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

Escape characters are special characters in a string that change the interpretation of the characters that follow them. They are typically introduced by a backslash (`\`). When the interpreter encounters an escape character, it understands that the next character should be treated differently than normal.

#### Examples of usage:

- `\n` for a newline

- `\t` for a tab

- `\\` for a literal backslash

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

- `\n`: Newline. Moves the cursor to the next line.

- `\t`: Tab. Inserts a horizontal tab space.

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

To include a backslash character in a string, you need to escape it by using a double backslash (`\\`).

#### Example:

```python

print("This is a backslash: \\")

```

### 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?

The string "Howl's Moving Castle" uses double quotes to enclose the string, which allows the single quote (`'`) within the string to be used without escaping. If single quotes were used to enclose the string, the internal single quote would need to be escaped.

#### Example:

```python

string = "Howl's Moving Castle" # Using double quotes outside

# vs.

string = 'Howl\'s Moving Castle' # Escaping the single quote inside

```

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

To write a string of newlines without using the `\n` character, you can use triple quotes (either `"""` or `'''`).

#### Example:

```python

string = """This is line one.

This is line two.

This is line three."""

```

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

- `'Hello, world!'[1]`:

- Value: `'e'`

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

- Value: `'Hello'`

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

- Value: `'Hello'`

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

- Value: `'lo, world!'`

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

- `'Hello'.upper()`:

- Value: `'HELLO'`

- `'Hello'.upper().isupper()`:

- Value: `True`

- `'Hello'.upper().lower()`:

- Value: `'hello'`

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

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

- Value: `['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']`

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

- Value: `'There-can-only-one.'`

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

- \*\*Right-justifying\*\*: `.rjust(width)`

- \*\*Left-justifying\*\*: `.ljust(width)`

- \*\*Centering\*\*: `.center(width)`

#### Example:

```python

string = "Hello"

print(string.rjust(10)) # ' Hello'

print(string.ljust(10)) # 'Hello '

print(string.center(10)) # ' Hello '

```

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

The best way to remove whitespace characters from the start or end of a string is to use the `.strip()` method. To remove whitespace from just the start or just the end, use `.lstrip()` and `.rstrip()`, respectively.

#### Example:

```python

string = " Hello, world! "

print(string.strip()) # 'Hello, world!'

print(string.lstrip()) # 'Hello, world! '

print(string.rstrip()) # ' Hello, world!'

```