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?

Answers:

1. Escape characters are special characters in strings that are used to represent certain non-printable or special characters. They are preceded by a backslash (\) in the string. For example, `\n` represents a newline character, `\t` represents a tab character, etc.

2. The escape character `\n` stands for a newline character, which creates a line break in the string. The escape character `\t` stands for a tab character, which creates horizontal spacing or indentation.

3. To include a backslash character itself in a string, you can use a double backslash (`\\`). For example, `'This is a backslash: \\\\'` would produce the string `'This is a backslash: \\'`.

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

5. If you don't want to use the `\n` character to represent newlines, you can use triple quotes (`'''` or `"""`) to create a multiline string. Each line in the string will be considered as a separate line. For example:

```

multiline\_string = '''

This is line 1.

This is line 2.

This is line 3.

'''

```

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 (excluding 5), which is `'Hello'`.

- `'Hello, world!'[:5]` is equivalent to `'Hello, world!'[0:5]` and returns `'Hello'`.

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

7. The values of the following expressions are:

- `'Hello'.upper()` returns the string `'HELLO'`, which is the uppercase version of the original string.

- `'Hello'.upper().isupper()` returns `True` because the string `'HELLO'` is entirely uppercase.

- `'Hello'.upper().lower()` returns the string `'hello'`, which is the lowercase version of the uppercase string `'HELLO'`.

8. The values of the following expressions are:

- `'Remember, remember, the fifth of July.'.split()` splits the string into a list of substrings based on whitespace characters. The resulting list is `['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']`.

- `'-'.join('There can only one.'.split())` splits the string into a list of substrings based on whitespace characters, and then joins them back using a hyphen (`-`) as the separator. The resulting string is `'There-can-only-one.'`.

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

- `string.rjust(width)` right-justifies the string within a field of specified width by adding spaces on the left side if necessary.

- `string.ljust(width)` left-justifies the string within a field of specified width by adding spaces on the right side if necessary.

- `string.center(width)` centers the string within a field of specified width by adding spaces equally on both sides if necessary.

10. The best way to remove whitespace characters from the start or end of a string is by using the `string.strip()` method. It

removes leading and trailing whitespace characters from the string. If you only want to remove leading whitespace, you can use `string.lstrip()`, and for trailing whitespace, you can use `string.rstrip()`.