1. Escape characters are special characters that are used to represent certain **non-printable or special characters within a string**. They are denoted by **a backslash (\)** followed by a specific character or sequence
   commonly used escape characters in Python:

   \n: Newline character
   \t: Tab character
   \': Single quote character
   \": Double quote character
   ****: Backslash character
   \b: Backspace character
   \r: Carriage return character

   These escape characters can be used in python as following :

   # Newline character
   print("Hello\nthere")

   # Tab character
   print("Name:\tajay")

   # Single and double quotes
   print('Ram said, "Hi"')
   print("He\'s there.")

   # Backslash
   print("This is a backslash: \\")

   # Backspace
   print("Hello\bGames") #

   # Carriage return
   print("Python\rGood")

   Output:

   Hello (using '\n')
   there

   Name: John (using'\t')

   Ram said, "Hi" **(**using \\")
   He's there. (using\\')

   **This is a backslash: ** (using\\)

HellGames **(**using'\r')

Good (using '\b')

2. **Escape characters:-**

\n: Newline character(prints character/ string in next line after using this)
\t: Tab character(inserts horizontal tab after string, where we have used it)

Newline character
print("Hello\nthere")

output:-
hello
there

Tab character
print("Name:\tajay")
Name: ajay

3. To include a backslash character **(\)** in a string, you can use a **double backslash (\\).** The double backslash serves as an escape sequence where the first backslash acts as an escape character, and the second backslash is the literal backslash you want to include in the string

4. The single quote character in the word **Howl's** is not escaped because the string literal is enclosed in double quotes ("). In Python, single quotes can be used to define string literals, and it is not necessary to escape a single quote character if the string itself is enclosed in double quotes.

   However if 'Howl's Moving Castle' is enclosed in single quotes, it will give syntax error because the single quote character in the word **"Howl's"** is not escaped

   To correct the string, we can escape the single quote using backlash operator**('\')** in following way :-

   print('Howl\'s Moving Castle')

5. If you don't want to use the newline character (\n) directly in a string, you can use triple quotes **(''' or """)** to create a multi-line string

multi_line = """Line 1
Line 2
Line 3"""
print(multi_line)

output:-

Line 1
Line 2
Line 3

6.
Expression: **'Hello, world!'[1]**
Value: **'e'**

Expression: **'Hello, world!'[0:5]**
Value: **'Hello'**

Expression: **'Hello, world!'[:5]**
Value: **'Hello'**

Expression: **'Hello, world!'[3:]**
Value: **'lo, world!'**

7. **'Hello'.upper():** This expression will return '**HELLO'**
'Hello'.upper().isupper(): This expression will return the boolean value **True.**
'Hello'.upper().lower(): This expression will return the string **'hello'.**

8. 'Remember, remember, the fifth of July.'.split():- The value of this expression is [**'Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']**

'-'.join('There can only one.'.split()):The value of this expression is **'There-can-only-one.'.**

9. There are three methods for right-justifying, left-justifying, and centering a string:

rjust(): This method right-justifies the string to a specified width.
ljust(): This method left-justifies the string to a specified width.
center(): This method centers the string to a specified width.
These methods all take two arguments: the string to be justified and the width of the string. The width can be specified as an integer or a string
For example :- consider the code :-

```
text = "Hello"
Right_justified = text.rjust(10)
print(Right_justified)
```

 # Output: "   Hello"

10. We use the **str.strip()** method to remove whitespace characters from the start and end of a string. This method returns a new string with leading and trailing whitespace characters removed

Consider the code :

```
text = "   Hello World   "
stripped_text = text.strip()
print(stripped_text)

# Output: "Hello World"
```

If only leading whitespace characters are to be removed, then you can use the **str.lstrip()** method, and if only trailing whitespace characters are to be removed,use the **str.rstrip()** method