# Python Strings

Python string is a built-in type text sequence. It is used to handle textual data in python. Python Strings are immutable sequences. Creating Strings are simplest and easy to use in Python.

## Accessing Python Strings

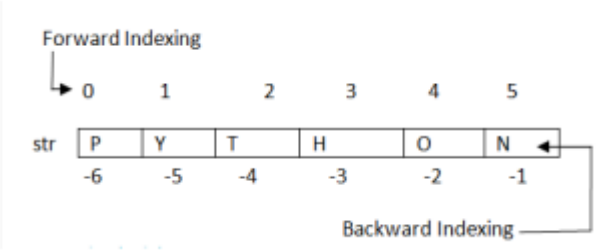
In Python, Strings are stored as individual characters in a contiguous memory location. The benefit of using String is that it can be accessed from both the directions (forward and backward).

Both forward as well as backward indexing are provided using Strings in Python.

Forward indexing starts with 0, 1, 2, 3, ....

Backward indexing starts with -1, -2, -3, -4, ....

**Example**




**Example to print each letter in a Word**

```
# Example to access string

name = "Compusoft"
length = len(name)
for i in range(0, length):
    print(name[i])
```

## String Operators

To perform operation on string, Python provides basically 3 types of Operators that are given below.

- Basic Operators
- Membership Operators
- Relational Operators

## String Basic Operators

There are two types of basic operators in String **+** and *** .**

1. **Concatenation Operator (+)**
   The concatenation operator (+) concatenates two Strings and creates a new String.

   **Example**

```
print("Compu"+"soft")
```

```
var1 = "compu"
var2 = "soft"
print(var1+var2)

# Below will give us an error
print("abc"+3)
```

2. **Replication Operator (*)**
   Replication operator uses two parameters for operation, one is the integer value and the other one is the String argument.
   The Replication operator is used to repeat a string number of times. The string will be repeated the number of times which is given by the integer value.

```
print("Python"*5)
```

| Expression | Output |
|---|---|
| "soono"*2 | 'soonosoono' |
| 3*'1' | '111' |
| '$'*5 | '$$$$$' |

NOTE: We can use Replication operator in any way i.e., int * string or string * int. Both the parameters passed cannot be of same type.

## String Membership Operators

There are two types of Membership operators

- **in** : "in" operator returns true if a character or the entire substring is present in the specified string, otherwise false.
- **not in** :"not in" operator returns true if a character or entire substring does not exist in the specified string, otherwise false.

```
var1 = "mountain"
var2 = "cycle"
var3 = "aeroplane"
var4 = "mount"
var5 = "plane"

print(var5 in var3)
print(var4 in var1)
print(var5 not in var2)
print(var3 in var5)
print(var5 not in var3)
```

## String Relational Operators

All the comparison (relational) operators i.e., (<=, >=, ==, !=, <, >) are also applicable for strings. The Strings are compared based on the ASCII value or Unicode.

### ASCII Seven Bit Character Set


| Dec | Char | | Dec | Chr | Dec | Chr | Dec | Chr |
|---|---|---|---|---|---|---|---|---|
| 0 | NUL | (null) | 32 | Space | 64 | @ | 96 | ` |
| 1 | SOH | (start of heading) | 33 | ! | 65 | A | 97 | a |
| 2 | STX | (start of text) | 34 | " | 66 | B | 98 | b |
| 3 | ETX | (end of text) | 35 | # | 67 | C | 99 | c |
| 4 | EOT | (end of transmission) | 36 | $ | 68 | D | 100 | d |
| 5 | ENQ | (enquiry) | 37 | % | 69 | E | 101 | e |
| 6 | ACK | (acknowledge) | 38 | & | 70 | F | 102 | f |
| 7 | BEL | (bell) | 39 | ' | 71 | G | 103 | g |
| 8 | BS | (backspace) | 40 | ( | 72 | H | 104 | h |
| 9 | TAB | (horizontal tab) | 41 | ) | 73 | I | 105 | i |
| 10 | LF | (NL line feed, new line) | 42 | * | 74 | J | 106 | j |
| 11 | VT | (vertical tab) | 43 | + | 75 | K | 107 | k |
| 12 | FF | (NP form feed, new page) | 44 | , | 76 | L | 108 | l |
| 13 | CR | (carriage return) | 45 | - | 77 | M | 109 | m |
| 14 | SO | (shift out) | 46 | . | 78 | N | 110 | n |
| 15 | SI | (shift in) | 47 | / | 79 | O | 111 | o |
| 16 | DLE | (data link escape) | 48 | 0 | 80 | P | 112 | p |
| 17 | DC1 | (device control 1) | 49 | 1 | 81 | Q | 113 | q |
| 18 | DC2 | (device control 2) | 50 | 2 | 82 | R | 114 | r |
| 19 | DC3 | (device control 3) | 51 | 3 | 83 | S | 115 | s |
| 20 | DC4 | (device control 4) | 52 | 4 | 84 | T | 116 | t |
| 21 | NAK | (negative acknowledge) | 53 | 5 | 85 | U | 117 | u |
| 22 | SYN | (synchronous idle) | 54 | 6 | 86 | V | 118 | v |
| 23 | ETB | (end of trans. block) | 55 | 7 | 87 | W | 119 | w |
| 24 | CAN | (cancel) | 56 | 8 | 88 | X | 120 | x |
| 25 | EM | (end of medium) | 57 | 9 | 89 | Y | 121 | y |
| 26 | SUB | (substitute) | 58 | : | 90 | Z | 122 | z |
| 27 | ESC | (escape) | 59 | ; | 91 | [ | 123 | { |
| 28 | FS | (file separator) | 60 | < | 92 | \ | 124 | \| |
| 29 | GS | (group separator) | 61 | = | 93 | ] | 125 | } |
| 30 | RS | (record separator) | 62 | > | 94 | ^ | 126 | ~ |
| 31 | US | (unit separator) | 63 | ? | 95 | _ | 127 | DEL |

**Example**

```
print("PYTHON" == "PYTHON")
print("PYTHON" == "Python")

print("PYTHON" < "Python")
print("Z" < "z")
```

## String Slice Notation

Python String slice can be defined as a substring which is the part of the string. Therefore further substring can be obtained from a string.

There can be many forms to slice a string, as string can be accessed or indexed from both the direction and hence string can also be sliced from both the directions.

```
# Syntax for slicing

# <string_name>[startIndex:endIndex]
# <string_name>[:endIndex]
# <string_name>[startIndex:]

name = "Anirudh"
print(name[0:4])
print(name[2:6])
print(name[:7])
print(name[3:])
print(name[3:]+name[3:5])

# Note :- startIndex is inclusive and endIndex is exclusive
```

# String Functions and Methods

Python provides various predefined or built-in string functions. They are as follows:

| Methods | Descriptions |
|---|---|
| capitalize() | It capitalizes the first character of the String. |
| count(string,begin,end) | It Counts number of times substring occurs in a String between begin and end index. |
| endswith(suffix ,begin=0,end=n) | 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. |
| index(substring, beginIndex, endIndex) | It throws an exception if string is not found and works same as find() method. |
| isalnum() | It returns True if 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 when all the characters are alphabets and there is at least one character, otherwise False. |
| isdigit() | It returns True if all the characters are digit 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. |
| isupper() | It returns True if characters of a string are in Upper case, otherwise False. |
| isspace() | It returns True if the characters of a string are whitespace, otherwise false. |

| | |
|---|---|
| len(string) | It returns the length of a string. |
| lower() | It converts all the characters of a string to Lower case. |
| upper() | It converts all the characters of a string to Upper Case. |
| startswith(str ,begin=0,end=n) | It returns a Boolean value if the string starts with given str between begin and end. |
| swapcase() | It inverts case of all characters in a string. |
| lstrip() | It removes all leading whitespace of a string and can also be used to remove particular character from leading. |
| rstrip() | It removes all trailing whitespace of a string and can also be used to remove particular character from trailing. |

## 1) Capitalize()

This method capitalizes the first character of the String.

```
name = "akash"

print(name.capitalize())
```

## 2) Count()

This method counts number of times substring occurs in a String between begin and end index

```
string = "Python is easy language."
print(string.count('e'))

var = "g"
print(string.count(var))

print(string.count(var, 0, 6))
```

### 3) Endswith()

This method returns a Boolean value if the string terminates with given suffix between begin and end.

```
string = "Python is easy language."

print(string.endswith('.'))
print(string.endswith('e'))

print(string.endswith('n', 0, 6))
```

### 4) Find()

This method returns the index value of the string where substring is found between begin index and end index.

```
string = "Python is easy language."

print(string.find('o'))
print(string.find('y'))
print(string.find('z'))
print(string.find('i', 7, 9))

var1 = "h"
print(string.find(var1))
```

### 5) Index()

This method returns the index value of the string where substring is found between begin index and end index.

```
string = "Python is easy language."

print(string.index('o'))
print(string.index('y'))
print(string.index('z'))
```

## 6) Isalnum()

This method returns True if characters in the string are alphanumeric i.e., alphabets or numbers and there is at least 1 character. Otherwise it returns False.

```
string1 = "Python is easy language"
string2 = "Python3iseasylanguage"
string3 = "Python3iseasylanguage."

print(string1.isalnum())
print(string2.isalnum())
print(string3.isalnum())
```

## 7) Isalpha()

It returns True when all the characters are alphabets and there is at least one character, otherwise False.

```
string1 = "Python is easy language"
string2 = "Python3iseasylanguage"
string3 = "Pythoniseasylanguage"

print(string1.isalpha())
print(string2.isalpha())
print(string3.isalpha())
```

## 8) Isdigit()

This method returns True if all the characters are digit and there is at least one character, otherwise False.

```
string1 = "abg487"
string2 = "4 45 8 7"
string3 = "587457"

print(string1.isdigit())
print(string2.isdigit())
print(string3.isdigit())
```

## 9) Islower()

This method returns True if the characters of a string are in lower case, otherwise False.

```
string1 = "nikhil"
string2 = "nikhil234"
string3 = "Nikhil234"

print(string1.islower())
print(string2.islower())
print(string3.islower())
```

## 10) Isupper()

This method returns False if characters of a string are in Upper case, otherwise False.

```
string1 = "NIKHIL"
string2 = "NIKHIL1234"
string3 = "NiKHIL"

print(string1.isupper())
print(string2.isupper())
print(string3.isupper())
```

## 11) Isspace()

This method returns True if the characters of a string are whitespace, otherwise false.

```
string1 = "    "
string2 = "\t\n"
string3 = "Ni KHIL"

print(string1.isspace())
print(string2.isspace())
print(string3.isspace())
```

## 12) Len()

This method returns the length of a string

```
string1 = "Python"
string2 = "Python is great"

print(len(string1), len(string2))
```

## 13) Lower()

It converts all the characters of a string to Lower case

```
string1 = "Python is great."
string2 = "PYTHOn is GReaT."
string3 = "PYTHOn is GReaT. 123"

print(string1.lower())
print(string2.lower())
print(string3.lower())
```

## 14) Upper()

This method converts all the characters of a string to upper case.

```
string1 = "Python is great."
string2 = "PYTHOn is GReaT."
string3 = "PYTHOn is GReaT. 123"

print(string1.upper())
print(string2.upper())
print(string3.upper())
```

## 15) Startswith()

This method returns a Boolean value if the string starts with given str between begin and end.

```
string1 = "Python is great."
var = "P"

print(string1.startswith("y"))
print(string1.startswith(var))
print(string1.startswith("i", 7, 10))
```

## 16) Swapcase()

It inverts case of all characters in a string.

```
string1 = "Python is great. Hello123"
print(string1.swapcase())

string2 = string1.swapcase()
print(string2)
```

### 17) Lstrip()

It removes all leading whitespace of a string and can also be used to remove particular character from leading.

```
string1 = "     Python is great."
print(string1.lstrip())

string2 = "@@@@@@@@@@@@Python is great."
print(string2.lstrip("@"))

# It does not remove from the Right side
string3 = "@@@@@@@@@@@@Python is great.@@@@@@"
print(string3.lstrip("@"))
```

### 18) Rstrip()

It removes all trailing whitespace of a string and can also be used to remove particular character from trailing.

```
string1 = "Python is great.          "
print(string1.rstrip())

string2 = "Python is great.######"
print(string2.rstrip("#"))

# It does not remove from the Left side
string3 = "       Python is great.#####"
print(string3.rstrip("#"))
```

## QUIZ

1) Write a Python program to calculate the length of a string entered by User.
2) Write a Python program to reverse a string entered by User.
3) Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string.

   String: 'aeroplane'

   Answer: 'aene'

4) Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '#', except the first char itself.
   String: 'aeroplane'
   Answer: 'aeropl#ne'

## HW

1) Remove the even index characters from the string entered by User.
   String: “PythonisGreat”
   Answer: “PtoGet”

2) Write a Python program to reverse a string if it's length is a multiple of 4.