Chapter 3: Python Strings and Operators

3.1 Python Strings:

- Strings in python are surrounded by either single quotation marks, or double quotation marks. *Example:* 'hello' is the same as "hello".
- To display a string value use the print() function.
- Example:

print("Hello")
print('Hello')
	Output:
	Hello
	Hello

Assign String to a Variable: Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example:

a = "Hello"
print(a)
	Output:
	Hello

Multiline Strings: To assign a multiline string to a variable by using three quotes:

Example: [Make use 3 double quotes or 3 single quotes:]

a = """Sutex Bank College of
Computer Applications and
Science """
 	
b=''' Veer Narmad South
Gujarat University '''
 	
print(a)
print(b)
	Output:
	
	Sutex Bank College of
	Computer Applications and
	Science
	
	Veer Narmad South
	Gujarat University

Strings are Character Arrays

- Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
- However, Python does not have a character data type, a single character is simply a string with a length of 1.
- Square brackets can be used to access elements of the string.

Example:
[Get the character at position 1 (remember that the first character has the position 0):]

```
a = "Hello, World!"
print(a[1])
print(a[5])
```

Output:

```
e
,
```

Looping Through a String

- Since strings are arrays, we can loop through the characters in a string, with a for loop.

Example

```
for x in "banana":
  print(x)
```

Output:

```
banana
```

Python - Slicing Strings

- To return a range of characters by using the slice syntax.
- Specify the start index and the end index, separated by a colon, to return a part of the string.

Example: [Get the characters from position 2 to position 5 (not included):]

```
b = "Hello, World!"
print(b[2:5])
```

Output:

```
llo
```

Note: The first character has index 0.

Slice from the Start:

- By leaving out the start index, the range will start at the first character:

Example: [Get the characters from the start to position 5 (not included):]

```
b = "Hello, World!"
print(b[:5])
```

Output:

```
Hello
```

Slice To the End

- **By leaving out the end index, the range will go to the end:**

Example: [Get the characters from position 2, and all the way to the end:]

```
b = "Hello, World!"
print(b[2:])
```

Output:

```
llo, World!
```

Negative Indexing

- Use negative indexes to start the slice from the end of the string:

Example
Get the characters:
From: "o" in "World!" (position -5) To, "d" in "World!" (position -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

Output:

```
orl
```

String Methods:

Sr No	**string method**	**Explanation**	**Example**	**Output**

1	**center**	The center() method will center align the string, using a specified character (space is default) as the fill character.	txt = "banana"
			x = txt.center(20)
			print(x)
				banana

2	**count**	The count() method returns the number of times a specified value appears in the string.	txt = "I love apples, apple are my favorite fruit"
			x = txt.count("apple")
			print(x)
				2

3	**join**	The join() method takes all items in an iterable and joins them into one string.
		
		A string must be specified as the separator.
			txt = ("John", "Peter", "Vicky")
			x = "#".join(txt)
			
			print(x)
				John#Peter#Vicky

4	**len**	To get the length of a string, use the len() function.	a = "Hello, World!"
			x=len(a)
			print(x)
				13

5	**max**	The max() methods is used to find the largest characters in a string.	txt = "ABCD"
			x = max(txt)
			print(x)
				D

6	**min**	The min() methods is used to find the smallest characters in a string.	txt = "ABCD"
			x = min(txt)
			print(x)
				A

7	**replace**	The replace() method replaces a specified phrase with another specified phrase.
		By Default, All occurrences of the specified phrase will be replaced
			txt = "one one was a race horse, two two was one too."
			x = txt.replace("one", "three")
			print(x)
			
			Replace the two first occurrence of the word "one":
			
			txt = "one one was a race horse, two two was one too."
			x = txt.replace("one", "three", 2)
			print(x)
				three three was a race horse, two two was three too.
				
				----------------------- 	
				three three was a race horse, two two was one too.

8	**lower**	The lower() method returns a string where all characters are lower case.
		
		Symbols and Numbers are ignored
			txt = "Hello my FRIENDS"
			x = txt.lower()
			print(x)
				hello my friends

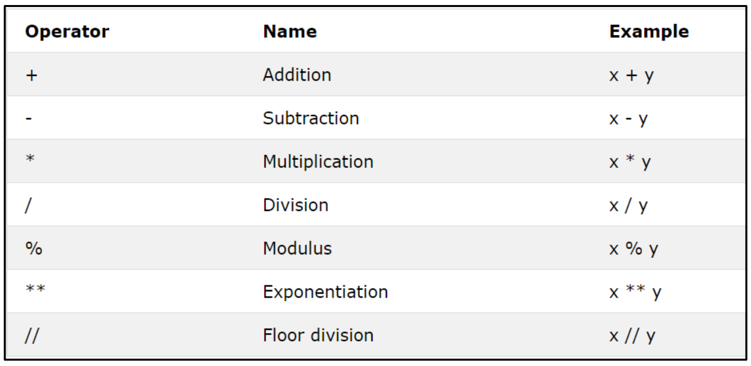
9	**upper**	The upper() method returns a string where all characters are in upper case.
		
		Symbols and Numbers are ignored.
			txt = "Hello my FRIENDS"
			x = txt.upper()
			print(x)
				HELLO MY FRIENDS

10	**split**	The split() method splits a string into a list.
		
		You can specify the separator, default separator is any whitespace.
			txt = "welcome to the jungle"
			x = txt.split()
			print(x)
				['welcome', 'to', 'the', 'jungle']

3.2 Python Operators

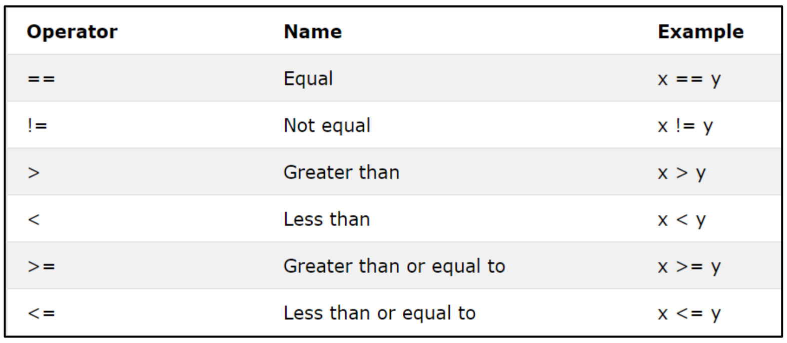
- Operators are used to perform operations on variables and values.
- Python divides the operators in the following groups:
  1 Arithmetic operators
  2 Comparison operators
  3 Assignment operators
  4 Logical operators
  5 Identity and member operators

1. **Python Arithmetic Operators**
   Arithmetic operators are used with numeric values to perform common mathematical operations:



Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

2. **Python Comparison Operators**
   Comparison operators are used to compare two values:



Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

3. Python Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

4. Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example (x=1)	Output
and	Returns True if both statements are true	x < 5 and x < 10	1(true)
or	Returns True if one of the statements is true	x == 5 or x > 4	1(true)
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)	0 (false)

5. Python Identity and Member Operators

1. Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example (x=5 , y=10)	Output
is	Returns True if both variables are the same	x is y	0(false)
is not	Returns True if both variables are not the same	x is not y	1(true)

2. Member Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example	Output
in	Returns True if a sequence with the specified value is present in the object	x=("apple","banana") print("apple" in x)	True
not in	Returns True if a sequence with the specified value is not present in the object	x=("apple","banana") print("mango" not in x)	True