13: WAP to count the numbers of characters in the string and store them in a dictionary data structure.

In [28]:
```
my_string = "This is a string"
string_length = len(my_string)
my_dict = {"charcter_num": string_length}
print(my_dict)
```

```
{'charcter_num': 16}
```

- Start
- Make a string and declare it's value
- Save the length of the string in a variable using len function
- Make a dictionary and then make a key and provide the string length variable as the value.
- Print the result.
- Stop

14: WAP in python to demonstrate use of slicing in string

In [29]:
```
my_string = "Thisisastring"
slice1 = my_string[2::]
slice2 = my_string[:3:]
slice3 = my_string[::4]
print("{}\n{}\n{}".format(slice1, slice2, slice3))
```

```
isisastring
Thi
Titg
```

- Start
- Make a string and declare it's value
- Make use of square brackets and colon to slice the string, experiment with the start, stop and step values.
- Print the result
- Stop

15: WAP a python program to demonstrate concatenation of a string.

In [30]:
```
my_string1 = "This is "
my_string2 = "a string"
new_string = my_string1 + my_string2
print(new_string)
```

```
This is a string
```

- Start
- Make two diffrent strings and declare their values.

- Now using the addition operator (+) join the two strings together and store that into a new variable
- Print the result
- Stop

16: WAP in python to demonstrate replication of strings

```
In [31]: my_string = "A String\t"
         replicated_string = my_string * 3
         print(replicated_string)
```

```
A String        A String        A String
```

- Start
- Make a string and declare it's values
- Now using the multiplication operator (*) replication the string to a desired amount and store that into a new variable
- Print the result
- Stop

17: WAP to calculator the greatest of three numbers

```
In [32]: first_num = 3
         second_num = 6
         third_num = 1
         numbers = [first_num, second_num, third_num]
         print(max(numbers))
```

```
6
```

- Start
- Make three varibles and declare them with three integers
- Now, make a list and put those varibles in the list either by using append method or manually
- After that use the max function on the list to get the result
- Print the result
- Stop

18: WAP to initialize a 1D array

```
In [33]: my_array = [23, 21, 44]
         print(my_array)
```

```
[23, 21, 44]
```

- Start
- Make a list and declare the values in it, make sure to have same types of values to make it a array.
- Print the result

- Stop

19: WAP to sort 1D array

```
In [34]: my_array = [5, 2, 6, 7]
         print(sorted(my_array))
```

```
[2, 5, 6, 7]
```

- Start
- Make a list and then declare values with same type to make it an array.
- Using the sorted method, sort the array
- Print the result
- Stop

20: WAP to initialize a 2D array

```
In [35]: my_array = [[1, 5, 2], [3, 2, 2]]
         print(my_array)
```

```
[[1, 5, 2], [3, 2, 2]]
```

- Start
- Make a list and as values, provide more lists to make it a 2D array, also make sure to have the same types of values to make a valid array.
- Print the result
- Stop

21: WAP that defines and prints a matrix

```
In [42]: matrix = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
         for row in matrix:
             for column in row:
                 print(column, end="")
             print()
```

```
123
456
789
```

- Start
- Make a 2D array using list and declare the values as desired
- Run the matrix in a nested loop to print the result
- Stop

22: WAP to define a function using the def keyword

```
In [39]: def foo():
             print("Greetings sir!")
```

```
foo()
```

```
Greetings sir!
```

- Start
- Declare a function using the def keyword
- Define the body of the function
- Print the result
- Stop

23: WAP to define a parameterized function to swap two numbers

In [40]:
```
def swap(a, b):
    temp = a
    a = b
    b = temp
    print("Value of A: {} and B: {}".format(a, b))

swap(a=45, b=22)
```

```
Value of A: 22 and B: 45
```

- Start
- Declare a function and add parameters as two numbers
- Now using a temporary variable, swap those two numbers
- Print the result by calling the function
- Stop

24: WAP to calculate the factorial of a number using recursion

In [43]:
```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)
fact = factorial(5)
print(fact)
```

```
120
```

- Start
- Declare a function with a parameter
- Using if condition setup a base condition to avoid infinte looping
- Call the function inside the definition of the function to ensure recursion
- Print the result, hence calling the function
- Stop