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

Ans - Escape characters are special symbols that have a different meaning when preceded by a backslash \ in a string.

For example, \n means a new line, \t means a tab, and \" means a double quote.

You can use escape characters to insert characters that are illegal or hard to type in a string, such as quotes, backslashes, or non-printable characters.

Examples of how to use escape characters in Python:

To print a string with double quotes inside, use \" to escape the quotes:

print("She said, \"Hello, world!\"")

# Output: She said, "Hello, world!"

To print a string with multiple lines, use \n to escape the new line:

print("This is the first line.\nThis is the second line.")

# Output:

# This is the first line.

# This is the second line.

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

Ans - , \n means a new line, \t means a tab,

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

Ans - To print a string with a backslash inside, use \\ to escape the backslash:

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

# Output: This is a backslash: \

1. 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 - In Python, we can use either single quotes ' or double quotes " to enclose a string.

If we use single quotes, then any single quote inside the string must be escaped with a backslash \, otherwise it will end the string prematurely.

For example:

print('She said, \'Hello, world!\'')

# Output: She said, 'Hello, world!'

However, if we use double quotes, then we don’t need to escape any single quote inside the string, because they are not considered as string delimiters.

For example:

print("Howl's Moving Castle")

# Output: Howl's Moving Castle

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

Ans- We can use triple quotes (''' or """) to enclose a string that spans multiple lines.

The newlines will be preserved when the string is displayed. For example:

string = """This is a string

with multiple

newlines"""

print(string)

# Output:

# This is a string

# with multiple

# newlines

6. What are the values of the given expressions?

'Hello, world!'[1]

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

'Hello, world!'[0:5]

Hello, world!'[0:5] returns the substring from index 0 to index 4 (not including 5), which is Hello.

'Hello, world!'[:5]

'Hello, world!'[:5] returns the same substring which is Hello, since omitting the start index means starting from 0 by default.

'Hello, world!'[3:]

'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() returns a new string with all uppercase letters, which is 'HELLO'.

'Hello'.upper().isupper()

'Hello'.upper().isupper() first calls the upper() method on the string 'Hello' and returns 'HELLO', then calls the isupper() method on the resulting string and returns True, because all characters in the string are uppercase.

'Hello'.upper().lower()

'Hello'.upper().lower() first calls the upper() method on the string 'Hello' and returns 'HELLO', then calls the lower() method on the resulting string and returns 'hello', which is a new string with all lowercase letters.

8. What are the values of the following expressions?

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

'Remember, remember, the fifth of July.'.split() returns a list of substrings that are separated by whitespace characters, which is ['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.'].

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

'-' .join('There can only one.'.split()) first splits the string 'There can only one.' on whitespace characters and returns a list of substrings, which is ['There', 'can', 'only', 'one.'], then joins the elements of the list with hyphens and returns a single string, which is 'There-can-only-one.'.

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

Ans- There are three string methods in Python that can help you align a string in a given width: rjust(), ljust(), and center().

These methods respectively right-justify, left-justify, and center a string by padding it with a specified character (default is a space) until the given width is reached.

The syntax for these methods is:

<string>.rjust(width, fillchar)

<string>.ljust(width, fillchar)

<string>.center(width, fillchar)

where <string> is any valid Python string, width is an integer that specifies the length of the output string, and fillchar is an optional argument that specifies the character to use for padding. If fillchar is omitted, a space is used by default.

Here are some examples of how to use these methods in Python:

# Right-justify a string with 20 characters and use '-' as the fill character

s = "Hello"

print(s.rjust(20, '-'))

# Output: ---------------Hello

# Left-justify a string with 10 characters and use the default fill character (space)

s = "World"

print(s.ljust(10))

# Output: World

# Center a string with 15 characters and use '\*' as the fill character

s = "Python"

print(s.center(15, '\*'))

# Output: \*\*\*\*Python\*\*\*\*

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

Ans - Whitespace characters are any characters that are not visible on the screen, such as spaces, tabs, or newlines.

To remove whitespace characters from the start or end of a string in Python, you can use the strip() method.

This method returns a new string that has all the leading and trailing whitespace characters removed.

For example:

s = " Hello, world! "

print(s.strip())

# Output: Hello, world!

If you only want to remove whitespace characters from the start or the end of the string, you can use the lstrip() or rstrip() methods respectively.

These methods return a new string that has all the leading or trailing whitespace characters removed.

For example:

s = " Hello, world! "

print(s.lstrip())

# Output: Hello, world!

print(s.rstrip())

# Output: Hello, world!