Iterating a String

```
s = 'Hello'

s
```

```
'Hello'
```

```
for i in s:
    print(i)
```

```
H
e
l
l
o
```

```
for i in range(len(s)):
    print(s[i])
```

```
H
e
l
l
o
```

```
for i in range(0,5,2):
    print(i)
```

```
0
2
4
```

String Concatenation

```
s1 = 'Hello '

s2 = ' World'

s = s1 , s2

s
```

```
('Hello ', ' World')
```

```
print(type(s))
```

```
<class 'tuple'>
```

```
s = s1 + s2

s
```

```
'Hello  World'
```

```
print(type(s))
```

```
<class 'str'>
```

String Repitition

```
s1 = 'Hi'
```

```
s1 = s1*5.2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[32], line 1
----> 1 s1 = s1*5.2

TypeError: can't multiply sequence by non-int of type 'float'
```

```
s1*3
```

```
'HiHiHi'
```

```
s1 #Doesn't affect the original string
```

```
'Hi'
```

```
s1[0] = 'M' # Immutable Object
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[42], line 1
----> 1 s1[0] = 'M'

TypeError: 'str' object does not support item assignment
```

String Slicing

```
s1 = 'Hello World'
```

```
s1[6::]
```

```
'World'
```

```
s1[0:len(s1):]
```

```
'Hello World'
```

```
s1[:len(s1):]
```

```
'Hello World'
```

```
s1[0::]
```

```
'Hello World'
```

```
s1[len(s1)-1:0:-1]
```

```
'dlroW olle'
```

```
s1[-1:-len(s1)-1:-1]
```

```
'dlroW olleH'
```

```
s1[::-1]
```

```
'dlroW olleH'
```

```
len(s1)
```

```
11
```

in and not in operator

```
s1 = 'Hello World'
```

```
'World' not in s1
```

```
False
```

```
'o' in s1
```

```
True
```

help()

```
help(s1.rfind)
```

```
Help on built-in function rfind:

rfind(...) method of builtins.str instance
    S.rfind(sub[, start[, end]]) -> int

    Return the highest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.
```

find(), rfind(), index(), rindex(), count()

```
s = 'hello, how are you'
```

```
s.find('o')
```

```
4
```

```
s.find('o',7)
```

```
8
```

```
s.find('z')
```

```
-1
```

```
s.index('o')
```

```
4
```

```
s.index('z')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[84], line 1
----> 1 s.index('z')

ValueError: substring not found
```

```
s.rfind('o') #from right side
```

```
16
```

```
s.find('o',5,10) #can specify the range as well
```

```
8
```

```
s.rindex('o')
```

```
16
```

```
s.count('o')
```

```
3
```

Alignment --> ljust(), rjust(), center()

```
s = 'python'
```

```
s
```

```
'python'
```

```
s.ljust(10)
'python    '
s.ljust(10,'*')
'python****'
s.rjust(10,'*')
'****python'
s.center(10,'*')
'**python**'
```

strip(), lstrip(), rstrip()

```
s = '     python      '
s = s.lstrip()
s
'python      '
s = s.rstrip()
s
'python'
s = '     python      '
s.strip()
'python'
s1 = '...   ....    +++++aaapython'
s1.lstrip()
'...   ....    +++++aaapython'
s1.lstrip('.')
'   ....    +++++aaapython'
s1.lstrip('. ')
'+++++aaapython'
s1.lstrip('. +')
```

```
'aaapython'
```

```
s1.lstrip('. +a')
```

capitalize(), upper(), lower(), title(), swapcase()

```
s = 'hello students'
s.capitalize()
'Hello students'
s.upper()
'HELLO STUDENTS'
s.lower()
'hello students'
s.title()
'Hello Students'
s.swapcase()
'HELLO STUDENTS'
```

isupper(), islower(), istitle()

```
s = 'HELLO'
s.isupper()
True
s.islower()
False
s.istitle()
False
s = ''
s.istitle()
False
s.isupper()
False
```

isalnum(), isalpha(), isascii()

```
s ='abcz123'
```

```
s.isalnum()
True
s = 'abc#s'
s.isalnum()
False
s = '123'
s.isalnum()
True
s = '\u03b1\u03b2\u03b3' #alpha-beta-gama in greek language
s
'αβγ'
s.isalnum()
True
s.isalpha()
True
s.isascii()
False
s = '\u0AE7\u0AE8\u0ae9'  #unicodes for gujrati 1 - 2 - 3
s
'□□□'
s.isalnum()
True
s.isalpha()
False
s.isascii()
False
```

isidentifier()

```
s = 'length'
s.isidentifier()
True
s = '1length'
s.isidentifier()
False
s = 'length#'
s.isidentifier()
False
s = 'if'
s.isidentifier()
True
```

isdecimal(), isdigit() and isnumeric()

```
s = '123'
s.isdecimal()
True
s = '123.5'
s.isdecimal()
False
s = '\u0AE7\u0AE8\u0ae9'  #unicodes for gujrati 1 - 2 - 3
s
'□□□'
s.isdecimal()
True
s = '\u0AE7\u0AE8\u0ae9.\u0ae9'
s
```

```
'□□□.□'
s.isdecimal()
False
s = '162\u00B2'  # unicodes for superscript 2
s
'162²'
s.isdecimal()
False
s.isdigit()
True
s = '5\u00BD'  # unicode for 1/2 superscript
s
'5½'
s.isdecimal()
False
s.isdigit()
False
s.isnumeric()
True
s = '\u0661\u0662\u0663' #unicodes for arabic 1 - 2- 3
s
'١٢٣'
s.isdecimal()
True
s.isdigit()
True
s.isnumeric()
True
```

startswith(), endswith(), removesuffix(), removeprefix(), partition(), rpartition()

```
s = 'python is very easy'
s.startswith('python')
True
s.startswith('py')
True
s.startswith('is')
False
s.startswith('is',7)
True
s.endswith('easy')
True
s.endswith('is')
False
s.endswith('is',0, 9)
True
s = 'abcd@gmail.com'
s.endswith('gmail.com')
True
s = 'python is very easy'
s.removeprefix('py')
'thon is very easy'
s.removeprefix('very')
'python is very easy'
s.removesuffix('sy')
'python is very ea'
s.removesuffix('easy')
```

```
'python is very '
s.removesuffix('py')
'python is very easy'
s = 'python is very easy it is very easy'
s.partition('is')
('python ', 'is', ' very easy it is very easy')
ps = s.partition('is')
print(type(ps))
<class 'tuple'>
s.rpartition('is')
('python is very easy it ', 'is', ' very easy')
```

replace()

```
s = 'a-b-c-d-e'  #with count parameter
s.replace('-',',')
'a,b,c,d,e'
s.replace('-',',',2)
'a,b,c-d-e'
s.replace('-',',',7)
'a,b,c,d,e'
```

join()

```
s1 = 'xyz'
s2 = 'abc'
s1.join(s2)
'axyzbxyzc'
s1 = '  xyz  '
s2 = 'abc'
s1.join(s2)
```

```
'a  xyz  b  xyz  c'
```

```
s1 = ' '
```

```
s2 = 'abc'
```

```
s1.join(s2)
```

```
'a b c'
```

```
'/'.join(s2)
```

```
'a/b/c'
```

```
li = ['hi', 'hello', 'how', 'are', 'you']
```

```
ans = ' '.join(li)
```

```
ans
```

```
'hi hello how are you'
```

split()

```
s = 'communication dbms python maths c c++ java'   #max split parameter
```

```
s.split()
```

```
['communication', 'dbms', 'python', 'maths', 'c', 'c++', 'java']
```

```
s.split(' ')
```

```
['communication', 'dbms', 'python', 'maths', 'c', 'c++', 'java']
```

```
s.split(' ',2)
```

```
['communication', 'dbms', 'python maths c c++ java']
```

```
s = 'communication,dbms,python,maths, c, c++, java'
```

```
s.split(',')
```

```
['communication', 'dbms', 'python', 'maths', ' c', ' c++', ' java']
```

```
s.split(',',4)
```

```
['communication', 'dbms', 'python', 'maths', ' c, c++, java']
```

```
s = s.replace(',','-')
```

```
s
```

```
'communication-dbms-python-maths- c- c++- java'
```

```
s.split('-',2)
```

```
['communication', 'dbms', 'python-maths- c- c++- java']
```

rsplit()

```
s = 'communication dbms python maths c c++ java'
```

```
s.rsplit()
```

```
['communication', 'dbms', 'python', 'maths', 'c', 'c++', 'java']
```

```
s.rsplit(' ')
```

```
['communication', 'dbms', 'python', 'maths', 'c', 'c++', 'java']
```

```
s.split(' ',2)
```

```
['communication', 'dbms', 'python maths c c++ java']
```

```
s.rsplit(' ',2)
```

```
['communication dbms python maths c', 'c++', 'java']
```

splitlines()

```
s = 'professional\tcommunication\ndbms\npython\tprogramming\nmaths\nc\
nc++\njava'
```

```
s
```

```
'professional\tcommunication\ndbms\npython\tprogramming\nmaths\nc\nc+
+\njava'
```

```
print(s)
```

```
professional    communication
dbms
python    programming
maths
c
c++
java
```

```
s.splitlines()
```

```
['professional\tcommunication',
 'dbms',
 'python\tprogramming',
 'maths',
 'c',
```

```
 'c++',
 'java']

s.splitlines(keepends=True)

['professional\tcommunication\n',
 'dbms\n',
 'python\tprogramming\n',
 'maths\n',
 'c\n',
 'c++\n',
 'java']

#differentiate between split() and splitlines()

s.splitlines()

['professional\tcommunication',
 'dbms',
 'python\tprogramming',
 'maths',
 'c',
 'c++',
 'java']

s.split()

['professional',
 'communication',
 'dbms',
 'python',
 'programming',
 'maths',
 'c',
 'c++',
 'java']
```

C Style Formatting

```
rollno = 10
name = 'Ravi'
avg = 86.787984554

print('Student name is %s, his roll no is %d and his average marks is
%f'%(name,rollno,avg))
```

```
Student name is Ravi, his roll no is 10 and his average marks is
86.787985
```

```
print('Roll no is :%10d ...'%rollno)
```

```
Roll no is :         10 ...
```

```
print('Name of the student is : %-8s.........'%name)
```

```
Name of the student is : Ravi    .........
```

```
print('Avergae marks is:%6.3f'%avg)
```

```
Avergae marks is:86.788
```

Formatted Printing

```
a = 22
b = 7
c = a/b
print('Division of {} and {} is {}'.format(a,b,c))
print('Division of {0} and {1} is {2}'.format(a,b,c))
print('Division of {2} and {1} is {0}'.format(c,b,a))
```

```
Division of 22 and 7 is 3.142857142857143
Division of 22 and 7 is 3.142857142857143
Division of 22 and 7 is 3.142857142857143
```

```
print('Division of {0:7} and {1:10} is{2:5.4}'.format(a,b,c)) # width

print('Division of {0:^10} and {1:^15} is{2:6.4}'.format(a,b,c))
#alignment

print('Division of {0:<10} and {1:^15} is{2:4.2f}'.format(a,b,c))

print('division of {0:10} and {1:^15} if {2:7.2f}'.format(a,b,c))

print(f'division of {a:10} and {b:^15} is {c:7.2f}')
```