Assignment-6

1. Escape Characters and Usage:

Escape characters are used within strings to represent characters that would otherwise have a special meaning within the string itself. The most common escape character is the backslash (`\`). Here are some examples:

- `\n`: Newline (creates a new line)
- `\t`: Horizontal tab
- `\"`: Double quote (to include a double quote within a double-quoted string)
- `\'`: Single quote (to include a single quote within a single-quoted string)

To use an escape character, simply place a backslash before the character you want to escape.

2. Escape Characters `\n` and `\t`:

- `\n`: Inserts a newline character, causing the text to jump to the next line when displayed.
- `\t`: Inserts a horizontal tab character, moving the cursor a certain number of spaces (usually 4 or 8 depending on the environment).

3. Including Backslash Characters:

To include a literal backslash (`\`) within a string, you need to escape it using another backslash. This is because the single backslash is used for escaping other characters. So, to represent a backslash, use `\\`.

4. Single Quote in "Howl's Moving Castle"

In the string `"Howl\'s Moving Castle"` (using double quotes), the single quote is not a problem because it's within double quotes and doesn't conflict with the string delimiter. The backslash (`\`) escapes the single quote, making it a literal character within the string.

5. Newlines Without `\n`:

Here are a few ways to create newlines without using `\n`:

- Use triple quotes (either `'''` or `"""`): These quotes allow you to write multi-line strings without needing escape characters for newlines.
- Concatenate strings: You can combine multiple string literals using the `+` operator, each on a separate line.

6. String Slicing Examples:

- `'Hello, world!'[1]`: This extracts the character at index 1, which is "e" (zero-based indexing).
- `'Hello, world!'[0:5]`: This extracts a substring from index 0 (inclusive) to index 5 (exclusive), resulting in "Hello".
- `'Hello, world!'[:5]`: This is equivalent to `[0:5]`, also resulting in "Hello".
- `'Hello, world!'[3:]`: This extracts everything from index 3 (inclusive) to the end, resulting in "lo, world!".

7. String Methods:

- `'Hello'.upper()`: Converts the string to uppercase ("HELLO").
- `'Hello'.upper().isupper()`: Checks if the string is all uppercase (True).
- `'Hello'.upper().lower()`: Converts the string to lowercase ("hello").

8. String Splitting and Joining:

- `'Remember, remember, the fifth of July.'.split()`: This splits the string on whitespace (spaces), resulting in a list of words: `['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']`
- `'-'.join('There can only one.'.split())`: This splits the string on spaces, then joins the words using a hyphen (-) as the delimiter, resulting in "There-can-only-one."

9. String Justification:

- **Right-justify:** Use `rjust(width, fillchar)`. Example: `'hello'.rjust(10, '*')` gives "*****hello".
- **Left-justify:** Use `ljust(width, fillchar)`. Example: `'hello'.ljust(10, '*')` gives "hello*****".
- **Center:** Use `center(width, fillchar)`. Example: `'hello'.center(10, '*')` gives "**hello***". (These methods might vary slightly depending on the programming language.)
-

10. Removing Whitespace:

- Use `strip()` to remove leading and trailing whitespace (spaces, tabs, newlines). Example: `' hello world '.strip()` gives "hello world".
- Use `lstrip()` to remove leading whitespace. Example: `' hello world '.lstrip()` gives "hello world ".
- Use `rstrip()` to remove trailing whitespace. Example: `' hello world '.rstrip()` gives " hello world". (These methods might vary slightly depending on the programming language.)