Assignment-6

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

Ans: Escape characters are special characters in programming that allow you to insert characters into a string literal that could otherwise not be easily typed. These characters are preceded by a backslash () to indicate that they are escape characters and not regular characters.

Other escape characters are:

Code Result

\’ Single Quote

\\ Backslash

\n New line

\r Carriage Return

\t Tab

\b Backspace

\f Form feed

\ooo Octal value

\xhh Hex value

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

**Ans:** The escape characters n stand for New line and t stand for Tab.

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

**Ans:** To include a backslash character (**\**) in a string, you need to escape it with another backslash (**\\**). This tells the programming language that the backslash is part of the string literal and not an escape character.

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 single quote in Howl's is fine because you've used double quotes to mark the beginning and end of the string. In most programming languages, string literals can be defined using either single quotes (**'**) or double quotes (**"**). When defining a string with double quotes, any single quotes within the string do not need to be escaped. Similarly, when defining a string with single quotes, any double quotes within the string do not need to be escaped.

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

**Ans:** If you don't want to use the **\n** escape character to represent a newline in a string, you can use a combination of escape characters to create multiple lines of text. Alternatively, you can define a string literal with multiple lines using a string literal with triple quotes (either single quotes or double quotes). This allows you to include line breaks directly within the string literal.

For eg., print(“Hello”)

Print(“World”)

We see the output printed in two lines:

Hello

World

6. What are the values of the given expressions?

'Hello, world!'[1]

'Hello, world!'[0:5]

'Hello, world!'[:5]

'Hello, world!'[3:]

**Ans:** The values of the given expressions are:

* 'Hello, world!'[1] is 'e' because this expression accesses the second character in the string (indexing in programming starts at 0).
* 'Hello, world!'[0:5] is 'Hello' because this expression is slicing the string to return the characters from index 0 to 4 (the slice [0:5] returns the characters at indices 0, 1, 2, 3, and 4).
* 'Hello, world!'[:5] is 'Hello' because this expression is equivalent to 'Hello, world!'[0:5]. The default starting index is 0 if not specified.
* 'Hello, world!'[3:] is 'lo, world!' because this expression is slicing the string to return all the characters from index 3 to the end of the string. The ending index is omitted, and the slice returns all characters from the starting index to the end of the string.
* Top of Form

7. What are the values of the following expressions?

'Hello'.upper()

'Hello'.upper().isupper()

'Hello'.upper().lower()

**Ans:**

The values of the following expressions are:

* 'Hello'.upper() is 'HELLO', because this expression is calling the upper() method on the string 'Hello', which returns a new string with all characters in uppercase.
* 'Hello'.upper().isupper() is True, because this expression is calling the isupper() method on the result of the previous expression, which returns True if all characters in the string are uppercase.
* 'Hello'.upper().lower() is 'hello', because this expression is calling the lower() method on the result of the previous expression, which returns a new string with all characters in lowercase.

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() is ['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.'], because this expression is calling the split() method on the string 'Remember, remember, the fifth of July.', which splits the string into a list of strings at whitespace characters (by default).
* '-'.join('There can only one.'.split()) is 'There-can-only-one.', because this expression is calling the split() method on the string 'There can only one.', which splits the string into a list of strings at whitespace characters. The join() method is then called on the string '-' with the list of strings as an argument, which concatenates the list elements with the specified separator string '-'.

Top of Form

Ans:

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

Ans: rjust( ), center( ), or ljust() are the methods for right-justifying, centering, or left-justifying.

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 Python is to use the strip() method. The strip() method removes all leading and trailing whitespaces from a string and returns the resulting string.