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

Ans: Escape characters are special characters in text and code that are preceded by a backslash (\) to represent characters or sequences that are difficult to type directly.

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

Ans: The escape character \n stands for "newline," and the escape character \t stands for "tab."

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

Ans: To include a backslash character (\) in a string in Python, you need to escape it by using another backslash before it. In other words, you use \\ to represent a single backslash within a string.

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: The string "Howl's Moving Castle" is a correct value in Python because the single quote character within the word "Howl's" is not causing a problem, thanks to the use of double quotes (") to enclose the string.

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

Ans: If you want to create a string with newlines without using the \n escape sequence, you can use triple-quoted strings, which allow you to include newlines directly. Triple-quoted strings are enclosed in either triple single quotes (''') or triple double quotes (""") and can span multiple lines.

6. What are the values of the given expressions?

'Hello, world!'[1] - This expression accesses the character at index 1 in the string, which is 'e'.

'Hello, world!'[0:5] - This expression slices the string from index 0 (inclusive) to index 5 (exclusive), so it includes characters at indices 0, 1, 2, 3, and 4. The result is 'Hello'.

'Hello, world!'[:5] - This expression slices the string from the beginning (index 0) up to index 5 (exclusive), so it includes characters at indices 0, 1, 2, 3, and 4. The result is 'Hello', which is the same as the previous expression.

'Hello, world!'[3:] - This expression slices the string starting from index 3 (inclusive) to the end of the string. It includes characters at indices 3 and beyond. The result is 'lo, world!'.

7. What are the values of the following expressions?

'Hello'.upper() - This expression converts the string to uppercase, resulting in 'HELLO'.

'Hello'.upper().isupper() - In this expression, we first convert the string to uppercase with 'Hello'.upper(), resulting in 'HELLO'. Then, we use the isupper() method, which checks if all characters in the string are uppercase. Since all characters in 'HELLO' are indeed uppercase, this expression evaluates to True.

'Hello'.upper().lower() - In this expression, we first convert the string to uppercase with 'Hello'.upper(), resulting in 'HELLO'. Then, we use the lower() method, which converts all characters in the string to lowercase. So, the final result is 'hello'.

8. What are the values of the following expressions?

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

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

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

Ans:

1. Right-Justify: str.rjust(width, fillchar) adds padding on the left.
2. Left-Justify: str.ljust(width, fillchar) adds padding on the right.
3. Center: str.center(width, fillchar) adds padding on both sides.

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

Ans: The best way to remove whitespace characters (such as spaces, tabs, and newline characters) from the start or end of a string in Python is to use the str.strip() method.