**Python Strings**

A string is usually a bit of text (sequence of characters). In Python we use ” (double quotes) or ‘ (single quotes) to represent a string.

**1. How to create a String in Python**

There are several ways to create strings in Python.  
1. We can use ‘ (single quotes), see the string str in the following code.  
2. We can use ” (double quotes), see the string str2 in the source code below.  
3. Triple double quotes “”” and triple single quotes ”’ are used for creating multi-line strings in Python. See the strings str3 and str4 in the following example.

# lets see the ways to create strings in Python

str = 'beginnersbook'

print(str)

str2 = "Chaitanya"

print(str2)

# multi-line string

str3 = """Welcome to

Beginnersbook.com"""

print(str3)

str4 = '''This is a tech

blog'''

print(str4)

**Output:**

beginnersbook

Chaitanya

Welcome to

Beginnersbook.com

This is a tech

blog

**2. How to access strings in Python**

A string is nothing but an array of characters so we can use the indexes to access the characters of a it. Just like arrays, the indexes start from 0 to the length-1.

You will get **IndexError** if you try to access the character which is not in the range. For example,  
if a string is of length 6 and you try to access the 8th char of it then you will get this error.

You will get **TypeError** if you do not use integers as indexes, for example if you use a float as an index then you will get this error.

str = "Kevin"

# displaying whole string

print(str)

# displaying first character of string

print(str[0])

# displaying third character of string

print(str[2])

# displaying the last character of the string

print(str[-1])

# displaying the second last char of string

print(str[-2])

Output:

Kevin

K

v

n

i

**3. Python String Operations**

Lets see the operations that can be performed on the strings.

**3.1. Getting a substring in Python – Slicing operation**

We can slice a string to get a **substring** out of it. To understand the concept of **slicing** we must understand the positive and negative indexes in Python (see the example above to understand this). Lets take a look at the few examples of slicing.

str = "Beginnersbook"

# displaying whole string

print("The original string is: ", str)

# slicing 10th to the last character

print("str[9:]: ", str[9:])

# slicing 3rd to 6th character

print("str[2:6]: ", str[2:6])

# slicing from start to the 9th character

print("str[:9]: ", str[:9])

# slicing from 10th to second last character

print("str[9:-1]: ", str[9:-1])

**Output:**

The original string is: Beginnersbook

str[9:]: book

str[2:6]: ginn

str[:9]: Beginners

str[9:-1]: boo

**3.2 Concatenation of strings in Python**

The + operator is used for **string concatenation in Python**. Lets take an example to understand this:

str1 = "One"

str2 = "Two"

str3 = "Three"

# Concatenation of three strings

print(str1 + str2 + str3)

Output:

OneTwoThree

Note: When **+ operator** is used on numbers it adds them but when it used on strings it concatenates them. However if you try to use this between string and number then it will throw TypeError.

For example:

s = "one"

n = 2

print(s+n)

Output:

TypeError: must be str, not int

**3.3 Repetition of string – Replication operator**

We can use \* operator to repeat a string by specified number of times.

str = "ABC"

# repeating the string str by 3 times

print(str\*3)

Output:

ABCABCABC

**3.4 Python Membership Operators in Strings**

**in**: This checks whether a string is present in another string or not. It returns true if the entire string is found else it returns false.  
**not in**: It works just opposite to what “in” operator does. It returns true if the string is not found in the specified string else it returns false.

str = "Welcome to beginnersbook.com"

str2 = "Welcome"

str3 = "Chaitanya"

str4 = "XYZ"

# str2 is in str? True

print(str2 in str)

# str3 is in str? False

print(str3 in str)

# str4 not in str? True

print(str4 not in str)

**Output:**

True

False

True

**3.5 Python – Relational Operators on Strings**

The **relational operators** works on strings based on the ASCII values of characters.  
The ASCII value of a is 97, b is 98 and so on.  
The ASCII value of A is 65, B is 66 and so on.

str = "ABC"

str2 = "aBC"

str3 = "XYZ"

str4 = "XYz"

# ASCII value of str2 is > str? True

print(str2 > str)

# ASCII value of str3 is > str4? False

print(str3 > str4)

Output:

True

False