➢ operations on Strings

Concatenation (+)
It combines two strings into one.

```
# example

var1 = 'Python'
var2 = 'String'
print (var1+var2)
# Python String
```

Repetition (*)
This operator creates a new string by repeating it a given number of times.

```
# example

var1 = 'Python'
print (var1*3)
# PythonPythonPython
```

Slicing [ ]
The slice operator prints the character at a given index.

```
# example
var1 = 'Python'
print (var1[2])
# t
```

Range Slicing [x:y]
It prints the characters present in the given range.

```
# example
var1 = 'Python'
```

```
print (var1[2:5])
# tho
```

Membership (in)

This operator returns ‘True’ value if the character is present in the given String.

```
# example
var1 = 'Python'
print ('n' in var1)
# True
```

Membership (not in)

It returns ‘True’ value if the character is not present in the given String.

```
# example
var1 = 'Python'
print ('N' not in var1)
# True
```

Ex.

```
>>> "come" in s1
True
>>> "come" not in s1
False
>>>
```

Iterating (for)

With this operator, we can iterate through all the characters of a string.

```
# example
for var in var1:
```

```
print (var, end ="")
# Python
```

Raw String (r/R)

We can use it to ignore the actual meaning of Escape characters inside a string. For this, we add ‘r’ or ‘R’ in front of the String.

```
# example
print (r'\n')
# \n
print (R'\n')
# \n
```

Conversion Functions

1. capitalize() – Returns the string with the first character capitalized and rest of the characters in lower case.

```
var = 'PYTHON'
print (var.capitalize())
# Python
```

2. lower() – Converts all the characters of the String to lowercase

```
var = 'TechBeamers'
print (var.lower())
# techbeamers
```

3. upper() – Converts all the characters of the String to uppercase

```
var = 'TechBeamers'
print (var.upper())
# TECHBEAMERS
```

4. swapcase() – Swaps the case of every character in the String means that lowercase characters got converted to uppercase and vice-versa.

```
var = 'TechBeamers'
print (var.swapcase())
# tECHbEAMERS
```

5. title() – Returns the ‘titlecased’ version of String, which means that all words start with uppercase and the rest of the characters in words are in lowercase.

```
var = 'welcome to Python programming'
print (var.title())
# Welcome To Python Programming
```

6. count(str[, beg [, end]]) – Returns the number of times substring ‘str’ occurs in the range [beg, end] if beg and end index are given else the search continues in full String Search is case-sensitive.

```
var='TechBeamers'
str='e'
print (var.count(str))
# 3
var1='Eagle Eyes'
print (var1.count('e'))
# 2
var2='Eagle Eyes'
print (var2.count('E',0,5))
# 1
```

➢ String subscript

The subscript creates a slice by including a colon within the braces:

string[subscript]

The subscript creates a slice by including a colon within the braces:

string[begin:end:step]

It works by doing [begin:end:step] - by leaving begin and end off and specifying a step of -1, it reverses a string

```
str = 'Python String'
print(str[::-1])
```

O/p

gnirtS nohtyP

➢ String Methods

Python has several built-in methods associated with the string data type. These methods let us easily modify and manipulate strings. Built-in methods are those that are defined in the Python programming language and are readily available for us to use. Here are some of the most common string methods.

➢ String count() :-

This method returns the number of occurrences of a substring in the given string.

ex. str = "Python is Object Oriented"
substr = "Object"
print(str.count(substr)) # return 1, becasue the word Object exist 1 time in str

output : 1

➢ **String len() method**

String len() method return the length of the string.

```
str = "Hellow World!"
print(len(str))
```
Output : 13

- The Slice Operator

  You can take subset of string from original string by using [] operator also known as slicing operator.
  Syntax: s[start: end]

  This will return part of the string starting from index start to index end - 1.
  Let's take some examples.

```
>>> s =
"Welcome"
>>> s[1:3]
el
```

ex. >>> s = "Welcome"
>>>
>>> s[:6]
'Welcom'
>>>
>>> s[4:]
'ome'
>>>
>>> s[1:-1]
'elcom'

Ex.
s = "Welcome"

```
print(s[1:3])
print(s[:6])
print(s[4:])
print(s[1:-1])
```

- Index Operator: Working with the Characters of a String,

  String index() method returns the index of a substring inside the given string.

  index(substr,start,end)

end(optional) by default its equal to the length of the string.

```
str = "Python is Object Oriented"
    substr = "is"
    print(str.index(substr))
```

Output : 7

- **startswith() method**

String startswith() method returns Boolean TRUE, if the string Starts with the specified substring otherwise, it will return False.

```
Ex. str = "Python is Object Oriented"
print(str.startswith("Python"))
print(str.startswith("Object"))
```

Ouput :

True
False

- **String endswith() method**

String endswith() method returns Boolean TRUE, if the string Ends with the specified substring otherwise, it will return False.

```
str = "Python is Object Oriented"

print(str.endswith("Oriented"))

print(str.endswith("Object"))
```

Output : True

False

- **String split() method**

String split() method break up a string into smaller strings based on a delimiter or character.

Ex.

```
str = 'Python is Object Oriented'
print(str.split())
```

output :

['Python', 'is', 'Object', 'Oriented']

Ex.

```
str = 'Python,is,Object,Oriented'

print(str.split(','))
```

output :

['Python', 'is', 'Object', 'Oriented']

- Python returned split string as a List

```
str = 'Python,is,Object,Oriented'

sList = str.split(',')
for temp in sList:
```

```
    print (temp)
```
output :

Python

is

Object

Oriented

➢ String Comparison

You can use (> , < , <= , <= , == , !=) to compare two strings. Python compares string lexicographically i.e using ASCII value of the characters.

Suppose you have str1 as "Mary" and str2 as "Mac". The first two characters from str1 and str2 (M and M) are compared. As they are equal, the second two characters are compared. Because they are also equal, the third two characters (r and c) are compared. And because r has greater ASCII value than c, str1 is greater than str2.

How to execute the comparison

String comparison in Python takes place character by character. That is, characters in the same positions are compared from both the strings.

If the characters fulfill the given comparison condition, it moves to the characters in the next position. Otherwise, it merely returns False.

Here are some more examples:

```
>>> "tim" == "tie"
False
>>> "free" != "freedom"
True
```

```
>>> "arrow" > "aron"
True
>>> "right" >= "left"
True
>>> "teeth" < "tee"
False
>>> "yellow" <= "fellow"
False
>>> "abc" > ""
True
>>>
```