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

In Python, some characters in a string representation are not printable and are interpreted differently by Python. For instance, single quotation (‘), double quotation ("), backslash (\) and so on. These characters cannot be stored directly in a string in Python. We use escape characters to make Python consider these differently interpreted characters as part of the string.

As Python encounters an escape character, it interprets the next character in the string after the escape character differently. The majority of the escape characters begin with a backslash (\)

E.g: print (‘It\'s raining’) —> It’s raining

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

The special characters n and t stands for **new line** and **tab** respectively.

The symbol ‘\n' is interpreted as ‘go to next line' (Enter Button)

The symbol ‘\t' is interpreted as ‘Tab(tab Button)

E.g:

print(“Hello\nWorld”) —>

Hello

World

print(“Hello\tWorld”) —>

Hello World

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

To view a backslash (\) using print(), it must be preceded by another (\) in order for Python to interpret it correctly.

E.g:

print(“Hello\\World") —>

Hello\World

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?

The enclosure in the string above is done with double quotation marks, (“). As a result, the apostrophe (‘) is correctly interpreted.

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

For a multi-line description, triple quotes (‘") at the beginning and end of the string may be used instead of n. A simple enter command when defining the string is interpreted as the next line while using (‘“).

print('''

My name is Harish

I've enrolled into Full Stack Data Science course

‘’’) —————>

My name is Harish

I've enrolled into Full Stack Data Science course

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()

By default, split() splits the entered string based on whitespaces and stored each split segments in a list. But notice, “Remember,” come together before the space, so it will be stored as a single element similarly for the last segment “July.” also

———> [‘Remember,’, 'remember,', 'the', 'fifth', 'of', 'July.']

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

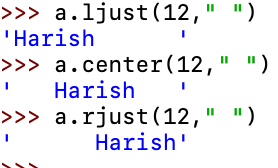
As previously stated, split() divides the string into a list of components, and the join method recombines them into a string based on the additional string supplied.

———> 'There-can-only-one.'

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

Right Justification – rjust()

Left Justification – ljust()

Centre aligned – center()

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

Strip(), a built-in Python string function, removes all whitespaces from the beginning and end of a string. It cannot, however, strip in the intermediate white spaces.

“strip()” returns a copy of the string with the whitespaces at the beginning and end deleted.

s = ' foo \t ‘

print(s.strip()) ————->

foo