

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

Answer: - Escape characters are special characters used to represent certain characters that are difficult to type or include in a string directly. They are denoted by a backslash (\) followed by a specific character or sequence. For example, to include a newline character, you can use \n.

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

Answer: - The escape character \n stands for a newline, and \t stands for a tab.

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

Answer: - To include a backslash character in a string, you can use a double backslash (\\). For example, "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?

Answer: - The string "Howl's Moving Castle" is a correct value because the single quote character (') is not an issue within a string that is enclosed in double quotes ("). This is because single quotes can be included within a string that is enclosed in double quotes without the need for escaping.

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

Answer: - If you don't want to use the \n character to represent newlines, you can use triple quotes (""") to write a string spanning multiple lines. For example:

```
my_string = """Line 1
```

```
Line 2
```

```
Line 3"""
```

6. What are the values of the given expressions?

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

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

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

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

Answer: - The values of the given expressions are as follows:

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

- `'Hello, world!'[0:5]` returns the substring from index 0 to index 4 (excluding index 5), which is `'Hello'`.
- `'Hello, world![:5]` is equivalent to `'Hello, world!'[0:5]` and returns the substring from index 0 to index 4 (excluding index 5), which is `'Hello'`.
- `'Hello, world!'[3:]` returns the substring from index 3 to the end of the string, which is `'lo, world!'`.

7. What are the values of the following expressions?

`'Hello'.upper()`

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

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

Answer: - The values of the following expressions are as follows:

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

8. What are the values of the following expressions?

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

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

Answer: - The values of the following expressions are as follows:

- `'Remember, remember, the fifth of July.'.split()` splits the string into a list of words: `['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']`.
- `'-'.join('There can only one.'.split())` splits the string into words and then joins them with a hyphen (`'-'`): `'There-can-only-one.'`.

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

Answer: - The methods for right-justifying, left-justifying, and centering a string are as follows:

- Right-justifying: Use the `'str.rjust(width)'` method to right-justify a string by padding it with spaces on the left.
- Left-justifying: Use the `'str.ljust(width)'` method to left-justify a string by padding it with spaces on the right.
- Centering: Use the `'str.center(width)'` method to center a string by padding it with spaces on both sides.

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

Answer: - The best way to remove whitespace characters from the start or end of a string is by using the `str.strip()` method. It removes any leading or trailing whitespace characters from the string. For example, `" Hello, world! ".strip()` returns `"Hello, world!"`.