# Assignment 6

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

**Ans. -** Escape characters are special characters used to represent characters that are difficult or impossible to type directly. Following are some examples-

\n : Newline (line break)

\t : Horizontal tab

\\ : Backslash

\' : Single quote

\" : Double quote

```
Eg. print("Hello\nWorld")      # Newline
    OUTPUT : Hello
                    World
     print("Name:\tOm")      # Tab
    OUTPUT : Name:            Om
     print("This is a backslash: \\")     # Backslash
    OUTPUT : This is a backslash: \


    # Single and double quotes
     print('It\'s a "great" day!')
    OUTPUT : It's a "great" day!
      print("He said, \"Hello!\"")
    OUTPUT : He said, "Hello!"
```

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

**Ans. -** * \n: Newline

```
Eg.  print("Hello\nWorld")
```

OUTPUT : Hello

World

* \t: Horizontal Tab

Eg. print("Name:\tOm")

OUTPUT : Name: Om

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

**Ans. -** str = “ Hello \\ World”

print(str) // OUPTUT : 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?

**Ans.-** In the string "Howl's Moving Castle", the single quote inside the string is not a problem

because the string is enclosed in double quotes.

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

**Ans. -** We can use triple quotes

```
str = """Hello
World"""
print(str)
```

6. What are the values of the given expressions?

'Hello, world!'[1]

'Hello, world!'[0:5]

'Hello, world!'[:5]

'Hello, world!'[3:]

**Ans. -** 'Hello, world!'[1] : This will accesses the character at index 1 of the string. i.e ‘e’

'Hello, world!'[0:5] : This will slices the string from index 0 up to but not including index 5.

output will be ‘Hello’.

'Hello, world!'[:5] : This will slices the string from the start up to but not including index 5.

output will be ‘Hello’.

'Hello, world!'[3:] : This will slices the string from index 3 to the end.

output will be 'lo, world!'

7. What are the values of the following expressions?

'Hello'.upper()

'Hello'.upper().isupper()

'Hello'.upper().lower()

**Ans. -** * 'Hello'.upper() : This will converts all characters in the string to uppercase.

OUTPUT : 'HELLO'

* 'Hello'.upper().isupper() : First, 'Hello'.upper() converts the string to uppercase, resulting in 'HELLO'.

Then, .isupper() checks if the resulting string is in uppercase.

OUTPUT : True

* 'Hello'.upper().lower() : First, 'Hello'.upper() converts the string to uppercase, resulting in 'HELLO'.

Then, .lower() converts the resulting string to lowercase.

OUTPUT : ‘hello’

8. What are the values of the following expressions?

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

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

**Ans. - i)** 'Remember, remember, the fifth of July.'.split() : This expression splits the string into a list of words using whitespace as the delimiter.

Value: ['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.']

**ii)** '-'.join('There can only one.'.split()) : First, 'There can only one.'.split() splits the string into a list of words: ['There', 'can', 'only', 'one.'].

list of words: ['There', 'can', 'only', 'one.'].

Then, '-'.join(...) joins these words using '-' as the separator.

Value: 'There-can-only-one.'

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

**Ans. -** i) rjust(width, fillchar) method is used to right-justify a string within a field of a specified width

```
Eg. text = 'Hello'
     right_justified = text.rjust(10, '-')
   print(right_justified)  # Output: -----Hello
```

ii) ljust(width, fillchar) method is used to left-justify a string within a field of a specified width.

```
Eg. text = 'Hello'
              left_justified = text.ljust(10, '-')
              print(left_justified)  # Output: Hello-----
```

iii) center(width, fillchar) method is used to center a string within a field of a specified width.

```
Eg. text = 'Hello'
              centered = text.center(10, '-')
              print(centered)  # Output: --Hello---
```

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 is to use the strip() method. This method removes any leading and trailing whitespace characters, including spaces, tabs, and newlines.

```
Eg. text = '   Hello, world!   '
```

```
stripped_text = text.strip()

print(f"'{stripped_text}'")  # Output: 'Hello, world!'
```