

# Assignment 10 – Working with Strings

## String

String consist of Unicode characters.

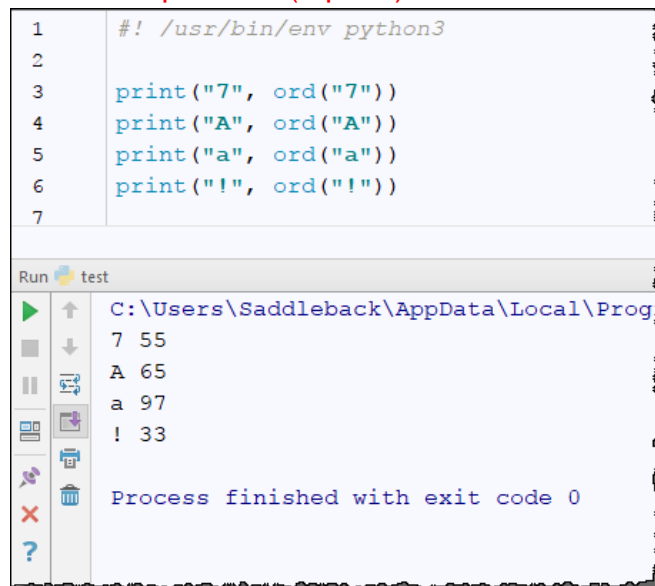
## String Functions

- `ord(char)` Returns the integer value of the Unicode character.
- `len(str)` Returns the integer value for the length of a string.

### *ord Function*

1. Code the following:

#### Screen Capture #1 (1 point)



```

1  #!/usr/bin/env python3
2
3  print("7", ord("7"))
4  print("A", ord("A"))
5  print("a", ord("a"))
6  print("!", ord("!"))
7

```

Run test

```

C:\Users\Saddleback\AppData\Local\Programs\Python\Python39\python.exe
7 55
A 65
a 97
! 33

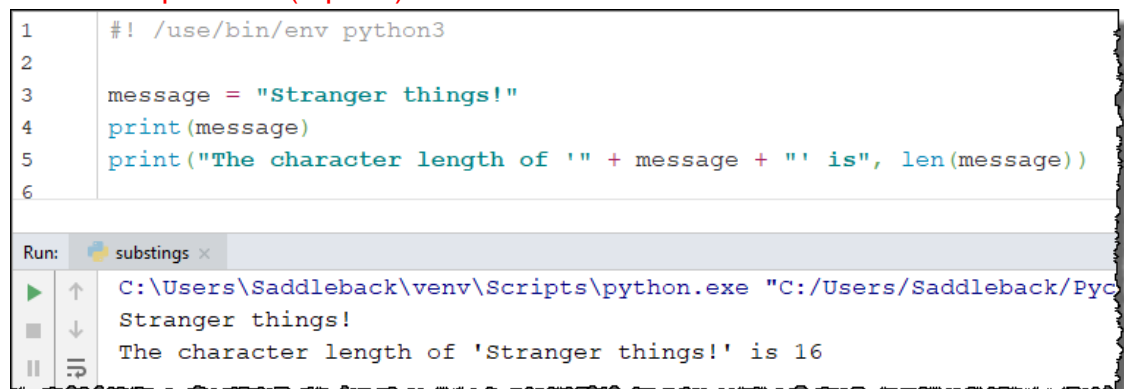
Process finished with exit code 0

```

### *len Function*

2. Code the following:

#### Screen Capture #2 (1 point)



```

1  #!/use/bin/env python3
2
3  message = "Stranger things!"
4  print(message)
5  print("The character length of '" + message + "' is", len(message))
6

```

Run: substings x

```

C:\Users\Saddleback\venv\Scripts\python.exe "C:/Users/Saddleback/Py
Stranger things!
The character length of 'Stranger things!' is 16

```

*Character String List*

3. Code the following:

**Screen Capture #3 (1 point)**

```

1  #! /use/bin/env python3
2
3  message = "Stranger things!"
4  print(message)
5  print()
6  print("message[0]  = ", message[0])
7  print("message[1]  = ", message[1])
8  print("message[3]  = ", message[3])
9  print("message[-1] = ", message[-1])
10

```

Run: substings x

C:\Users\Saddleback\venv\Scripts\python.exe  
Stranger things!  
  
message[0] = S  
message[1] = t  
message[3] = a  
message[-1] = !

4. Code the following:

**Screen Capture #4 (1 point)**

```

1  #! /use/bin/env python3
2
3  message = "Stranger things!"
4  print(message)
5  print()
6  print("message[16] = ", message[16])
7  print("message[1]  = ", message[1])
8  print("message[3]  = ", message[3])
9  print("message[-1] = ", message[-1])
10

```

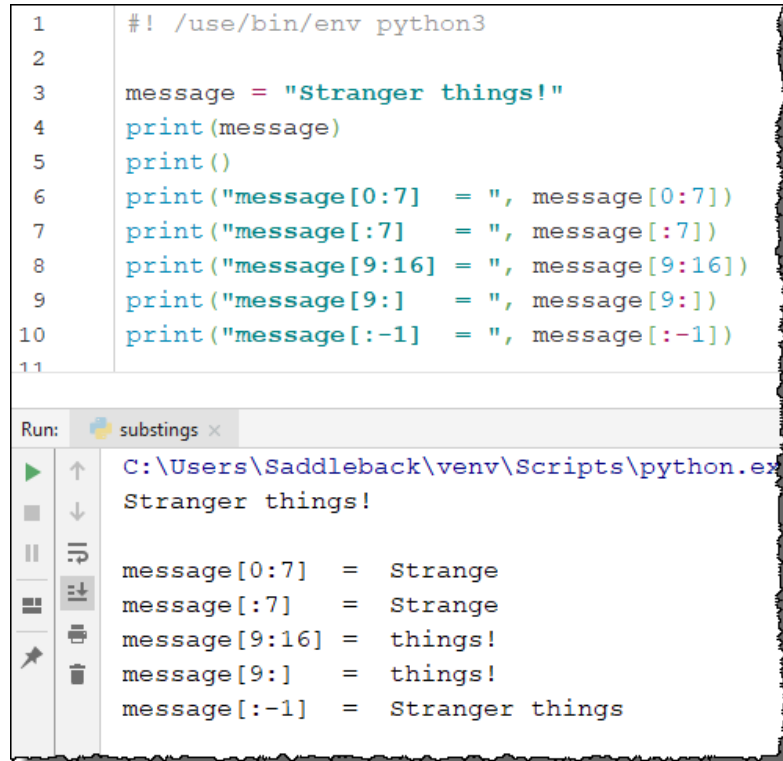
Run: substings x

C:\Users\Saddleback\venv\Scripts\python.exe "  
Stranger things!  
Traceback (most recent call last):  
  
File "C:/Users/Saddleback/PycharmProjects/A  
 print("message[16] = ", message[16])  
IndexError: string index out of range

*Substring*

This will extract pieces (substrings) from a string

5. Code the following and test

**Screen Capture #5 (1 point)**

The screenshot shows a Python script in a text editor and its execution output in a terminal window. The script defines a string 'message' and uses various slicing techniques to extract parts of it. The terminal output shows the results of these slices.

```
1  #! /use/bin/env python3
2
3  message = "Stranger things!"
4  print(message)
5  print()
6  print("message[0:7] = ", message[0:7])
7  print("message[:7] = ", message[:7])
8  print("message[9:16] = ", message[9:16])
9  print("message[9:] = ", message[9:])
10 print("message[:-1] = ", message[:-1])
11
```

Run: substings x

C:\Users\Saddleback\venv\Scripts\python.exe  
Stranger things!

message[0:7] = Strange  
message[:7] = Strange  
message[9:16] = things!  
message[9:] = things!  
message[:-1] = Stranger things

*Repeating Characters*

This will replicate strings.

6. Add the following to repeat the "=" 30 times

**Screen Capture #6 (1 point)**

```

1  #!/usr/bin/env python3
2
3  print('=' * 30)
4  print('=' * 30)
5

```

Run: movie-1 x

```

C:\Users\Saddleback\AppData\Local\Python\Scripts\python.exe "C:/Users/Saddleback/AppData/Local/Python/Scripts/python.exe"
"=" * 30
=====
Process finished with exit code 0

```

*Searching for a String*

This will search for a substring in a string

7. Code the following substring search:

**Screen Capture #7 Show both found and not found for word entry (2 points)**

```

1  #!/usr/bin/env python3
2
3  message = "Stranger things!"
4  print(message)
5  print()
6  print("Is 'things' found in message = ", "things" in message)
7  print("Is 'Things' found in message = ", "Things" in message)
8  print("Is 'hings' found in message = ", "hings" in message)
9  print("Is ' things' found in message = ", " things" in message)
10 print()
11
12 word = input("Enter a search word: ")
13 if word in message:
14     print(word, "was found in", message)
15 else:
16     print(word, "was NOT found in", message)
17

```

Run: substings x

```

C:\Users\Saddleback\venv\Scripts\python.exe "C:/Users/Saddleback/venv/Scripts/python.exe"
Stranger things!

Is 'things' found in message = True
Is 'Things' found in message = False
Is 'hings' found in message = True
Is ' things' found in message = True

Enter a search word: Science
Science was NOT found in Stranger things!

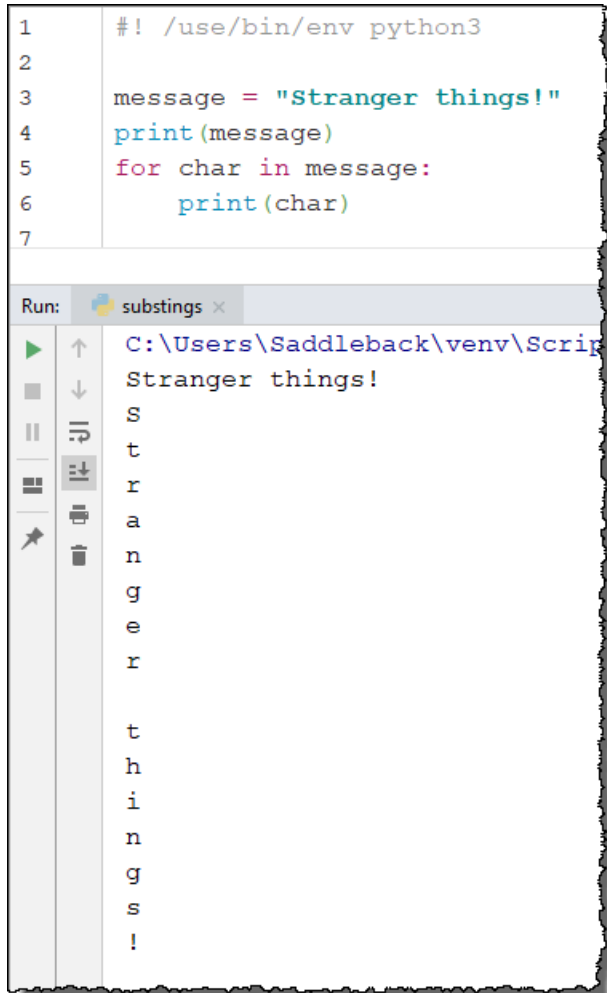
```

*Loop Through a String*

This will loop through a string one character at a time.

8. Code the following:

Screen Capture #8 (1 point)



The screenshot shows a code editor with a Python script and a terminal window below it. The script defines a message and loops through each character to print it. The terminal shows the output of the script, which is the string "Stranger things!" printed character by character on separate lines.

```
1  #! /use/bin/env python3
2
3  message = "Stranger things!"
4  print(message)
5  for char in message:
6      print(char)
7
```

Run: substings x

C:\Users\Saddleback\venv\Scripts>python3 script.py

Stranger things!

S

t

r

a

n

g

e

r

t

h

i

n

g

s

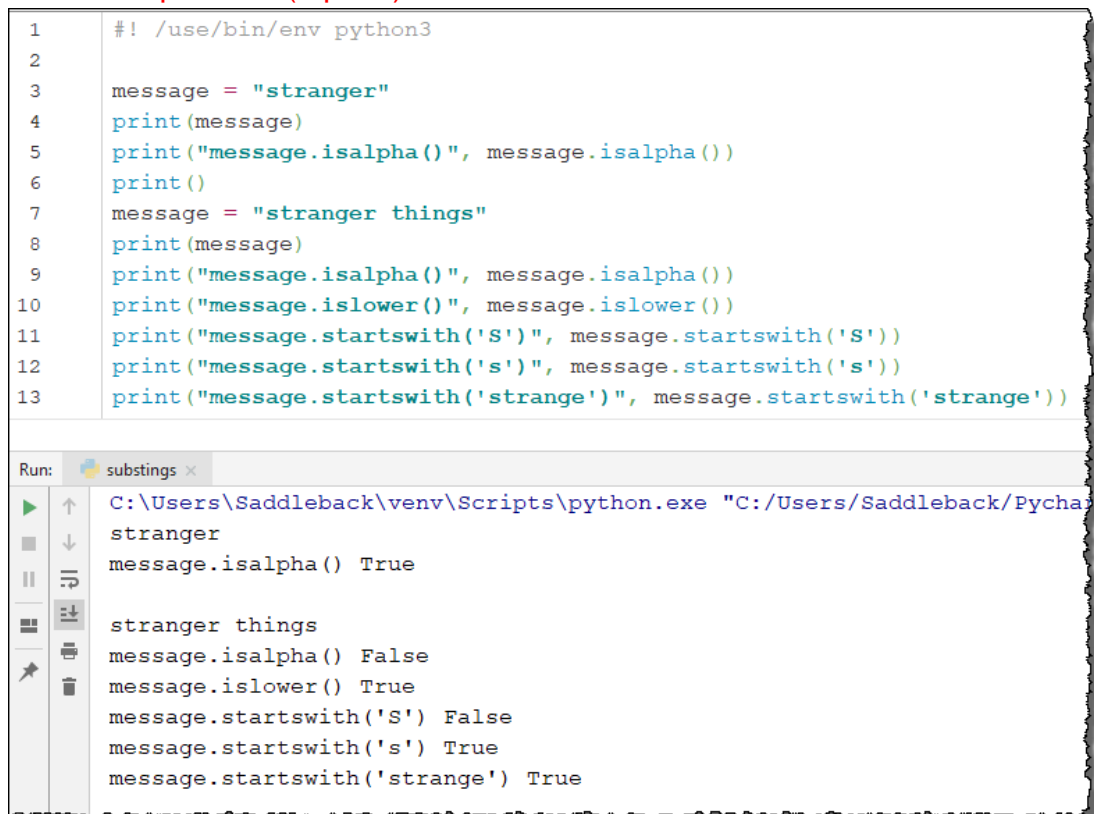
!

### String Methods

- `.isalpha()` Returns True if all characters are alphabetic.
- `.islower()` Returns True if all characters are lowercase.
- `.isupper()` Returns True if all characters are uppercase.
- `.isdigit()` Returns True if all characters are digits (0 through 9).
- `.startswith(chr)` Returns True if the string starts with a specified string
- `.endswith(chr)` Returns True if the string ends with a specified string
- `.lower()` Converts the string to all lowercase letters.
- `.upper()` Converts the string to all uppercase letters.
- `.title()` Converts the string to all title case letters.
- `.lstrip()` Removes preceding whitespace from the string.
- `.rstrip()` Removes trailing whitespace from the string.
- `.strip()` Removes preceding and trailing whitespace from the string.
- `.ljust()` Left justifies the string with trailing spaces.
- `.rjust()` Right justifies the string with preceding spaces.
- `.center()` Centers the string with preceding and trailing spaces.

9. Code the following:

#### Screen Capture #9 (1 point)



```

1  #! /use/bin/env python3
2
3  message = "stranger"
4  print(message)
5  print("message.isalpha()", message.isalpha())
6  print()
7  message = "stranger things"
8  print(message)
9  print("message.isalpha()", message.isalpha())
10 print("message.islower()", message.islower())
11 print("message.startswith('S')", message.startswith('S'))
12 print("message.startswith('s')", message.startswith('s'))
13 print("message.startswith('strange')", message.startswith('strange'))

```

Run: substings x

```

C:\Users\Saddleback\venv\Scripts\python.exe "C:/Users/Saddleback/Pychar
stranger
message.isalpha() True

stranger things
message.isalpha() False
message.islower() True
message.startswith('S') False
message.startswith('s') True
message.startswith('strange') True

```

10. Code the following:

Screen Capture #10 (1 point)

```
1  #! /use/bin/env python3
2
3  message = "Stranger things!"
4  print(message)
5  print()
6  print("message.lower() = ", message.lower())
7  print("message.upper() = ", message.upper())
8  print("message.title() = ", message.title())
```

Run: substings x

```
C:\Users\Saddleback\venv\Scripts\python.exe "C:/
Stranger things!

message.lower() =  stranger things!
message.upper() =  STRANGER THINGS!
message.title() =  Stranger Things!
```

11. Code the following:

Screen Capture #11 (1 point)

```
1  #! /usr/bin/env python3
2
3  message = "  demogorgon  "
4
5  print(message)
6  print("message.lstrip() '" + message.lstrip() + "'")
7  print("message.rstrip() '" + message.rstrip() + "'")
8  print("message.strip() '" + message.strip() + "'")
```

Run: test test

```
C:\Users\Saddleback\AppData\Local\Programs\Python\Python
demogorgon
message.lstrip() 'demogorgon '
message.rstrip() ' demogorgon'
message.strip() 'demogorgon'

Process finished with exit code 0
```

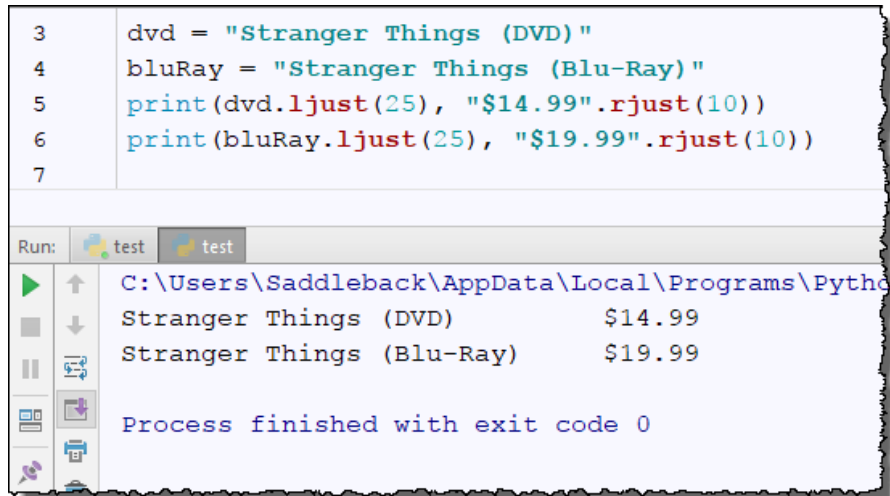
12. Code the following:

Screen Capture #12 (1 point)

```

3   dvd = "Stranger Things (DVD)"
4   bluRay = "Stranger Things (Blu-Ray)"
5   print(dvd.ljust(25), "$14.99".rjust(10))
6   print(bluRay.ljust(25), "$19.99".rjust(10))
7

```



Run: test test

C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe

Stranger Things (DVD) \$14.99

Stranger Things (Blu-Ray) \$19.99

Process finished with exit code 0

### More String Methods

- `.find(subString)` Returns index if substring is found, -1 if not.
- `.replace(old, new)` Returns string with old substrings replaced with new.
- `.split(delimiter)` Splits a string into a list of substrings based on delimiter.

### Find Method

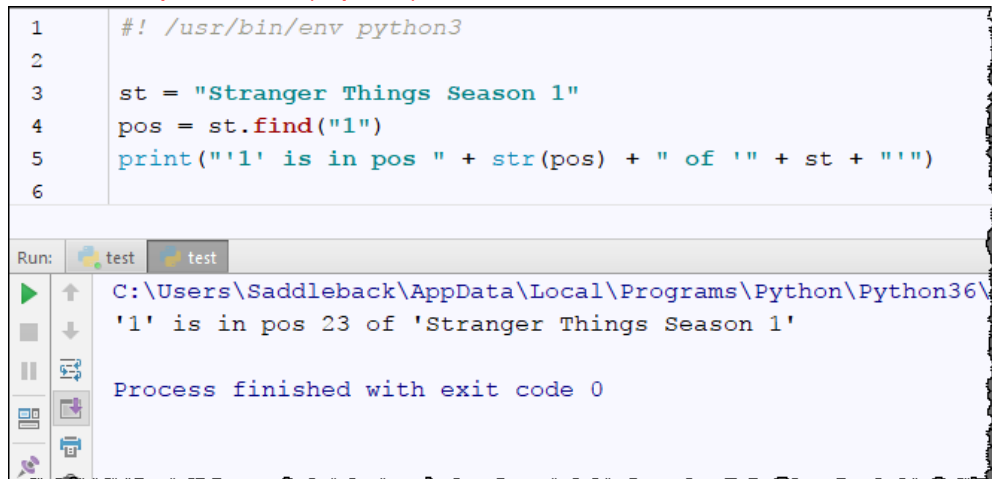
13. Code the following:

Screen Capture #13 (1 point)

```

1   #!/usr/bin/env python3
2
3   st = "Stranger Things Season 1"
4   pos = st.find("1")
5   print("'1' is in pos " + str(pos) + " of '" + st + "'")
6

```



Run: test test

C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe

'1' is in pos 23 of 'Stranger Things Season 1'

Process finished with exit code 0



*Replace Method*

14. Code the following:

Screen Capture #14 (1 point)

```
1  #!/usr/bin/env python3
2
3  st = "Stranger Things Season 1"
4  print("Before replace: " + st)
5  st = st.replace("1", "2")
6  print("After replace:  " + st)
7
```

Run: test test

C:\Users\Saddleback\AppData\Local\Programs\Python\Python39\python.exe

Before replace: Stranger Things Season 1  
After replace: Stranger Things Season 2

Process finished with exit code 0

*Split Method*

15. Code the following:

Screen Capture #15 (1 point)

```
1  #!/usr/bin/env python3
2
3  st = "Stranger Things Season 1"
4  st = st.split(" ") # " " is the default
5  for word in st:
6      print(word)
7
```

Run: test test

C:\Users\Saddleback\AppData\Local\Programs\Python\Python39\python.exe

Stranger  
Things  
Season  
1

Process finished with exit code 0

## Hangman Application

The code for the following hangman application can be found starting on page 291 of the text.

```
C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe C:/cimp110/ha
Play the H A N G M A N game

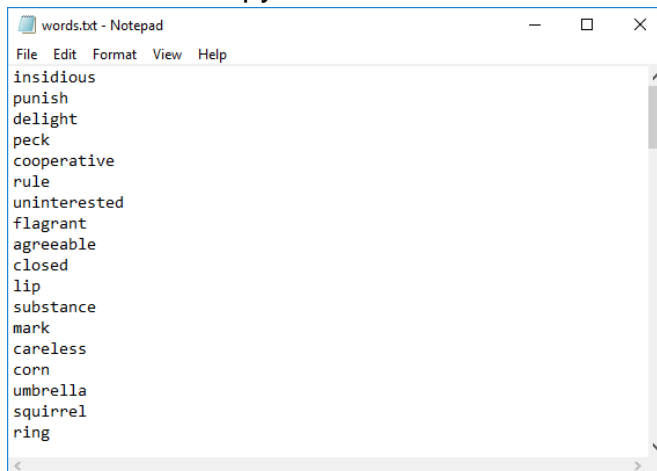
-----
Word: _ _ _ _   Guesses: 0   Wrong: 0   Tried:
Enter a letter:
```

### Setup

16. Create a new hangman folder in the cimp110 folder. This is where we will put all the files associated with this program.

### Create the Word File

17. In the hangman folder, create a text file named words.txt. Open a web browser and go to: <https://www.randomlists.com/random-words>. Create a list of 10 or more words. Copy the list into the words.txt file and save.



### Create the wordlist Module

18. Create a new python file named **wordlist.py** in the hangman folder.
19. We will need to add the random module to be able to get a random word and the os module to access the words.txt file

```
1 import random
2 import os
```

20. Create a variable with the file name

```
3
4 FILENAME = "words.txt"
5
```

21. Create a function to load the words from the file into an array.

```

6
7 def read_words():
8     try:
9         # if os.path.isfile(FILENAME):
10        words = []
11        with open(FILENAME) as f:
12            words = f.readlines()
13        return words
14    except FileNotFoundError:
15        print("Could not find " + FILENAME + " file.")
16        return ""
17    except Exception as e:
18        print(type(e), e)
19        return ""
20

```

22. Create a function to return one of the words from the list at random.

- a. Note that since the words in the file are on their own line, each word is followed by a new line character which we will need to strip off.

```

21
22 def get_random_word():
23     word = random.choice(read_words())
24     # We need to strip off the new line character
25     word = word.rstrip("\n")
26     return word
27

```

23. For testing, create the Main that will call the read\_words() function.

- a. Note: Run wordlist.py to make sure only a single word is displayed.

```

28
29 def main():
30     """
31     A simple test that will display a single
32     random word from the list of words
33     """
34     word = get_random_word()
35     print(word)
36
37
38 if __name__ == "__main__":
39     main()
40

```

**Create the hangman Module**

24. Create a new python file named hangman.py in the hangman folder.

25. We will first need to include the wordlist module

```
1  import wordlist
2
3
```

26. Create a function to get the word from the word list and force it to upercase

```
3
4  # Get a random word from the word list
5  def get_word():
6      word = wordlist.get_random_word()
7      return word.upper()
8
```

27. Create a function to add spaces between each letter of the word, this is for displaying the word

```
8
9
10 # Add spaces between letters
11 def add_spaces(word):
12     word_with_spaces = " ".join(word)
13     return word_with_spaces
14
```

28. Create a function to display the game on the screen

```
15
16 # Draw the display
17 def draw_screen(num_wrong, num_guesses, guessed_letters, displayed_word):
18     print("-" * 79)
19     print("Word:", add_spaces(displayed_word),
20           " Guesses:", num_guesses,
21           " Wrong:", num_wrong,
22           " Tried:", add_spaces(guessed_letters))
23
24
```

29. Create a function for getting the guess from the user.

- a. The guess must be a single letter that has not been previously been guessed.

```

23
24
25     # Get the next letter from the user
26 def get_letter(guessed_letters):
27     while True:
28         guess = input("Enter a letter: ").strip().upper()
29
30         # Make sure the user enters a letter and only one letter
31         if guess == "" or len(guess) > 1:
32             print("Invalid entry. " +
33                   "Please enter one and only one letter.")
34             continue
35         # Don't let the user try the same letter more than once
36         elif guess in guessed_letters:
37             print("You already tried that letter.")
38             continue
39         else:
40             return guess
41

```

30. Create a function to control the game flow.

- a. This function will start by retrieving the word
- b. Next the variables will be created and initialize
- c. Then the initial screen will be drawn on the screen

```

42
43     # The input/process/draw technique is common in game programming
44 def play_game():
45     word = get_word()
46
47     word_length = len(word)
48     remaining_letters = word_length
49     displayed_word = "_" * word_length
50
51     num_wrong = 0
52     num_guesses = 0
53     guessed_letters = ""
54
55     draw_screen(num_wrong, num_guesses, guessed_letters, displayed_word)
56

```

- d. Next, we will loop through the users guesses until the user has reached the maximum number of guesses (10) and there are no more unknown letters
  - i. Get the user guess
  - ii. Add it to the letters guessed
  - iii. See if the guess is in the word
  - iv. If it is, redisplay all the letters
  - v. If not, increment the number of wrong guesses
- e. Increment the number of guesses the user has taken
- f. Redraw the screen

```

56
57     while num_wrong < 10 and remaining_letters > 0:
58         guess = get_letter(guessed_letters)
59         guessed_letters += guess
60
61         pos = word.find(guess, 0)
62         if pos != -1:
63             displayed_word = ""
64             remaining_letters = word_length
65             for char in word:
66                 if char in guessed_letters:
67                     displayed_word += char
68                     remaining_letters -= 1
69                 else:
70                     displayed_word += "_"
71         else:
72             num_wrong += 1
73
74         num_guesses += 1
75
76         draw_screen(num_wrong, num_guesses, guessed_letters, displayed_word)
77

```

- b. Print a separator line
- c. Determine if the user won or lost and display the proper message

```

77
78     print("-" * 79)
79     if remaining_letters == 0:
80         print("Congratulations! You got it in",
81               num_guesses, "guesses.")
82     else:
83         print("Sorry, you lost.")
84         print("The word was:", word)
85
86

```

31. Lastly, add the main function

- This will create the game and prompt the user for a new game after the previous game has ended.

```

85
86
87 def main():
88     print("Play the H A N G M A N game")
89     while True:
90         play_game()
91         print()
92         again = input("Do you want to play again (y/n)?").lower()
93         if again != "y":
94             break
95
96
97 if __name__ == "__main__":
98     main()
99

```

32. Run and play the game to prove it works.

Code Validation (2 points)

Screen Capture #16 (2 points)

```

Word: _ O _ I _ _  Guesses: 6  Wrong: 4  Tried: E A I O S T
Enter a letter: m

-----

Word: _ O _ I _ _  Guesses: 7  Wrong: 5  Tried: E A I O S T M
Enter a letter: n

-----

Word: _ O _ I N _  Guesses: 8  Wrong: 5  Tried: E A I O S T M N
Enter a letter: l

-----

Word: L O _ I N _  Guesses: 9  Wrong: 5  Tried: E A I O S T M N L
Enter a letter: g

-----

Word: L O _ I N G  Guesses: 10 Wrong: 5  Tried: E A I O S T M N L G
Enter a letter: v

-----

Word: L O V I N G  Guesses: 11 Wrong: 5  Tried: E A I O S T M N L G V

Congratulations! You got it in 11 guesses.

Do you want to play again (y/n)?:

```

## Extra Credit

To get full points for each extra credit, you must include screen captures of the running output as well as the python code (.py) files.

### Extra Credit #1 – HTML Converter (+1 Extra Credit)

Complete a program that reads an HTML file and converts it to plain text.

```
HTML Converter
```

```
Grocery List
```

```
* Eggs
```

```
* Milk
```

```
* Butter
```

#### *Specifications:*

- Store the following data in a file named groceries.html:  
    <h1>Grocery List</h1>  
    <ul>  
        <li>Eggs</li>  
        <li>Milk</li>  
        <li>Butter</li>  
    </ul>
- When the program starts, it should read the contents of the file, remove the HTML tags, remove any spaces to the left of the tags, add astresks (\*) before the list items, and display the contents of the HTML tags on the console as shown above.



### Extra Credit #2 – Pig Latin Translator (+1 Extra Credit)

Create a program that translates English to Pig Latin.

```
Pig Latin Translator

Enter text: Tis but a scratch.
English:   tis but a scratch
Pig Latin: istay utbay away atchscray

Continue? (y/n): y

Enter text: We are the knights who say nee!
English:   we are the knights who say nee
Pig Latin: eway areway ethay ightsknay owhay aysay eenay

Continue? (y/n): n

Bye!
```

#### Specifications:

- Convert the English to lowercase before translating.
- Remove any punctuation characters before translating.
- Assume that words are separated from each other by a single space.
- If the word starts with a vowel, just add **way** to the end of the word.
- If the word starts with a consonant, move all of the consonants that appear before the first vowel to the end of the word, then add a **ay** to the end of the word.
- If a word starts with the letter **y**, the **y** should be treated as a consonant. If the **y** appears anywhere else in the word, it should be treated as a vowel.

#### Note

- There are no official rules for Pig Latin. Most people agree on how words that begin with consonants are transformed, but there are many different ways to handle words that begin with vowels.