Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes.

Each character is encoded in the ASCII or Unicode character. So we can say that Python strings are also called the collection of Unicode characters.

str = "Hello Creative !"

Here, if we check the type of the variable **str** using a Python script

1. **print**(type(str)),

then it will **print** a string (str).

In Python, strings are treated as the sequence of characters, which means that Python doesn't support the character data-type; instead, a single character written as 'p' is treated as the string of length 1.

```
# Using single quotes
str1 = 'creative'
print(str1)

# Using double quotes
str2 = "Hello creative"
print(str2)

# Using triple quotes
str3 = ''' Triple quotes are generally used for
    represent the multiline or
    docstring'''
print(str3)
```

Strings indexing and splitting

str = "HELLO"

H	E	L	L	O
0	1	2	3	4

Like other languages, the indexing of the Python strings starts from 0. For example, The string "HELLO" is indexed as given in the below figure.

str[0] = 'H'

str[1] = 'E'

str[2] = 'L'

str[3] = 'L'

str[4] = 'O'

Slice Operator

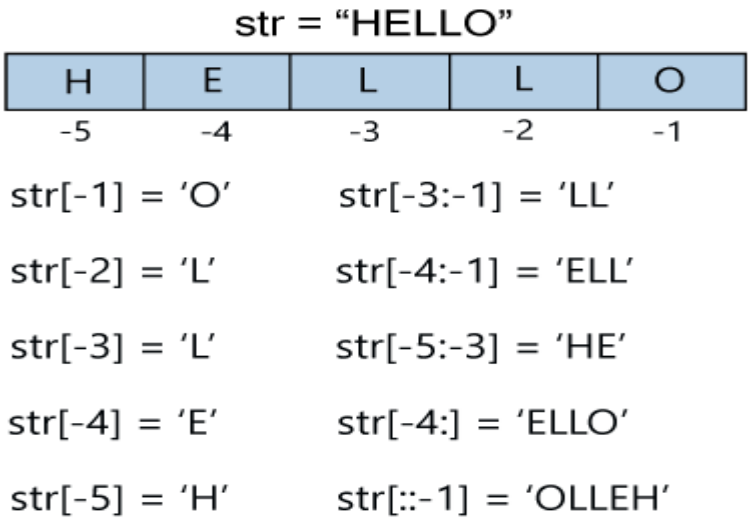
```
# Given String
str = "Creative Design"

# Start Oth index to end
print(str[0:])

# Starts 1th index to 4th index
```

```
print(str[1:5])

# Starts 2nd index to 3rd index
print(str[2:4])

# Starts 0th to 2nd index
print(str[:3])

#Starts 4th to 6th index
print(str[4:7])
```

we can do the negative slicing in the string;

it starts from the rightmost character, which is indicated as -1.

The second rightmost index indicates -2, and so on.




```
# Given String
str = "Creative Design"

print(str[-1])
print(str[-3])
print(str[-2:])
print(str[-4:-1])
print(str[-7:-2])
# Reversing the given string
print(str[::-1])
```

Reassigning Strings

Updating the content of the strings is as easy as assigning it to a new string.

The string object doesn't support item assignment i.e.,

A string can only be replaced with new string since its content cannot be partially replaced.

Strings are immutable in Python.

Str=”creative”

Str[2]=”k”

It generate Eror..

the string **str** can be assigned completely to a new content as specified

Str=”creative”

Str=”design”

It produce result design.

String Operators

Operator	Description
+	It is known as concatenation operator used to join the strings given either side of the operator.
*	It is known as repetition operator. It concatenates the multiple copies of the same string.
[]	It is known as slice operator. It is used to access the sub-strings of a particular string.
[:]	It is known as range slice operator. It is used to access the characters from the specified range.
in	It is known as membership operator. It returns if a particular sub-string is present in the specified string.
not in	It is also a membership operator and does the exact reverse of in. It returns true if a particular substring is not present in the specified string.
%	It is used to perform string formatting. It makes use of the format specifiers used in C programming like %d or %f to map their values in python

Python String functions

Method	Description
capitalize()	It capitalizes the first character of the String.
casefold()	It returns a string to lowercse in such cases
count(string,begin,end)	It counts the number of occurrences of a substring in a String between begin and end index.
endswith(suffix ,begin=0,end=len(string))	It returns a Boolean value if the string terminates with given suffix between begin and end.
find(substring ,beginIndex, endIndex)	It returns the index value of the string where substring is found between begin index and end index.
format(value)	It returns a formatted version of S, using the passed value.
index(subsring, beginIndex, endIndex)	It throws an exception if string is not found. It works same as find() method.
isalnum()	It returns true if the characters in the string are alphanumeric i.e., alphabets or numbers and there is at least 1 character. Otherwise, it returns false.
isalpha()	It returns true if all the characters are alphabets and there is at least one character, otherwise False.

isdecimal()	It returns true if all the characters of the string are decimals.
isdigit()	It returns true if all the characters are digits and there is at least one character, otherwise False.
islower()	It returns true if the characters of a string are in lower case, otherwise false.
isnumeric()	It returns true if the string contains only numeric characters.
isupper()	It returns true if all the characters of the string(if exists) is true otherwise it returns false.
len(string)	It returns the length of a string.
lower()	It converts all the characters of a string to Lower case.
replace(old,new[,count])	It replaces the old sequence of characters with the new sequence. The max characters are replaced if max is given.
split(str,num=string.count(str))	Splits the string according to the delimiter str. The string splits according to the space if the delimiter is not provided. It returns the list of substring concatenated with the delimiter.
startswith(str,beg=0,end=len(str))	It returns a Boolean value if the string starts with given str between begin and end.
upper()	It converts all the characters of a string to Upper Case.

String capitalize() Method

Python **capitalize()** method converts first character of the string into uppercase without altering the whole string. It changes the first character only and skips rest of the string unchanged.

str = "Creative"

str2 = str.capitalize()

String Casefold() Method

Python **Casefold()** method returns a lowercase copy of the string. It is more simillar to lowercase method except it removes all case distinctions present in the string.

For example in German, 'β' is equivelent to "ss". Since it is already in lowercase, lowercase do nothing and prints 'β' whereas casefold converts it to "ss".

str = "CREATIVE"

str2 = str.casefold()

output=creative

String Count() Method

It returns the number of occurences of substring in the specified range.

It takes three parameters, first is a substring, second a start index and third is last index of the range.

Start and end both are optional whereas substring is required.

count(sub[, start[, end]])

```
str = "Creative Design & Multimedia Institute"
str2 = str.count('t')
print("occurences:", str2)
```

passing second parameter (start index).

```
str = "Creative Design & Multimedia Institute"
str2 = str.count('t',6)
print("occurences:", str2)
```

Using all three parameters and returning result from the specified range.

```
str = "Creative Design & Multimedia Institute"
str2 = str.count('t',2,10)
print("occurences:", str2)
```

String endswith() Method

Python **endswith()** method returns true of the string ends with the specified substring, otherwise returns false.

endswith(suffix[, start[, end]])

```
str = "Creative Design & Multimedia Institute."
str2 = str.endswith(".")
print(str2)
```

Providing start index of the range from where method starts searching.

```
str = "Creative Design & Multimedia InstituteD"
str2 = str.endswith("D",2)
print(str2)
```

```
str = "Creative Design & Multimedia InstituteD"
str2 = str.endswith("D",2,20)
print(str2)
```

String find() Method

Python **find()** method finds substring in the whole string and returns index of the first match. It returns -1 if substring does not match.

find(sub[, start[, end]])

```
str = "Creative Design Multimedia Institute"
str2 =str.find("Design")
print(str2)
```

It returns -1 if not found any match

String format() Method

Python **format()** method is used to perform format operations on string.

While formatting string a delimiter {} (braces) is used to replace it with the value. This delimeter either can contain index or positional argument.

format(*args, **kwargs)

```
str = "Java"
str2 = "python"
print("{} and {} are programming language".format(str,str2))
```

The delimiter (braces) are using numerical index to replace and format string.

```
str = "Java"
str2 = "C#"
str3 = "{1} and {0} both are programming languages".format(str,str2)
print(str3)
```

Formatting numerical value in different-different number systems.

```
val = 10
# Calling function
print("decimal: {0:d}".format(val)); # display decimal result
print("hex: {0:x}".format(val));     # display hexadecimal result
print("octal: {0:o}".format(val));   # display octal result
print("binary: {0:b}".format(val));  # display binary result
```

Formating float and percentile in string is pretty easy.

```
val = 100000000
print("decimal: {:,}".format(val)); # formatting float value
print("decimal: {:.2%}".format(10)); # formatting percentile value
```

String index() Method

Python **index()** method is same as the find() method except it returns error on failure.

This method returns index of first occurred substring and an error if there is no match found.

index(sub[, start[, end]])

```
str = "Creative Design & Multimedia Institute"
# Calling function
str2 = str.index("tim")
# Displaying result
print(str2)
```

```
str = "Creative Design & Multimedia Institute"
# Calling function
```

```
str2 = str.index("time",2,30)
# Displaying result
print(str2)
```

String isalnum() Method

Python **isalnum()** method checks whether the all characters of the string is alphanumeric or not.

A character which is either a letter or a number is known as alphanumeric. It does not allow special chars even spaces.

isalnum()

```
str = "Creative123"
# Calling function
str2 = str.isalnum()
# Displaying result
print(str2)
```

String isalpha() Method

Python **isalpha()** method returns true if all characters in the string are alphabetic.
It returns False if the characters are not alphabetic.
It returns either True or False.

```
str = "Creative"
# Calling function
str2 = str.isalpha()
# Displaying result
print(str2)
```

```
str = "Creative"
if str.isalpha().__eq__("true"):
    print("welcome")
else:
    print("go back")
```

String isdecimal() Method

Python **isdecimal()** method checks whether all the characters in the string are decimal or not.
Decimal characters are those have base 10.
This method returns boolean either true or false.

```
str = "123"     # True
str3 = "2.50"   # False
# Calling function
str2 = str.isdecimal()
str4 = str3.isdecimal()
# Displaying result
print(str2)
print(str4)
```

String isdigit() Method

Python **isdigit()** method returns True if all the characters in the string are digits. It returns False if no character is digit in the string.

```
str = '12345'
str2 = str.isdigit()
print(str2)
```

String isidentifier() Method

Python **isidentifier()** method is used to check whether a string is a valid identifier or not.

It returns True if the string is a valid identifier otherwise returns False.

Python language has own identifier definition which is used by this method.

```
str = "creative"
str2 = str.isidentifier()
print(str2)
```

String islower() Method

Python string **islower()** method returns True if all characters in the string are in lowercase. It returns False if not in lowercase.

```
str = "creative"
str2 = str.islower()
print(str2)
```

String can have digits also and this method works in letters case and ignores digits.

```
str = "creative2345"
str2 = str.islower()
print(str2)
```

String isnumeric() Method

Python **isnumeric()** method checks whether all the characters of the string are numeric characters or not.
It returns True if all the characters are true, otherwise returns False.
Numeric characters include digit characters and all the characters which have the Unicode numeric value property.

```
str = "2345"
str2 = str.isnumeric()
print(str2)
```

String isupper() Method

Python **isupper()** method returns True if all characters in the string are in uppercase. It returns False if characters are not in uppercase.

```
str = "CREATIVE"
str2 = str.isupper()
print(str2)
```

String join() Method

Python **join()** method is used to concat a string with iterable object. It returns a new string which is the concatenation of the strings in iterable. It throws an exception TypeError if iterable contains any non-string value.

It allows various iterables like: List, Tuple, String etc.

```
str = ":"
list = ['1','2','3']
print(str.join(list))
```

String lower() Method

Python **lower()** method returns a copy of the string after converting all the characters into lowercase.

```
str = "Creative Design"
print(str.lower())
```