Assignment - 6

Q1. What are escape characters, and how do you use them?

Sol:   
Escape characters are special characters used in programming and text processing to represent certain actions or characters that cannot be directly entered or represented in a string of text. They are typically preceded by a backslash () to indicate that they have a special meaning.

Q2. What do the escape characters n and t stand for?

Sol:

\n: Represents a newline character. When this escape sequence is encountered, it inserts a line break in the text.

\t: Represents a tab character. It is used to insert a horizontal tab in the text.

Q3. What is the way to include backslash characters in a string?

To include a backslash character () in a string, you can use the escape character \ (double backslash). The backslash itself is an escape character, so by using \, you are escaping the special meaning of the backslash and including it as a literal character in the string.

Here's an example:

Code:

my\_string = "This is a backslash: \\"

print(my\_string)

Output:

This is a backslash: \

Q4. The string &quot;Howl&#39;s Moving Castle&quot; is a correct value. Why isn&#39;t the single quote character in the

word Howl&#39;s not escaped a problem?

Sol:

In the string "Howl's Moving Castle," the single quote character (') in the word "Howl's" does not need to be escaped because the string itself is enclosed in double quotation marks ("). In many programming languages, including Python, single quotes and double quotes can be used interchangeably to define a string.

When a string is enclosed in double quotes, single quotes within the string do not need to be escaped, and vice versa. This allows for more flexibility in representing strings that contain quotation marks.

Q5. How do you write a string of newlines if you don&#39;t want to use the n character?

Sol:

Code:

my\_string = " This is line 1.\

This is line 2.\

This is line 3. "

print(my\_string)

Output:

This is line 1.

This is line 2.

This is line 3.

Q6. What are the values of the given expressions?

1. ‘Hello, world!’[1]
2. ‘Hello, world!’[0:5]
3. ‘Hello, world!’[:5]
4. ‘Hello, world!’[3:]

Sol:

1. ‘e’
2. ‘Hello,’
3. ‘Hello,’
4. ‘lo, world’

Q7. What are the values of the following expressions?

‘Hello’.upper()

‘Hello’.upper().isupper()

‘Hello’.upper().lower()

Sol:

* 'Hello'.upper() evaluates to 'HELLO'.
* 'Hello'.upper().isupper() evaluates to True.
* 'Hello'.upper().lower() evaluates to 'hello'.

Q8. What are the values of the following expressions?

1. ‘Remember, remember, the fifth of July.’.split()
2. ‘-‘.join(‘There can only one.’.split())

Sol:

1. [‘Remember’, ‘remember’, ‘the’, ‘fifth’, ‘of’, ‘July’]
2. ‘There-can-only-one.’

Q9. What are the methods for right-justifying, left-justifying, and centering a string?

Sol:

Right-justifying a string: The str.rjust(width, fillchar**)** method is used to right-justify a string within a specified width. It returns a new string that is right-justified by adding spaces (or a specified fill character) to the left of the original string until it reaches the specified width.

Example:

text = "Hello"

justified\_text = text.rjust(10)

print(justified\_text)

Output:

Hello

Left-justifying a string: The str.ljust(width, fillchar**)** method is used to left-justify a string within a specified width. It returns a new string that is left-justified by adding spaces (or a specified fill character) to the right of the original string until it reaches the specified width.

Example:

text = "Hello"

justified\_text = text.ljust(10)

print(justified\_text)

Output:

Hello

Centering a string: The str.center(width, fillchar**)** method is used to center a string within a specified width. It returns a new string that is centered by adding spaces (or a specified fill character) equally to the left and right of the original string until it reaches the specified width.

Example:

text = "Hello"

centered\_text = text.center(10)

print(centered\_text)

Output:

Hello

Q10. What is the best way to remove whitespace characters from the start or end?

Sol: str.strip(): This method removes leading and trailing whitespace characters (spaces, tabs, newlines) from a string and returns the modified string.

Code:

text = " Hello "

stripped\_text = text.strip()

print(stripped\_text)

Output:

Hello