008: Hangman Game

Learning Outcome:	Using custom Modules; Create a Hangman game on console.

Definitions/Concepts	
Custom Modules	• We can create our own modules in python and reuse it later when required, such modules are called custom modules. • The python code files have extension **.py** • These code files can be used as modules.
Using a custom Module	• Make sure the python file you're working on and the one to be used as a module are in same folder on your computer. Syntax: *from module_name import function/variable_name*

Assume these two python files in a folder:

The **wordfile.py** contains a list named **words**

wordfile.py

```
1  words=['able',
2  'about',
3  'above',
```

To use the list named **words** of **wordfile.py** in **main.py** we will write:

```
from wordfile import words
```

Now we can directly use **words.**

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

Hangman Game
Rules of the game: How to play Hangman (Youtube link)

Code snippets	

Creating the function **getWord** to return a random word from the list **words**.	`def getWord():`
	`  from wordfile import words`
	`  import random`
	`  return random.choice(words)`

Using the above function to *get a random word* and *display the message of the game.*	`word=getWord()`
	`word=list(word)`
	`print("You have to guess this word.\n",`
	`"- "*len(word))`

Creating a list **answer** (with dashes) which will be updated to store the correctly guessed letters of the word.	`answer="_"*len(word)` `answer=list(answer)`
Taking input from the user.	`guess=input("Enter your guess: ")`

Checking if the guessed letter is in the word or not, if it is present in the word, then updating the list **answer** with the correct guesses of the letter, otherwise printing a Try again message.	`if guess in word:`
	`  for i in range(len(word)):`
	`  if word[i]==guess:`
	`  answer[i]=guess`
	`  print("Correct guess")`
	`  print(" ".join(answer))`
	`else:`
	`  print("Wrong guess, Try again")`

To check if the guessed word is same as the original word after the guesses, and display the winning message	`if answer==word:` `  print("You completed the word, Hurray!!")`

The user has to repeatedly input letters as guesses, so we need to put the last three code snippets in a while loop, which will run until the whole word is guessed, i.e. *answer* becomes the same as *word*.

```
while True:
  if answer==word:
    print("You completed the word,
    Hurray!!")
    break

  guess=input("Enter your guess: ")

  if guess in word:
    for i in range(len(word)):
      if word[i]==guess:
        answer[i]=guess
        print("Correct guess")
        print(" ".join(answer))
  else:
    print("Wrong guess, Try again")
```

- while loop here has True as its condition, so it will run until we stop it explicitly.
- When **answer** is same as **word,** we need to come out of the loop, for that we use the keyword **break**

Extra functions used in this code.

`answer="_"*len(word)`	• This creates a String with no. of dashes. • The no. of dashes is the length of word. • Multiplying a string with a number forms a string with the repeated string.

`answer=list(answer)`	• This converts the **answer** from a String into a list. • So each character of the String becomes the element of the list.
`print(" ".join(answer))`	• **answer** is a list • This function will display the elements of the list with spaces between them.

Do you know?

In 2015, Python overtook French to be the most popular languages that are taught in primary schools.