**1. What are escape characters, and how do you use them?**

Ans No.1:

Escape characters are used to format the code within strings. They are represented by a backslash ( \ ) followed by another character. Following are some examples of escape characters along with its uses:  
- ‘\n’: Used to enter a new line.

- ‘\r’: Used to move the cursor to the beginning of the line.

- ‘\”’: Used to give double quotes.

**2. What do the escape characters n and t stand for?**

Ans No.2:

- \n: It stands for newline character and is used for the cursor to enter a newline.

- \t: It stands for tab character and is used to give tab spaces.

**3. What is the way to include backslash characters in a string?**

Ans No.3:

There are two ways to include a backslash character in a string:

1. We can use a backslash character before the backslash.

For example,

Input: print("C:\\Users\\Fariz\\Documents")

Output: C:\Users\Fariz\Documents

Note that two backslashes will print a single backslash as one of them will be considered an escape sequence.

2. Using a raw string.

For example,

Input: print(r"C:\Users\Fariz\Documents")

Output: C:\Users\Fariz\Documents

Note that the raw string ( r ) before the string will help the program to understand not to interpret any escape sequences in the string, in order to include backslashes without needing to escape them.

**4. The string "Howl's Moving Castle" is a correct value. Why isn't the single quote character in the word Howl's not escaped a problem?**

Ans No.4:

"Howl's Moving Castle" is a correct value as it is enclosed in double quotes ( “ ” ). In Python, when double quotes are used then single quotes aren’t considered delimiters for a string.

You only need to escape quotes within a string if they use the same type of quote as the one used to enclose the string.

**5. How do you write a string of newlines if you don't want to use the n character?**

Ans No.5:  
We can enclose a string into a multiline string literal represented by triple quotes('''). For example,

Input: mul\_string = """This is a

multiline

string"""

print(mul\_string)

Output: This is a

multiline

string

Note that the newlines given in the input are considered same in the output. You can use as many newline characters as you want in a multiline string, and they will be preserved when the string is printed or otherwise used in your code.

**6. What are the values of the given expressions?**

'Hello, world!'[1]

'Hello, world!'[0:5]

'Hello, world!'[:5]

'Hello, world!'[3:]

Ans No.6:

'Hello, world!'[1] - 'e'

'Hello, world!'[0:5] - 'Hello'

'Hello, world!'[:5] - 'Hello'

'Hello, world!'[3:] - 'lo, world!'

**7. What are the values of the following expressions?**

'Hello'.upper()

'Hello'.upper().isupper()

'Hello'.upper().lower()

Ans No.7:

‘HELLO’

True

‘ hello’

**8. What are the values of the following expressions?**

'Remember, remember, the fifth of July.'.split()

'-'.join('There can only one.'.split())

Ans No.8:  
['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']

'There-can-only-one.'

**9. What are the methods for right-justifying, left-justifying, and centering a string?**

Ans No.9:  
Following are the methods for justification:  
Right – justifying = str.rjust(length,fillchar)

Example:

# Will print a total of 10 characters including ‘Hello’ and will print remaining characters on right as given in the character fill.

Input:

string = "Hello"

result = string.rjust(10,'-')

print(result)

Output:

-----Hello

Left – justifying = str.ljust(length,fillchar)

Example:

# Will print a total of 10 characters including ‘Hello’ and will print remaining characters on left as given in the character fill.

Input:

string = "Hello"

result = string.ljust(10,'-')

print(result)

Output:

Hello-----

Centering String = str.center(length,fillchar)

Example:

# Will print a total of 10 characters including ‘Hello’ and will print remaining characters to centralize a string as given in the character fill.

Input:

string = "Hello"

result = string.center(10,'-')

print(result)

Output:

--Hello---

**10. What is the best way to remove whitespace characters from the start or end?**

Ans No.10:

To remove whitespace characters from both sides we can use .strip(). To remove leading whitespaces we can use .lstrip(). To remove trailing whitespaces we can use .rstrip().

Example:

Input:

string = " -Hello, world!- "

result1 = string.rstrip() # Removes trailing whitespace

result2 = string.lstrip() # Removes leading whitespace

result3 = string.strip() # Removes leading and traling whitespace

print(f"{result1},{result2},{result3}")

Output:

-Hello, world!-,-Hello, world!- ,-Hello, world!-