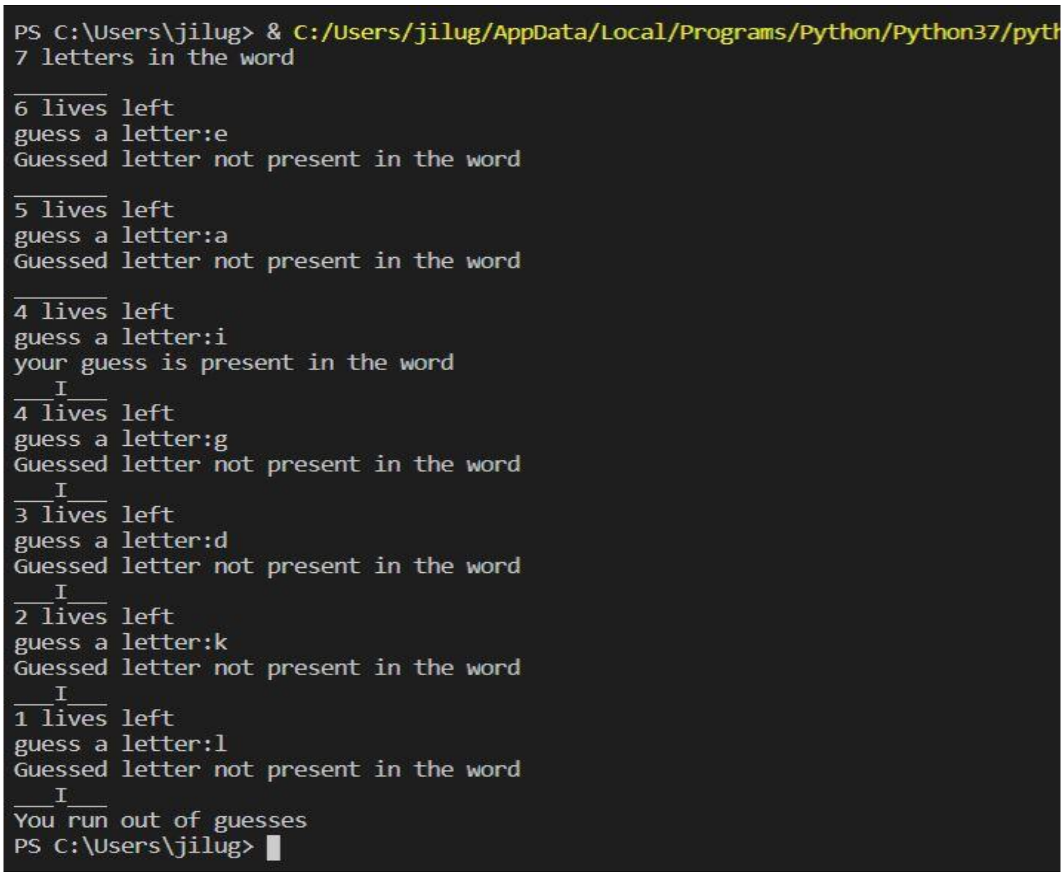
1. *A program for Hangman in python*

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

```python
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()
```

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
___I___
4 lives left
guess a letter:g
Guessed letter not present in the word
___I___
3 lives left
guess a letter:d
Guessed letter not present in the word
___I___
2 lives left
guess a letter:k
Guessed letter not present in the word
___I___
1 lives left
guess a letter:l
Guessed letter not present in the word
___I___
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

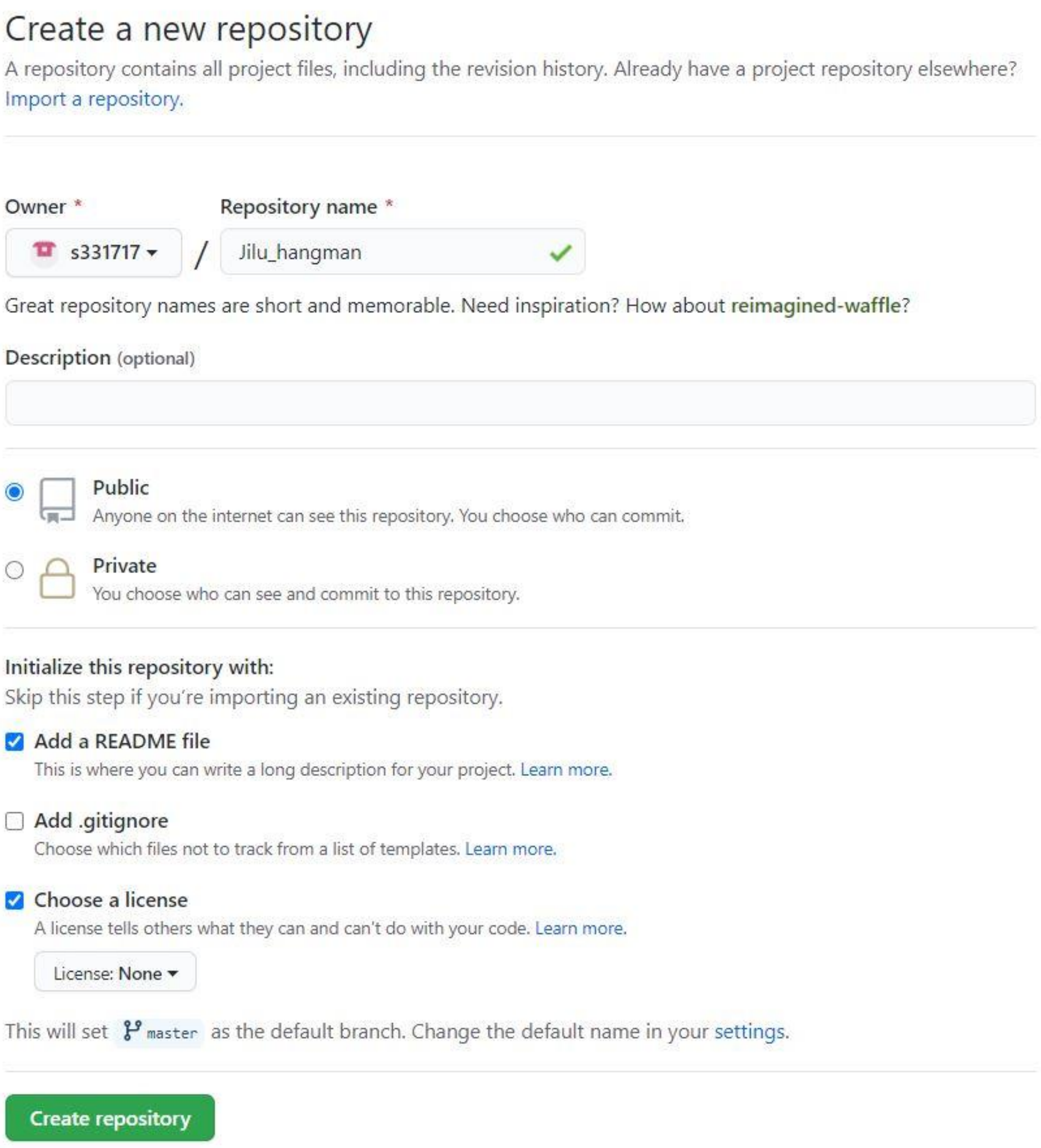
```
import random
from wordslist import words
```

```

def hangman():
    word =random.choice(words).upper()
    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()
```

3. Create a Git directory for your assignment

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

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

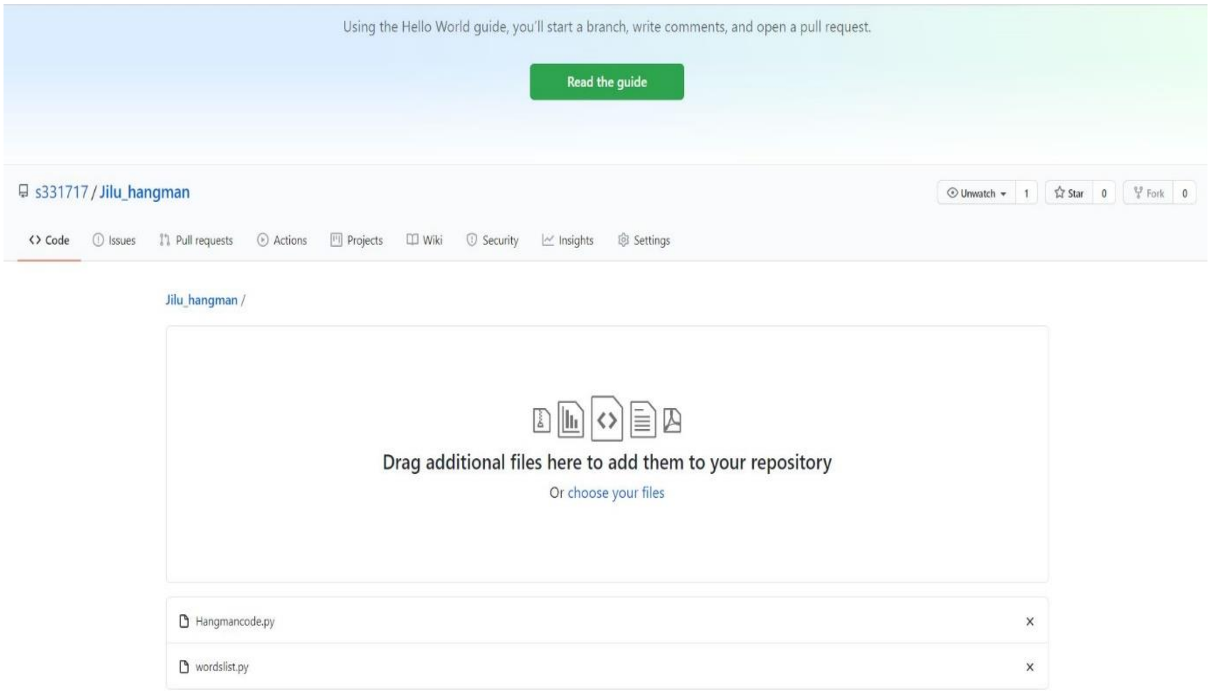
[ ] Add .gitignore
Choose which files not to track from a list of templates. Learn more.

[✓] Choose a license
A license tells others what they can and can't do with your code. Learn more.

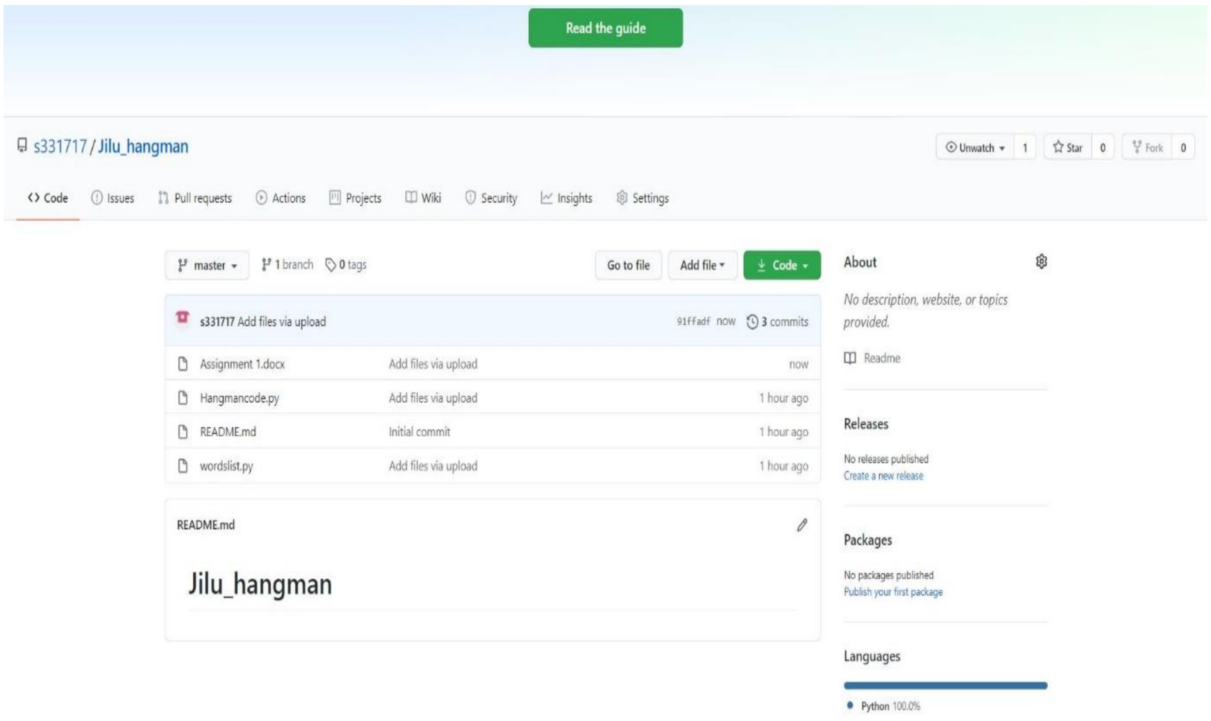
License: None

This will set `master` as the default branch. Change the default name in your settings.

Step 2: Choose your files related to the project



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



s331717 Add files via upload — 91ffadf now — 3 commits

Assignment 1.docx	Add files via upload	now
Hangmancode.py	Add files via upload	1 hour ago
README.md	Initial commit	1 hour ago
wordslist.py	Add files via upload	1 hour ago

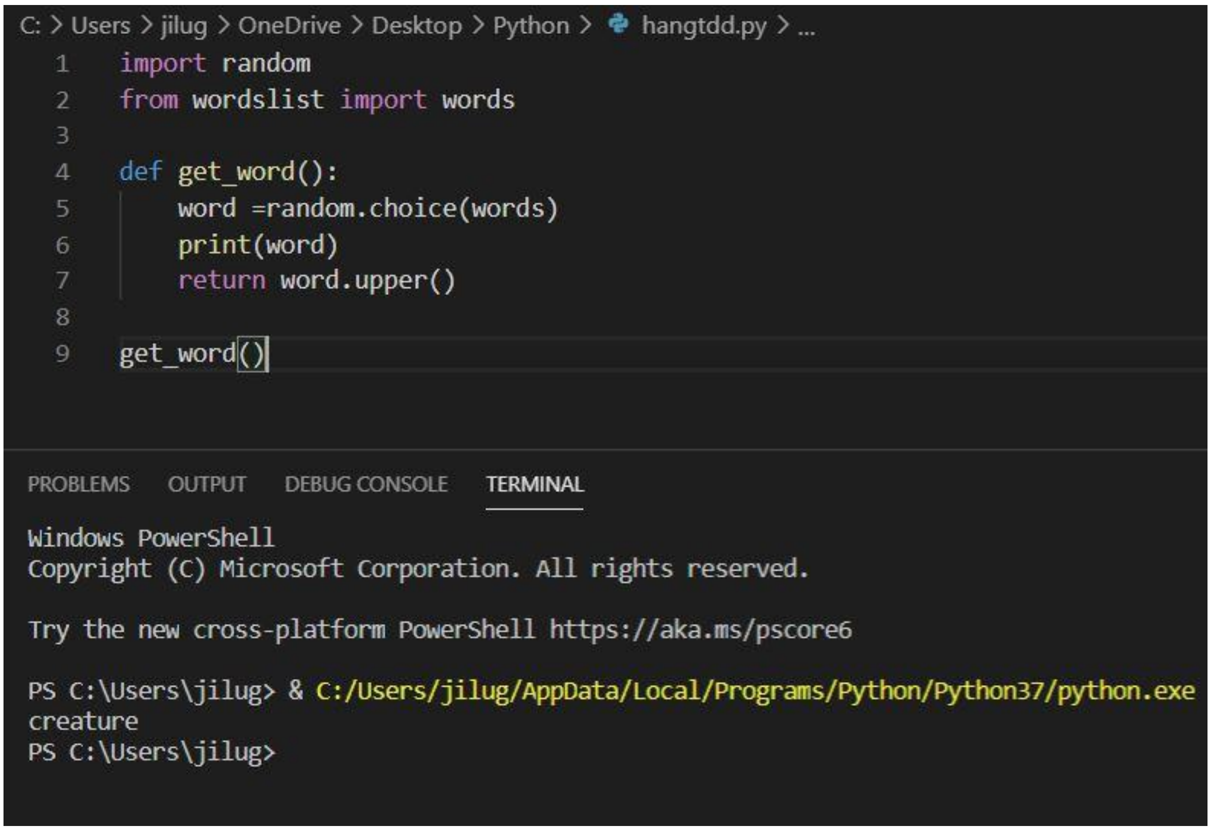
README.md

Jilu_hangman

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

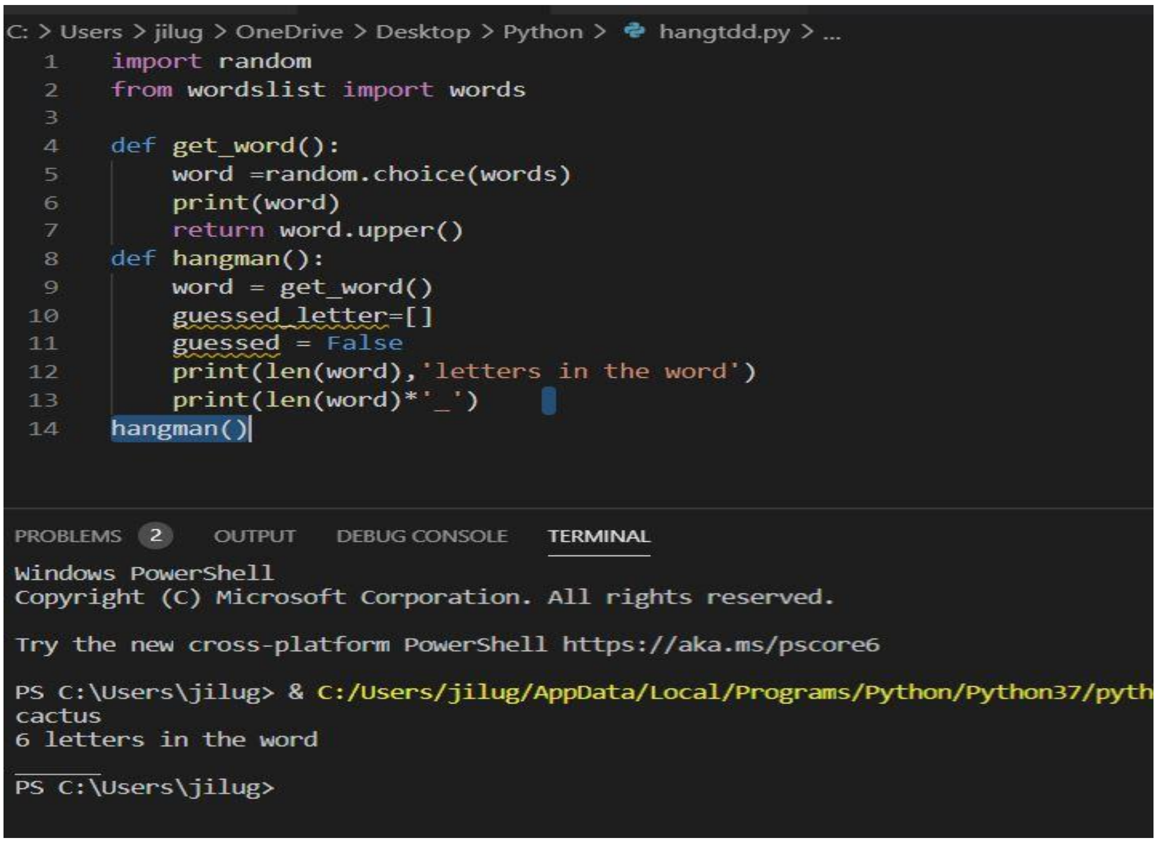
First requirement: check the random selection of word from wordlist


C: > Users > jilug > OneDrive > Desktop > Python > hangtdd.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

```python
import random
from wordslist import words

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

get_word()
```

```
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>
```

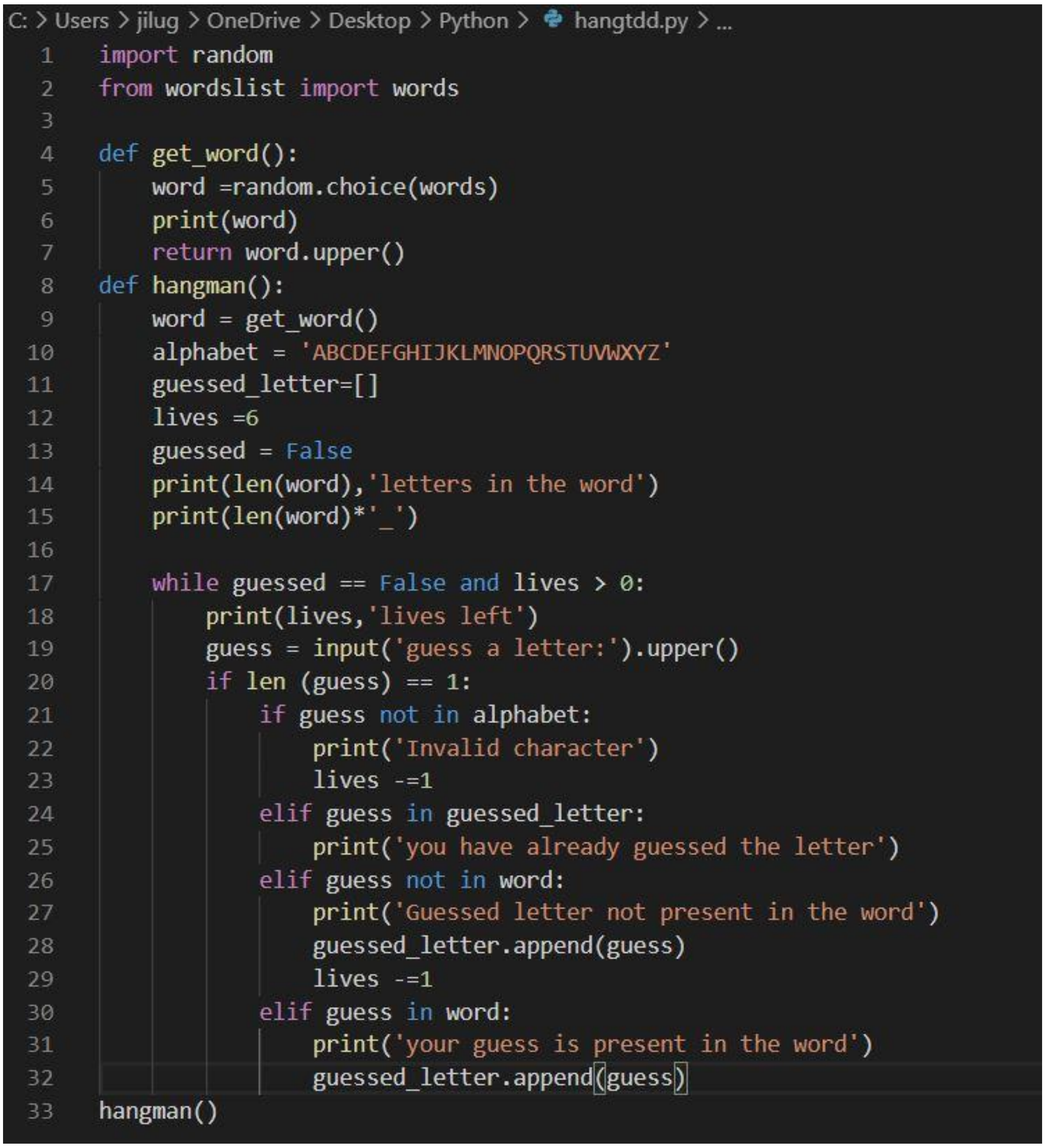
Second Requirement: To print the length of the word and ensure the blank spaces in the place of letters



```
C: > Users > jilug > OneDrive > Desktop > Python > hangtdd.py > ...
import random
from wordslist import words

def get_word():
    word =random.choice(words)
    print(word)
    return word.upper()
def hangman():
    word = get_word()
    guessed_letter=[]
    guessed = False
    print(len(word),'letters in the word')
    print(len(word)*'_')
hangman()
```

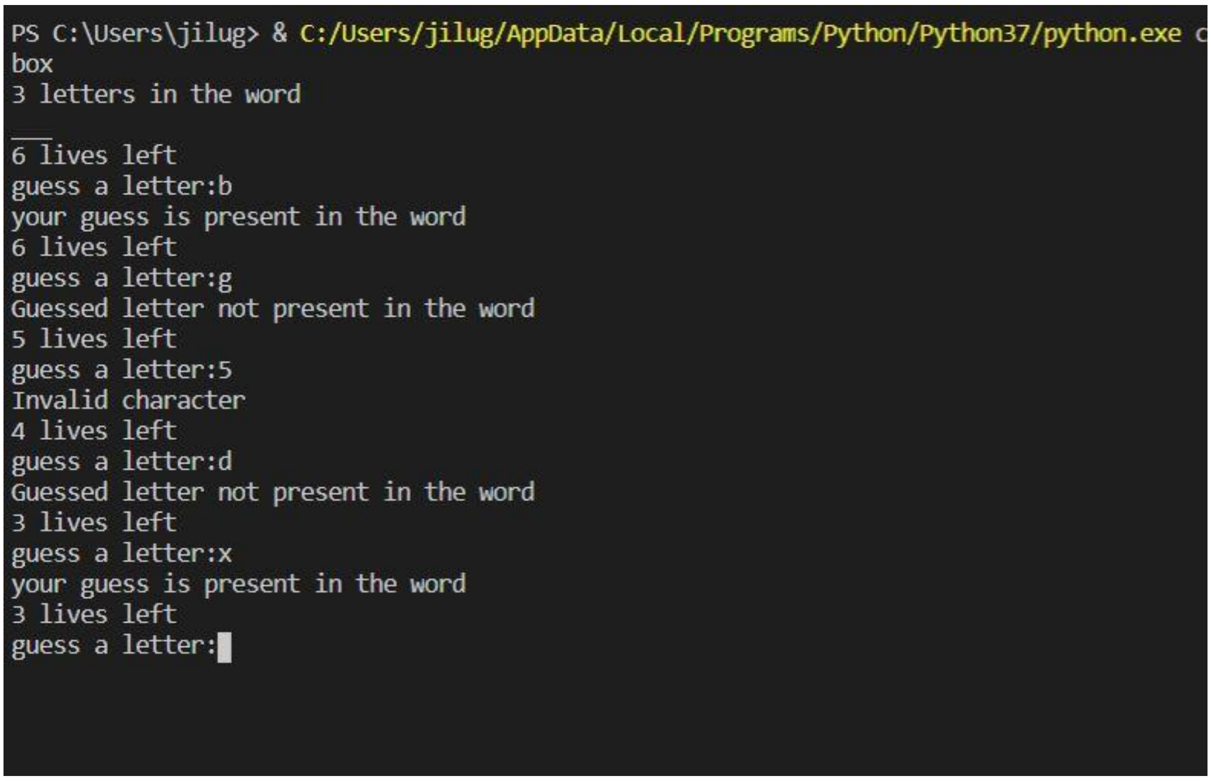
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Users\jilug> & C:/Users/jilug/AppData/Local/Programs/Python/Python37/pyth
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 > ...

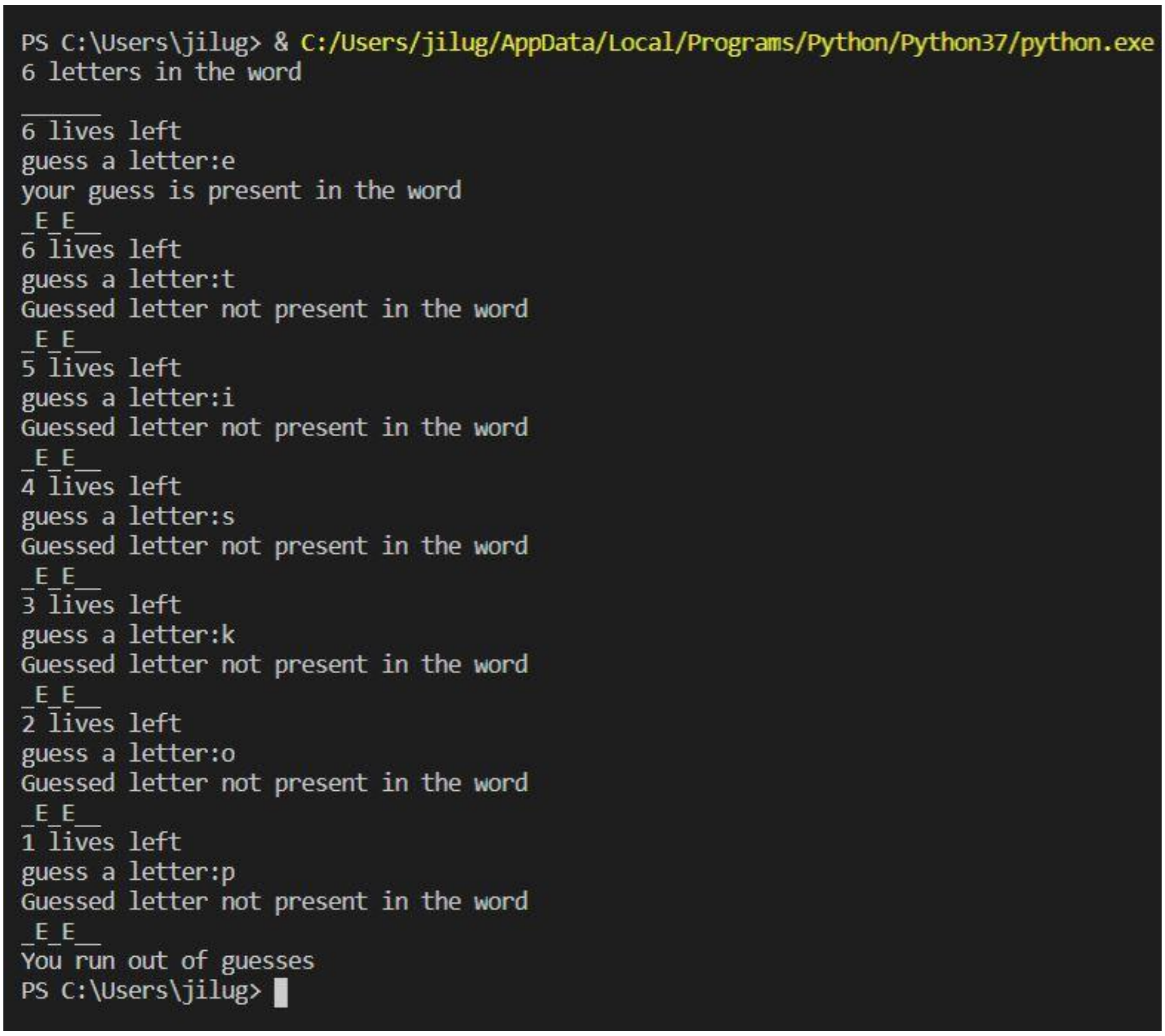
```python
import random
from wordslist import words

def get_word():
    word =random.choice(words)
    print(word)
    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)
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

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