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

Here are some commonly used escape characters in Python:

\': Represents a single quote character (').

\": Represents a double quote character (").

\\: Represents a backslash character (\).

\n: Represents a newline character, creating a new line in the string.

\t: Represents a tab character, creating horizontal tab spacing.

\r: Represents a carriage return character, which moves the cursor to the beginning of the current line.

\b: Represents a backspace character, which moves the cursor back one character.

\uXXXX: Represents a Unicode character with the hexadecimal value XXXX, where XXXX is the Unicode code point.

Eg:

print("I'm learning Python.") # 'I'm' is enclosed in single quotes using \' escape character.

print("She said, \"Hello!\"") # Double quotes are enclosed in a string using the escape character \".

print("C:\\path\\to\\file.txt") # Represents a file path with backslashes using the escape character \\.

print("First line.\nSecond line.") # Prints two lines with a newline character \n.

print("Name:\tJohn\tAge:\t25") # Prints name and age with tab spacing using the escape character \t.

output:

I'm learning Python.

She said, "Hello!"

C:\path\to\file.txt

First line.

Second line.

Name: John Age: 25

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

\n represents new line in the string.

\t represents tab spacing.

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

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

output:

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?

In Python, you can use either single quotes (') or double quotes (") to define string literals. When using one type of quote to define a string, you can include the other type of quote within the string without the need for escaping.

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

string\_with\_newlines = '''This is a string

with multiple

lines.'''

print(string\_with\_newlines)

output:

This is a string

with multiple

lines.

6. What are the values of the given expressions?

'Hello, world!'[1] Value is e

'Hello, world!'[0:5] Value is Hello

'Hello, world!'[:5] Value is Hello

'Hello, world!'[3:] Value is lo, world!

7. What are the values of the following expressions?

'Hello'.upper() Value is HELLO

'Hello'.upper().isupper() Value is True

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

Value is There-can-only-one.

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

1. Right-justifying a string: The rjust() method is used to right-justify a string by adding padding characters to the left of the string to achieve the desired width. The syntax is string.rjust(width, fillchar). Here, width is the total width of the resulting string, and fillchar (optional) is the character used for padding. If fillchar is not provided, a space character is used as the default padding character.

Eg:

string = 'Hello'

justified\_string = string.rjust(10)

print(justified\_string) # Output: ' Hello'

1. Left-justifying a string: The ljust() method is used to left-justify a string by adding padding characters to the right of the string to achieve the desired width. The syntax is string.ljust(width, fillchar). Here, width is the total width of the resulting string, and fillchar (optional) is the character used for padding. If fillchar is not provided, a space character is used as the default padding character.

Eg:

string = 'Hello'

justified\_string = string.ljust(10)

print(justified\_string) # Output: 'Hello '

1. Centering a string: The center() method is used to center a string by adding padding characters on both sides of the string to achieve the desired width. The syntax is string.center(width, fillchar). Here, width is the total width of the resulting string, and fillchar (optional) is the character used for padding. If fillchar is not provided, a space character is used as the default padding character.

Eg:

string = 'Hello'

centered\_string = string.center(10)

print(centered\_string) # Output: ' Hello '

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

Strip() removes leading and trailing whitespaces from a string.

Eg:

string = " Hello, World! "

trimmed\_string = string.strip()

print(trimmed\_string) # Output: "Hello, World!"