Instruction for turning in the hangman problem

Please submit code for the following functions. You can check the output by the following test cases.

(1) `def get_random_word( ).`
The word opens the sanitized file of words and returns it.

```
for _ in range(30):
     print(get_random_word())
```

should return 30 words without hyphens and of reasonable size.

(2) `def draw_errors(nr_errors):`
Draws a picture of the hangman. For instance, `draw_errors(5)` will draw

```
+----+
|    |
|    O
|   /|\
|   /
|
|
|
|
```

or something very similar. In particular, the right arm is printed correctly.

(3) `def display_word(secret, seen):`
Parameter `secret` is a string, the secret word to be guessed. Parameter `seen` is a list of letters that have been seen. We can assume that secret and the seen list has no capital letters.

```
def display_word("ahmedabad", ["a","b","c","d"])
```
returns
```
a _ _ _ d a b a d
```

(4) `def get_letter(seen)`
This function asks the user for a letter. It ensures (by repeatedly asking if necessary) that the user returns a lower-case letter that is not in seen, the list of letters previously entered.

Test case:
```
>>> seen = []
>>> get_letter(seen)
Please enter a letter: a
'a'
>>> get_letter(seen)
Please enter a letter: b
'b'
>>> get_letter(seen)
Please enter a letter: c
'c'
>>> get_letter(seen)
Please enter a letter: d
```

```
'd'
>>> get_letter(seen)
Please enter a letter: e
'e'
>>> seen
['a', 'b', 'c', 'd', 'e']
```

(5) `def check_success(secret, seen):`
The parameter `secret` is the word to be guessed and `seen` is the list of seen letters. The functions returns `True` or `False` depending whether all letters in secret are in seen.

```
>>> seen = ['a', 'b', 'c', 'd', 'e']
>>> secret = "ahmedabad"
>>> check_success(secret, seen)
False
>>> check_success('dada', seen)
```