

# Strings

A string is a sequence of 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 0's and 1's.

This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encoding used.

In Python, string is a sequence of Unicode character.

For more details about unicode

<https://docs.python.org/3.3/howto/unicode.html> (<https://docs.python.org/3.3/howto/unicode.html>)

## How to create a string?

Strings can be created by enclosing characters inside a single quote or double quotes.

Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

```
In [0]: myString = 'Hello'

        print(myString)

        myString = "Hello"
        print(myString)

        myString = '''Hello'''
        print(myString)

Hello
Hello
Hello
```

## How to access characters in a string?

We can access individual characters using indexing and a range of characters using slicing.

Index starts from 0.

Trying to access a character out of index range will raise an `IndexError`.

The index must be an integer. We can't use float or other types, this will result into `TypeError`.

Python allows negative indexing for its sequences.

```
In [0]: myString = "Hello"

        #print first Character
        print(myString[0])

        #print last character using negative indexing
        print(myString[-1])

        #slicing 2nd to 5th character
        print(myString[2:5])

H
o
llo
```

If we try to access index out of the range or use decimal number, we will get errors.

```
In [0]: print(myString[15])

-----
IndexError                                Traceback (most recent call last)
<ipython-input-2-78353fed94bc> in <module>()
----> 1 print(myString[15])

IndexError: string index out of range
```

```
In [0]: print(myString[1.5])

-----
TypeError                                Traceback (most recent call last)
<ipython-input-3-d32f87fd1591> in <module>()
----> 1 print(myString[1.5])

TypeError: string indices must be integers
```

## How to change or delete a string ?

Strings are immutable. This means that elements of a string cannot be changed once it has been assigned.

We can simply reassign different strings to the same name.

```
In [0]: myString = "Hello"
        myString[4] = 's' # strings are immutable

-----
TypeError                                 Traceback (most recent call last)
<ipython-input-6-d63a28c2a378> in <module>()
      1 myString = "Hello"
----> 2 myString[4] = 's' # strings are immutable

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

We cannot delete or remove characters from a string. But deleting the string entirely is possible using the keyword `del`.

```
In [0]: del myString # delete complete string

In [0]: print(myString)

-----
NameError                                 Traceback (most recent call last)
<ipython-input-8-60c083ddb862> in <module>()
----> 1 print(myString)

NameError: name 'myString' is not defined
```

## String Operations

### Concatenation

Joining of two or more strings into a single one is called concatenation.

The `+` operator does this in Python. Simply writing two string literals together also concatenates them.

The `*` operator can be used to repeat the string for a given number of times.

```
In [0]: s1 = "Hello "
        s2 = "Satish"

        #concatenation of 2 strings
        print(s1 + s2)

        #repeat string n times
        print(s1 * 3)

Hello Satish
Hello Hello Hello
```

### Iterating Through String

```
In [0]: count = 0
        for l in "Hello World":
            if l == 'o':
                count += 1
        print(count, ' letters found')

2 letters found
```

## String Membership Test

```
In [0]: print('l' in 'Hello World') #in operator to test membership

True
```

```
In [0]: print('or' in 'Hello World')

True
```

## String Methods

Some of the commonly used methods are lower(), upper(), join(), split(), find(), replace() etc

```
In [0]: "Hello".lower()
```

```
Out[0]: 'hello'
```

```
In [0]: "Hello".upper()
```

```
Out[0]: 'HELLO'
```

```
In [0]: "This will split all words in a list".split()
```

```
Out[0]: ['This', 'will', 'split', 'all', 'words', 'in', 'a', 'list']
```

```
In [0]: ' '.join(['This', 'will', 'split', 'all', 'words', 'in', 'a', 'list'])
```

```
Out[0]: 'This will split all words in a list'
```

```
In [0]: "Good Morning".find("Mo")
```

```
Out[0]: 5
```

```
In [0]: s1 = "Bad morning"

        s2 = s1.replace("Bad", "Good")

        print(s1)
        print(s2)
```

```
Bad morning
Good morning
```

## Python Program to Check where a String is Palindrome or not ?

```
In [0]: myStr = "Madam"

        #convert entire string to either lower or upper
        myStr = myStr.lower()

        #reverse string
        revStr = reversed(myStr)

        #check if the string is equal to its reverse
        if list(myStr) == list(revStr):
            print("Given String is palindrome")
        else:
            print("Given String is not palindrome")

        Given String is palindrome
```

## Python Program to Sort Words in Alphabetic Order?

```
In [0]: myStr = "python Program to Sort words in Alphabetic Order"

        #breakdown the string into list of words
        words = myStr.split()

        #sort the list
        words.sort()

        #print Sorted words are
        for word in words:
            print(word)

        Alphabetic
        Order
        Program
        Sort
        in
        python
        to
        words
```