Python Assignment - 6

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

=> Escape characters are special characters that are used to represent certain non-printable or reserved characters in strings. They are represented by a backslash (‘\’) followed by a specific character or sequence of characters. Escape characters allow you to include characters in strings that would otherwise be difficult to represent directly, such as newline characters or double quotes within a string.

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

=> ‘\n’ 🡪 New line(line break)

‘\t’ 🡪 Tab

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

=> To include a backslash character in a string, you need to use an escape character itself, which is another backslash (‘\’). This means you would use two backslashes ‘\\’ to represent a single backslash in the string.

Example:

My\_string = “This is backslash : \\”

print(My\_string)

output : This is backslash : \

**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 single quote character in the string "Howl's Moving Castle" is not causing a problem because the string is enclosed in double quotes (‘ " ‘), while the single quote is used as an apostrophe within the string. In Python, you can use either single quotes or double quotes to define a string.

When a string is enclosed in double quotes, single quotes within the string are treated as literal characters and do not need to be escaped. Similarly, when a string is enclosed in single quotes, double quotes within the string can be used without escaping them.

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

=>

1. **Multi-line string**: You can use triple quotes (‘ ''' ‘ or ‘ """ ‘) to create a multi-line string, where newlines are represented by actual line breaks in the code.
2. **Concatenation**: You can concatenate multiple string literals, each representing a line, using the + operator.

**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?**

=>

1. **Right-justifying**: The ‘str.rjust()’ method is used to right-justify a string by adding spaces on the left side of the string to reach the specified width.

Syntax:

string.rjust(width, fillchar)

1. **Left-justifying**: The ‘str.ljust()’ method is used to left-justify a string by adding spaces on the

right side of the string to reach the specified width.

Syntax:

string.ljust(width, fillchar)

1. **Centering:** The ‘str.center()’ method is used to center a string by adding spaces on both sides of the string to reach the specified width.

Syntax:

string.center(width, fillchar)

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

=> **str.strip():**

This method removes leading and trailing whitespace characters from the string.it returns

new string with the whitespace character removed.

Example:

text = ‘’ Hello “

Trimmed\_text = text.strip()

print(trimmed\_text)

output: “Hello”