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

Ans:

Escape characters are special characters used to represent certain actions or characters in a text string. They are typically preceded by a backslash character and allow us to include characters in strings that are otherwise difficult or impossible to include, such as double quotes or newlines.

An example that demonstrates using escape characters to include double quotes within a string:

"I said, \"Hello!\""

In this example, the backslash before each double quote indicates that the double quote is part of the string, rather than the end of the string. The output of this string would be:

I said, "Hello!"

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

Ans:

* "\n" stands for newline. It is used to represent the end of a line of text and the beginning of a new line. When used in a string, it creates a line break.
* "\t" stands for tab.

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

Ans:

To include a backslash character () itself in a string, you can use an escape sequence to represent the backslash character. The escape sequence for a backslash is "\".

For example, consider the following string:

"I am a string with a backslash \\ character"

The output of this string would be:

I am a string with a backslash \ character

In this way, we can include backslash characters in a string by using the backslash escape sequence "\".

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: Because in Python single quote inside a double code will be taken as it is it will not require any escape character.

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

Ans:

By using multiline string literals. In Python, for example, a multiline string literal is created by enclosing the string within triple quotes ("""string""").

Here is an example of using a multiline string literal to create a string with newlines:

string\_with\_newlines = """This is the first line.

This is the second line.

This is the third line."""

In this example, the triple quotes denote the start and end of the string, and the newlines are included directly within the string without using the "\n" escape sequence.

The output of this string would be:

This is the first line.

This is the second line.

This is the third line.

6. What are the values of the given expressions?

'Hello, world!'[1]

'Hello, world!'[0:5]

'Hello, world!'[:5]

'Hello, world!'[3:]

Ans:

'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:

'Hello'.upper() --> 'HELLO'

'Hello'.upper().isupper() --> True

'Hello'.upper().lower() --> 'hello'

8. What are the values of the following expressions?

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

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

Ans:

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

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

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

Ans:

ljust(): left-justifies a string within a specified width by padding it with spaces on the right

rjust(): right-justifies a string within a specified width by padding it with spaces on the left

center(): centers a string within a specified width by padding it with spaces on both sides equally.

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

Ans:

The best way to remove whitespace characters from the start or end of a string is to use the strip() method. The strip() method removes all whitespace characters (spaces, tabs, and newlines) from the beginning and end of the string, leaving the actual text intact.