PYTHON ASSIGNMENT – 6

QUESTIONS ;

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

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

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

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?

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

6. What are the values of the given expressions?

'Hello, world!'[1]

'Hello, world!'[0:5]

'Hello, world!'[:5]

'Hello, world!'[3:]

7. What are the values of the following expressions?

'Hello'.upper()

'Hello'.upper().isupper()

'Hello'.upper().lower()

8. What are the values of the following expressions?

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

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

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

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

SOLUTIONS:

1. **Escape characters** are special characters that are used to represent characters that are difficult or impossible to type directly, such as newline characters, tabs, or characters with special meanings. Escape characters are preceded by a backslash (****). For example, **\n** represents a newline character.
2. In Python:
    - **\n** stands for a newline character.
    - **\t** stands for a tab character.
3. To include a backslash character in a string, you can use a double backslash (****). For example:

```
path = "C:\\Users\\Username\\Documents"
```

4. The string "Howl's Moving Castle" is a correct value because the single quote character in the word "Howl's" is not a problem because the entire string is enclosed in double quotes. If the string were enclosed in single quotes, then the presence of a single quote within the string would require escaping.

5. To write a string of newlines without using the **\n** character, you can use triple-quotes (''' or """) to create a multiline string:

```
multiline_string = '''This is a multiline string.'''
```

6. The values of the given expressions are:
   - '**Hello, world!**'**[1]**: Returns the character at index 1, which is '**e**'.
   - '**Hello, world!**'**[0:5]**: Returns the substring from index 0 to 4 (excluding 5), which is '**Hello**'.
   - '**Hello, world!**'**[:5]**: Returns the substring from the beginning to index 4 (excluding 5), which is '**Hello**'.
   - '**Hello, world!**'**[3:]**: Returns the substring from index 3 to the end, which is '**lo, world!**'.

7. The values of the following expressions are:
   - '**Hello**'**.upper()**: Returns the string in uppercase, which is '**HELLO**'.
   - '**Hello**'**.upper().isupper()**: Checks if the string in uppercase is entirely in uppercase, which is **True**.
   - '**Hello**'**.upper().lower()**: Converts the uppercase string to lowercase, which is '**hello**'.

8. The values of the following expressions are:
   - '**Remember, remember, the fifth of July.**'**.split()**: Splits the string into a list of words, which is **['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']**.
   - '**-**'**.join('There can only one.'.split())**: Splits the string into a list of words and then joins them with a hyphen, resulting in '**There-can-only-one.**'.

9.The methods for right-justifying, left-justifying, and centering a string are:

- Right-justifying: **`rjust()`**
- Left-justifying: **`ljust()`**
- Centering: **`center()`**

EXAMPLE:

```
text = 'Python'
```

```
right_justified = text.rjust(10)
left_justified = text.ljust(10)
centered = text.center(10)
```

10. The best way to remove whitespace characters from the start or end of a string is to use the **`strip()`** method. For example:

```
text = "    Hello, world!    "
stripped_text = text.strip()
```

This removes leading and trailing whitespace from the string. If you only want to remove leading whitespace, use **`lstrip()`**, and for trailing whitespace, use **`rstrip()`**.