]
    lives =6
    guessed = False
    print(len(word),'letters in the word')
    print(len(word)*'_')

    while guessed == False and lives > 0:
        print(lives,'lives left')
        guess = input('guess a letter:').upper()
        if len(guess) == 1:
            if guess not in alphabet:
                print('Invalid character')
                lives -=1
            elif guess in guessed_letter:
                print('you have already guessed the letter')
            elif guess not in word:
                print('Guessed letter not present in the word')
                guessed_letter.append(guess)
                lives -=1
            elif guess in word:
                print('your guess is present in the word')
                guessed_letter.append(guess)
        status = ''
        if guessed == False:
            for letter in word:
                if letter in guessed_letter:
                    status += letter
                if status == word:
                    lives = 0
                    print('you win the game')
                elif letter not in guessed_letter:
                    status += '_'
            print(status)
        print('You run out of guesses')
    hangman()
```

```
PS C:\Users\jilug> & C:/Users/jilug/AppData/Local/Programs/Python/Python37/python.exe
6 letters in the word

_ _ _ _ _
6 lives left
guess a letter:e
your guess is present in the word
_ E _ _ _
6 lives left
guess a letter:t
Guessed letter not present in the word
_ E _ _ _
5 lives left
guess a letter:i
Guessed letter not present in the word
_ E _ _ _
4 lives left
guess a letter:s
Guessed letter not present in the word
_ E _ _ _
3 lives left
guess a letter:k
Guessed letter not present in the word
_ E _ _ _
2 lives left
guess a letter:o
Guessed letter not present in the word
_ E _ _ _
1 lives left
guess a letter:p
Guessed letter not present in the word
_ E _ _ _
You run out of guesses
PS C:\Users\jilug> █
```