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

**Ans.** Escape characters are special characters in a string that are preceded by a backslash () to indicate that the character should be interpreted literally, rather than as a special character. They are used to represent characters that are otherwise not easily typed into a string, such as newlines, quotes, or certain symbols.

For example, to include a double quote in a string, you can use an escape character:

print(“This is a \” quote\””)

OUTPUT:

This is a “quote”

Another example is to represent a newline in a string:

print(“Line 1\line 2’)

OUTPUT:

Line 1

Line 2

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

**Ans.** The escape characters "n" and "t" stand for "new line" and "tab", respectively.

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

**Ans.** In order to include a backslash character in a string, you need to escape it by adding another backslash before it. For example, to include a single backslash in a string, you would write "\".

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.** The string "Howl's Moving Castle" is a correct value because the single quote character within the word "Howl's" is not interpreted as a string delimiter in the programming language being used. In many programming languages, such as Python and JavaScript, string values can be defined using either single quotes or double quotes, as long as the delimiter used to start the string is used to end the string. So, in this case, the single quote within the string is treated as a character in the string, rather than as a delimiter, and does not cause any problems.

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

**Ans.** You can write a string of newlines by using escape characters "\n". This allows you to create a new line in your string without actually typing the "Enter" key. For example:

string\_with\_newlines = "Line 1\nLine 2\nLine 3"

This will create a string with three separate lines, even though all of the text is contained within a single string variable.

6. What are the values of the given expressions?

‘Hello, world!’[1]

‘Hello, world!’[0:5]

‘Hello, world!’[:5]

‘Hello, world!’[3:]

**Ans.** The value of the following expressions is:

'Hello, world!'[1] = 'e'

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

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

'Hello, world!'[3:] = 'lo, world!'

7. What are the values of the following expressions?

‘Hello’.upper()

‘Hello’.upper().isupper()

‘Hello’.upper().lower()

**Ans.** The value of the following expressions is:

‘Hello’.upper() = ‘HELLO’

‘Hello’.upper().isupper() =True

‘Hello’.upper().lower() = ‘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 following expressions are:

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

'-'.join('There can only one.'.split()) returns ‘There-can-only-one’.

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

**Ans.** Right-justifying, left-justifying, and centering a string can be achieved in several ways:

**Using string formatting:** By using the "format()" method and specifying the width of the field with "{:>X}" for right-justifying, "{:<X}" for left-justifying, and "{:^X}" for centering, where X is the width of the field.

**Using libraries:** There are various libraries available such as the "textwrap" module in Python, which provides functions for formatting text, including right-justifying, left-justifying, and centering a string.

**Manually:** By adding spaces to the beginning or end of a string to make it right-justified, left-justified, or centered, as needed.

In all cases, the string is padded with spaces to achieve the desired justification.

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

**Ans.** The best way to remove whitespace characters from the start or end of a string in English is to use the `strip()` method. The `strip()` method removes any leading and trailing whitespaces from a string. For example:

string = " Hello, World! "

string = string.strip()

print(string)

Output: "Hello, World!"