Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the print() function.

```
print("Hello")
print('Hello')
```

```
Hello
Hello
```

```
# String Representation
# ------
# Using single or double quote

print("Single line String")
print("------------------")
name_1 = "Kasaragod"
name_2 = 'Kasaragod'
print(name_1)
print(name_2)

print("\nMultiline String")
print("----------------")
address = '''Kasaragod,
Kerala,
India
'''
print(address)
```

```
Single line String
------------------
Kasaragod
Kasaragod

Multiline String
----------------
Kasaragod,
Kerala,
India
```

Strings as Array

You can access the elements in a string using square brackets by specifying the position.

```

```

```
name = "Kasaragod"
print(name[0])
print(name[3])
```

```
K
a
```

Loopin through a String

```
for i in "Kasaragod":
    print(i)

print("\n")
```

```
K
a
s
a
r
a
g
o
d
```

String Length

The len() function returns the length of a string.

```
name = "Kasaragod"
print("length : ", len(name))
```

```
length :  9
```

Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

```
location = "District : Kasaragod"
print("District" in location)

# Using if statement
location = "District : Kasaragod"
if "District" in location:
    print("'District' is present")
```

```
True
'District' is present
```

Check if Not

To check if a certain phrase or character is NOT present in a string, we can use the keyword not in

```
location = "District : Kasaragod"
print("District" not in location)

# Using if statement
location = "District : Kasaragod"
if "Distance" not in location:
    print("'Distance' is not present")
```

```
False
'Distance' is not present
```

String Slicing

- Return a range of characters by using the slice syntax.
- Posetive indexing are done by left to right, start from zero.
- Negative indexing are done by right to left, start from -1.

```
location = "District : Kasaragod"

print(location[2:7])
print(location[:20])
print(location[4:])

print(location[-8:-3])
```

```
stric
District : Kasaragod
rict : Kasaragod
asara
```

Modify Strings

```
# The upper() method returns the string in upper case
a = "District : Kasaragod "
print(a.upper())
```

```
DISTRICT : KASARAGOD
```

```
```

```
# The lower() method returns the string in lower case
print(a.lower())
```

```
     district : kasaragod
```

```
# The strip() method removes any whitespace from the beginning or the end
print(a.strip())
```

```
     District : Kasaragod
```

```
# The replace() method replaces a string with another string
print(a.replace("t", "J"))
print(a)
```

```
     DisJricJ : Kasaragod
     District : Kasaragod
```

```
# The split() method splits the string into substrings if it finds instances of the separ
print(a.split(":")) #default delimiter is space
```

```
     ['District ', ' Kasaragod ']
```

String Concatenation

```
a = "District :"
b = "Kasaragod"
print(a+b)
```

```
     District :Kasaragod
```

String Format

we cannot combine strings and numbers using +.

we can combine strings and numbers by using the format() method.

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are.

```
age = 36
details = "My age is {}"
print(details.format(age))

# The format() method takes unlimited number of arguments, and are placed into the respec
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

```
# You can use index numbers {0} to be sure the arguments are placed in the correct placeh
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

```
My age is 36
I want 3 pieces of item 567 for 49.95 dollars.
I want to pay 49.95 dollars for 3 pieces of item 567.
```

Escape Characters

```
\'   Single Quote
\\   Backslash
\n   New Line
\t   Tab
\b   Backspace
```

String Methods

```
capitalize()    Converts the first character to upper case
casefold()      Converts string into lower case
center()        Returns a centered string
count()         Returns the number of times a specified value occurs in a stri
find()          Searches the string for a specified value and returns the posi
index()         Searches the string for a specified value and returns the posi
isalnum()       Returns True if all characters in the string are alphanumeric
isalpha()       Returns True if all characters in the string are in the alphab
isdecimal()     Returns True if all characters in the string are decimals
isdigit()       Returns True if all characters in the string are digits
```