Assignment 1
(Riya Gupta 21052522)

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

Ans: Escape characters are special characters in a string that are preceded by a backslash \. They are used to represent characters that are impossible or difficult to type directly, such as tab (\t), newline (\n), or to include characters with special meaning in strings like double quotes (\") or single quotes (\').

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

Ans: The escape characters t and n stand for:
\t: Represents a tab character
\n: Represents a newline character.

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

Ans: To include a backslash character in a string, we need to use a double backslash (\\). For example, "It is a 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?

Ans: In the string "Howl's Moving Castle," the single quote character in the word "Howl's" is not a problem because the string is enclosed in double quotes. If the string were enclosed in single quotes, we would need to escape the single quote within the string using a backslash: 'Howl\'s Moving Castle'.

5.How do you write a string of newlines if you don't want to use the n character?
Ans: To write a string of newlines without using the \n character, we can use triple-quoted strings.

6. What are the values of the given expressions?
'Hello, world!'[1]: Returns the character at index 1, that is 'e'.
'Hello, world!'[0:5]: Returns the substring from index 0 to 4, which is 'Hello'.

'Hello, world!'[:5]: Returns the substring from the beginning to index 4, which is 'Hello'.
'Hello, world!'[3:]: Returns the substring from index 3 to the end, which is 'lo, world!'.

7. What are the values of the following expressions?

'Hello'.upper(): Returns 'HELLO'.
'Hello'.upper().isupper(): Returns True because the entire string is in uppercase.
'Hello'.upper().lower(): Returns 'hello' after converting to lowercase.

8. What are the values of the following expressions?

'Remember, remember, the fifth of July.'.split(): Splits the string into a list of words.
'-'.join('There can only one.'.split()): Joins the words with a hyphen ('-').

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

Ans: Right-justifying: str.rjust(width)
Left-justifying: str.ljust(width)
Centering: str.center(width)

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.

Code for Fun:

1. Given a string, the task is to reverse the order of the words in the given string.

Ex- Input- “reverse the order of the words” Output- “words the of order the reverse”

In [1]:
```
def reverse_words(input_string):
    return ' '.join(input_string.split()[::-1])
input_str = "reverse the order of the words"
output_str = reverse_words(input_str)
print(output_str)
```

```
words the of order the reverse
```

2. Write a program to check whether string is rotated by two places

(both anti-clockwise and clockwise). Ex- Str1= “Abhiraj” Str2=”jaAbhir” Output→ Yes

In [3]:
```
def is_rotated_by_two_places(str1, str2):
    return str1[-2:] + str1[:-2] == str2 or str1[2:] + str1[:2] == str2
str1 = "Abhiraj"
str2 = "jaAbhir"
result = is_rotated_by_two_places(str1, str2)
print("Output:", "Yes" if result else "No")
```

```
Output: No
```

3. Write a program to get a single string from two given strings, separated by a space and swap the first two characters of each string.

Sample String :'Abhi', 'Raj' Expected Result : 'Raj Abhi'

In [4]:
```
def swap_and_concat(str1, str2):
    return f"{str2[:2]}{str1[2:]} {str1[:2]}{str2[2:]}"
string1 = 'Abhi'
string2 = 'Raj'
result_str = swap_and_concat(string1, string2)
print("Expected Result:", result_str)
```

```
Expected Result: Rahi Abj
```

4. Write a Python program to remove the nth index character from a nonempty string.

In [5]:
```
def remove_nth_char(input_str, n):
    return input_str[:n] + input_str[n+1:]
original_str = "example"
n_index = 2
modified_str = remove_nth_char(original_str, n_index)
print("Modified String:", modified_str)
```

```
Modified String: exmple
```

5. Write a Python program to sort a string lexicographically.

In [6]:
```
def lexicographic_sort(input_str):
    return ''.join(sorted(input_str, key=lambda x: ord(x)))
unsorted_str = "python"
sorted_str = lexicographic_sort(unsorted_str)
print("Lexicographically Sorted String:", sorted_str)
```

```
Lexicographically Sorted String: hnopty
```

In [ ]:
```

```