

Python String Operations

String

- A string is a sequence of characters.
- A character is simply a symbol. For example, the English language has 26 characters.
- Computers do not deal with characters, they deal with numbers (binary). Even though you may see characters on your screen, internally it is stored and manipulated as a combination of 0s and 1s.
- This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encodings used.
- In Python, a string is a sequence of Unicode characters. Unicode was introduced to include every character in all languages and bring uniformity in encoding.

```
# defining strings in Python
# all of the following are equivalent
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
                the world of Python"""
print(my_string)
```

```
Hello
Hello
Hello
Hello, welcome to
                the world of Python
```

#Accessing string characters in Python

```
str = 'programiz'
```

```
print('str = ', str)
```

#first character

```
print('str[0] = ', str[0])
```

#last character

```
print('str[-1] = ', str[-1])
```

#slicing 2nd to 5th character

```
print('str[1:5] = ', str[1:5])
```

#slicing 6th to 2nd last character

```
print('str[5:-2] = ', str[5:-2])
```

```
str = programiz
```

```
str[0] = p
```

```
str[-1] = z
```

```
str[1:5] = rogr
```

```
str[5:-2] = am
```

P	R	O	G	R	A	M	I	Z	
0	1	2	3	4	5	6	7	8	9
-9	-8	-7	-6	-5	-4	-3	-2	-1	

```
>>> my_string = 'programiz'
>>> my_string[5] = 'a'
...
TypeError: 'str' object does not support item assignment
>>> my_string = 'Python'
>>> my_string
'Python'
```

```
>>> del my_string[1]
...
TypeError: 'str' object doesn't support item deletion
>>> del my_string
>>> my_string
...
NameError: name 'my_string' is not defined
```

Python String Operations

- Concatenation of Two or More Strings

```
# Python String Operations
str1 = 'Hello'
str2 = 'World!'

# using +
print('str1 + str2 = ', str1 + str2)

# using *
print('str1 * 3 =', str1 * 3)
```

```
>>> # using parentheses
>>> s = ('Hello '
...      'World')
>>> s
'Hello World'
```

● Iterating Through a string

```
# Iterating through a string
count = 0
for letter in 'Hello World':
    if(letter == 'l'):
        count += 1
print(count,'letters found')
```

3 letters found

● Built in functions

- The enumerate() function returns an enumerate object. It contains the index and value of all the items in the string as pairs. This can be useful for iteration.

```
str = 'cold'

# enumerate()
list_enumerate = list(enumerate(str))
print('list(enumerate(str) = ', list_enumerate)

#character count
print('len(str) = ', len(str))
```

```
list(enumerate(str)) = [(0, 'c'), (1, 'o'), (2, 'l'), (3, 'd')]
len(str) = 4
```

Escape Sequence	Description
<code>\newline</code>	Backslash and newline ignored
<code>\\</code>	Backslash
<code>\'</code>	Single quote
<code>\"</code>	Double quote
<code>\a</code>	ASCII Bell
<code>\b</code>	ASCII Backspace
<code>\f</code>	ASCII Formfeed
<code>\n</code>	ASCII Linefeed
<code>\r</code>	ASCII Carriage Return
<code>\t</code>	ASCII Horizontal Tab
<code>\v</code>	ASCII Vertical Tab
<code>\ooo</code>	Character with octal value ooo
<code>\xHH</code>	Character with hexadecimal value HH


```
# using triple quotes
print('''He said, "What's there?''')

# escaping single quotes
print('He said, "What\'s there?")

# escaping double quotes
print("He said, \"What's there?\")
```

```
He said, "What's there?"
He said, "What's there?"
He said, "What's there?"
```

```
>>> print("C:\\Python32\\Lib")
C:\Python32\Lib

>>> print("This is printed\nin two lines")
This is printed
in two lines

>>> print("This is \x48\x45\x58 representation")
This is HEX representation
```

Raw String to ignore escape sequence

```
>>> print(r"This is \x61 \ngood example")
This is \x61 \ngood example
```

format() Method

```
# default(implicit) order
default_order = "{}, {} and {}".format('John','Bill','Sean')
print('\n--- Default Order ---')
print(default_order)

# order using positional argument
positional_order = "{1}, {0} and {2}".format('John','Bill','Sean')
print('\n--- Positional Order ---')
print(positional_order)

# order using keyword argument
keyword_order = "{s}, {b} and {j}".format(j='John',b='Bill',s='Sean')
print('\n--- Keyword Order ---')
print(keyword_order)
```

```
--- Default Order ---
John, Bill and Sean
```

```
--- Positional Order ---
Bill, John and Sean
```

```
--- Keyword Order ---
Sean, Bill and John
```

```

>>> # formatting integers
>>> "Binary representation of {0} is {0:b}".format(12)
'Binary representation of 12 is 1100'

>>> # formatting floats
>>> "Exponent representation: {0:e}".format(1566.345)
'Exponent representation: 1.566345e+03'

>>> # round off
>>> "One third is: {0:.3f}".format(1/3)
'One third is: 0.333'

>>> # string alignment
>>> "|{:<10}|{: ^10}|{:>10}|".format('butter', 'bread', 'ham')
'|butter      |  bread    |      ham|'

```

Old style

```

>>> x = 12.3456789
>>> print('The value of x is %3.2f' %x)
The value of x is 12.35
>>> print('The value of x is %3.4f' %x)
The value of x is 12.3457

```

```
>>> "PrOgRaMiZ".lower()
'programiz'
>>> "PrOgRaMiZ".upper()
'PROGRAMIZ'
>>> "This will split all words into a list".split()
['This', 'will', 'split', 'all', 'words', 'into', 'a', 'list']
>>> ' '.join(['This', 'will', 'join', 'all', 'words', 'into', 'a', 'string'])
'This will join all words into a string'
>>> 'Happy New Year'.find('ew')
7
>>> 'Happy New Year'.replace('Happy', 'Brilliant')
'Brilliant New Year'
```