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

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

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

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

6. What are the values of the given expressions?

'Hello, world!'[1]

'Hello, world!'[0:5]

'Hello, world!'[:5]

'Hello, world!'[3:]

7. What are the values of the following expressions?

'Hello'.upper()

'Hello'.upper().isupper()

'Hello'.upper().lower()

8. What are the values of the following expressions?

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

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

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

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

ANSWER

1. Escape characters are special characters used in strings to represent characters that are difficult to include directly, such as newline, tab, or special characters like quotation marks. They are preceded by a backslash (\) in the string.

2. The escape characters '\n' stands for a newline, and '\t' stands for a tab.

3. To include a backslash character in a string, you need to escape it with another backslash. For example, if you want to include a single backslash in a string, you would write it as `'\\'`.

4. The single quote character in the string "Howl's Moving Castle" is not escaped because it is enclosed within double quotes. In Python, you can use either single quotes or double quotes to define strings, and if the string is enclosed in one type of quote, you can include the other type of quote character within the string without escaping it.

5. If you don't want to use the '\n' character to represent newlines, you can include actual newline characters by using triple-quoted strings. For example:

```python

multi\_line\_string = '''This is a

multi-line

string.'''

```

6. The values of the given expressions are:

- `'Hello, world!'[1]` returns the character at index 1, which is 'e'.

- `'Hello, world!'[0:5]` returns the substring from index 0 to 4, which is 'Hello'.

- `'Hello, world!'[:5]` is the same as the previous expression and returns 'Hello'.

- `'Hello, world!'[3:]` returns the substring from index 3 to the end, which is 'lo, world!'.

7. The values of the following expressions are:

- `'Hello'.upper()` returns 'HELLO' because it converts the string to uppercase.

- `'Hello'.upper().isupper()` returns `True` because the uppercase version of 'Hello' is in all uppercase.

- `'Hello'.upper().lower()` returns 'hello' because it first converts to uppercase and then to lowercase.

8. The values of the following expressions are:

- `'Remember, remember, the fifth of July.'.split()` splits the string into a list of words, separated by whitespace. It returns `['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']`.

- `'-'.join('There can only one.'.split())` splits the string into words and then joins them with '-' as the separator. It returns 'There-can-only-one.'.

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

- Right-justifying: `str.rjust(width, fillchar)` where `width` is the desired width of the string, and `fillchar` is the character used for padding on the left.

- Left-justifying: `str.ljust(width, fillchar)` where `width` is the desired width of the string, and `fillchar` is the character used for padding on the right.

- Centering: `str.center(width, fillchar)` where `width` is the desired width of the string, and `fillchar` is the character used for padding on both sides.

10. The best way to remove whitespace characters from the start or end of a string is to use the `strip()` method. For example:

- `str.strip()` removes leading and trailing whitespace.

- `str.lstrip()` removes leading whitespace.

- `str.rstrip()` removes trailing whitespace.