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

**Ans:** these are special characters, used to represent characters that cannot be directly typed or displayed in a string or character data type.

Represented by a backslash () followed by a character or code.

* **\n** : represents a newline character
* **\t** : represents a tab character
* **\r** : represents a carriage return character
* **\b** : represents backspace character
* **‘** : represents a single quote character
* **“** : represents a double quote character

A useful tool for **representing special characters** and **formatting text** in code.

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

* **\n** : represents a newline character
* **\t** : represents a tab character

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

Ans: If we use backslash character itself, then it will work as an escape character.

We **need to use two backslashes in a row to represent a single backslash character**.

Example: print(“c:\\windows\\system32”),,,,,output: c:\windows\system32

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:** Because single quotes are used to enclose the string, then the single quote inside the string {Howl’s}, will be treated as end of the string, therefore we **enclose** the whole string **inside** **double quote to not consider the single string as end of the string**.

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

If you don't want to use the "\n" character to create a string of newlines, you can use a **multiline string** instead. In Python, a multiline string is a string that spans multiple lines and is enclosed in **triple quotes** ('''...''' or """...""").

Example:

newlines = “””This is a

String with

Multiple lines.”””

Print(newlines)

Output:

This is a

String with

Multiple lines.

6. **What are the values of the given expressions?**

**'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’

**'Hello'.upper().isupper()** : True

**'Hello'.upper().lower()** : ‘hello’

8. **What are the values of the following expressions?**

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

['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']

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

'There-can-only-one.'

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

**Ans:** Following are the methods:

* ‘**rjust()**’ : add space to the left of the string, placing the space at the right side. The syntax is:

Example:

text = ‘Hello’

Result = text.rjust(10)

Print(result)

Output:

“ Hello”

* ‘**ljust()’ :** this adds space to the right of the string, placing the string at the left most corner

Example:

text = ‘Hello’

Result = text.ljust(10)

Print(result)

Output:

“Hello “

* ‘**center()**’ : it ads space on both side, placing the string at the center.

Example:

text = ‘Hello’

Result = text.center(10)

Print(result)

Output:

“ Hello ”

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

Ans:

* ‘**strip()’**

String = " ineuron "

String.strip()

Output:

“ineuron”

* ‘**lstrip()**’

String = " ineuron "

String.lstrip()

Output:

“ineuron ”

* ‘**rstrip()**’

String = " ineuron "

String.rstrip()

Output:

“ ineuron”