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

ANS :- Escape characters are special characters in Python that are used to represent certain non-printable or special characters within strings. They are denoted by a backslash (\) followed by a specific character or sequence. Escape characters allow you to include characters in strings that would otherwise be difficult or impossible to represent directly. They are used to perform various tasks such as inserting newline characters, adding tabs, including double quotes within a string, and representing special characters like backslashes. Here are some commonly used escape characters in Python:

\n: Represents a newline character and is used to insert a new line.

\t: Represents a tab character and is used to insert horizontal tab spacing.

\\: Represents a backslash character and is used to insert a literal backslash.

\': Represents a single quote character and is used to include a single quote within a single-quote delimited string.

\": Represents a double quote character and is used to include a double quote within a double quote delimited string

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

ANS :- In Python, the escape character \n represents a newline character, and the escape character \t represents a tab character.

\n: The newline character is used to create a new line in the output or within a string. When encountered in a string, it moves the cursor to the beginning of the next line.

\t: The tab character is used to insert horizontal tab spacing in the output or within a string. It creates a tab indentation, which typically corresponds to a certain number of spaces.

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

ANS :- To include a backslash character (\) in a string, we can use the escape character itself to escape the backslash. This means we need to use a double backslash (\\) to represent a single backslash in the string. Here's an example:

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?

ANS :- I n Python, the single quote character (') can be used within a string without being escaped as long as the string itself is enclosed in double quotes ("). Similarly, we can use double quotes within a string enclosed in single quotes without escaping them

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

ANS :- If we don't want to use the newline character (\n) directly in a string, we can use triple quotes (""" or ''') to create a multiline string. This allows us to include newlines without explicitly using the escape character.

6. What are the values of the given expressions?

'Hello, world!'[1]

'Hello, world!'[0:5]

'Hello, world!'[:5]

'Hello, world!'[3:]

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

- `'Hello, world!'[1]`: This expression accesses the character at index 1 in the string `'Hello, world!'`, which is `'e'`. So the value is `'e'`.

- `'Hello, world!'[0:5]`: This expression retrieves the substring starting from index 0 up to, but not including, index 5 in the string `'Hello, world!'`. The substring is `'Hello'`. So the value is `'Hello'`.

- `'Hello, world!'[:5]`: This expression is similar to the previous one. It retrieves the substring from the beginning of the string up to, but not including, index 5. So the value is `'Hello'`.

- `'Hello, world!'[3:]`: This expression retrieves the substring starting from index 3 until the end of the string `'Hello, world!'`. The substring is `'lo, world!'`. So the value is `'lo, world!'`.

In Python, string indexing starts at 0, so accessing a specific index retrieves the character at that position. Slicing allows you to extract substrings by specifying a range of indices. The starting index is inclusive, and the ending index is exclusive.

7. What are the values of the following expressions?

'Hello'.upper()

'Hello'.upper().isupper()

'Hello'.upper().lower()

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

'Hello'.upper(): This expression converts the string 'Hello' to uppercase letters. The resulting value is 'HELLO'.

'Hello'.upper().isupper(): This expression first converts the string 'Hello' to uppercase using the upper() method, resulting in 'HELLO'. The subsequent isupper() method checks if all the characters in the string are uppercase. Since they are, the expression evaluates to True.

'Hello'.upper().lower(): This expression first converts the string 'Hello' to uppercase using the upper() method, resulting in 'HELLO'. The subsequent lower() method converts all the characters to lowercase. The resulting value is 'hello'8. What are the values of the following expressions?

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

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

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

'Remember, remember, the fifth of July.'.split(): This expression splits the string 'Remember, remember, the fifth of July.' into a list of substrings using whitespace as the separator. The resulting list is ['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.'].

'-'.join('There can only one.'.split()): This expression splits the string 'There can only one.' into a list of substrings using whitespace as the separator (['There', 'can', 'only', 'one.']). Then, it joins the substrings together using a hyphen ('-') as the separator. The resulting string is 'There-can-only-one.'.

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

ANS :- In Python, the following methods can be used to adjust the alignment of a string:

1. str.ljust(width[, fillchar]): This method left-justifies a string within a specified width. It pads the string on the right with the specified fillchar (default is space) to reach the given width.

2. str.rjust(width[, fillchar]): This method right-justifies a string within a specified width. It pads the string on the left with the specified fillchar (default is space) to reach the given width.

3. str.center(width[, fillchar]): This method centers a string within a specified width. It pads the string on both sides with the specified fillchar (default is space) to reach the given width, with equal padding on each side

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

ANS :- In Python, the str.strip() method is commonly used to remove whitespace characters from the start and end of a string. It returns a new string with leading and trailing whitespace removed.