

Definition of a Strings:

In python, consecutive sequence of characters is known as a string. An individual character in a string is accessed using a subscript (index). The subscript should always be an integer (positive or negative). In python, strings indices begins with 0 in forward direction and -1 in backward direction.

Example

Declaring a string in python

```
>>>str="Computer Science"
```

```
>>>print(str)
```

Computer Science

To access the **first character** of the string

```
>>>print(str[0])
```

C

To access the **fourth character** of the string

```
>>>print(str[3])
```

p

To access the **last character** of the string

```
>>>print(str[-1])
```

```
>>e
```

To access the **third last character** of the string

```
>>>print(str[-3])
```

n

Consider the given figure

String A	H	E	L	L	O
Positive Index	0	1	2	3	4
Negative Index	-5	-4	-3	-2	-1

Important points about accessing elements in the strings using subscripts

- Positive subscript helps in accessing the string from the beginning
- Negative subscript helps in accessing the string from the end.
- Subscript 0 or -ve n (where n is length of the string) displays the first element.

Example : A[0] or A[-5] will display “H”

Subscript 1 or -ve (n-1) displays the second element.

Creating and initializing strings

A literal/constant value to a string can be assigned using a single quotes, double quotes or triple quotes.

Enclosing the string in single quotes

Example

```
>>>print ('A friend in need is a friend indeed')
```

A friend in need is a friend indeed

Example

```
>>>print(' This book belongs to Raghav\'s sister')
```

This book belongs to Raghav's sister

As shown in example 2, to include the single quote within the string it should be preceded by a backslash.

Enclosing the string in double quotes

Example

```
>>>print("A room without books is like a body without a soul.")
```

A room without books is like a body without a soul.

Enclosing the string in triple quotes

Example

```
>>>life="" "\ Live as if you were to die tomorrow.
```

```
Learn as if you were to live forever.\
```

```
---- Mahatma Gandhi ""
```

```
>>> print( life)
```

```
"Live as if you were to die tomorrow.
```

```
Learn as if you were to live forever."
```

```
---- Mahatma Gandhi
```

Triple quotes are used when the text is multiline. In the above example, backslash (\) is used as an escape sequence. An escape sequence is nothing but a special character that has a specific function. As shown above, backslash (\) is used to escape the double quote.

Strings are immutable

Strings are immutable means that the contents of the string cannot be changed after it is created. Let us understand the concept of immutability with help of an example.

Example

```
>>>str='honesty'
```

```
>>>str[2]='p'
```

TypeError: 'str' object does not support item assignment

Python does not allow the programmer to change a character in a string. As shown in the above example, str has the value "honesty". An attempt to replace "n" in the string by "p" displays a TypeError.

Traversing a string

Traversing a string means accessing all the elements of the string one after the other by using the subscript. A string can be traversed using:

for loop or while loop.

String traversal using for loop

```
a='Welcome'
```

```
for i in a:
```

```
    print(i, '-', end='')
```

output is W - e - l - c - o - m - e -

A is assigned a string literal "Welcome". On execution of the for loop, the characters in the string are printed till the end of the string.

String traversal using while loop

```
a='Welcome'
```

```
i=0
```

```
while i<len(a):
```

```
    print(a[i],end="")
```

```
i=i+1
```

output is Welcome

a is assigned a string literal "Welcome" i is assigned value 0. The len() function calculates the length of the string. On entering the while loop, the interpreter checks the condition. If the condition is true, it enters the loop. The first character in the string is displayed. The value i is incremented by 1. The loop continues till value i is less than len-1. The loop finishes as soon as the value of i becomes equal to len-1.

Strings basic operators

+ (Concatenation) operator: The + operator joins the text on both sides of the Operator. Consider the following example.

```
>>> "Tea"+"Pot"
```

```
"TeaPot"
```

```
>>> "1"+"1"
```

```
"11"
```

To give a white space between the two words, insert a space before the closing single quote of the first literal.

*** (Replication) operator:**

The * operator replicates the string the number of times the value given on the other side. Consider the following example

```
>>> 3*"go "
```

```
"gogogo"
```

```
>>> "1"*2
```

```
"11"
```

String Membership Operators

in operator:

The operator displays true if the string contains the given character or the sequence of characters otherwise it returns false. Consider the following example

```
>>>A="Save Earth"
```

```
>>> "S" in A
```

```
True
```

```
>>>"Save" in A
```

```
True
```

```
>>"SE" in A
```

```
False
```

Not in operator:

The operator displays true if the string does not contain the given character or the sequence of characters otherwise it returns false. Consider the following example

```
>>>"SE" not in "Save Earth"
```

```
True
```

```
>>>"Save " not in "Save Earth"
```

```
False
```

```
>>>"ve " not in "Save Earth"
```

```
False
```

String Slices

The term '*string slice*' refers to a part of the string, where strings are sliced using a range of indices. For a string say name[n : m] where n and m are integers and legal indices. Python will return a slice of the string by returning the characters falling between indices n and m starting at n,n+1,n+2... till m-1.

Let's understand Slicing in strings with the help of few examples.

String word	a	m	a	z	i	n	g
Positive Index	0	1	2	3	4	5	6
Negative Index	-7	-6	-5	-4	-3	-2	-1

Example

```
>>>word="amazing"  
>>> print(word[0:3])  
ama
```

The print statement prints the substring starting from subscript 0 and ending at subscript 2 .

Example

```
>>>print(word[3:])  
zing
```

Omitting the second index, directs the python interpreter to extract the substring till the end of the string

Example

```
>>>print(word[:3])
```

```
ama
```

Omitting the first index, directs the python interpreter to extract the substring before the second index starting from the beginning.

Example

```
>>>print(word[:])
```

```
amazing
```

Omitting both the indices, directs the python interpreter to extract the entire string starting from 0 till the last index

Example

```
>>>print( word[-2:])
```

```
ng
```

For negative indices the python interpreter counts from the right side (also shown above). So the last two letters are printed.

Example

```
>>>print (word[:-2])
```

```
amazi
```

Omitting the first index, directs the python interpreter to start extracting the substring from the beginning. Since the negative index indicates slicing from the end of the string. So the entire string except the last two letters is printed.

Prepared by Rani Sastry, PGT
AECS-4, Mumbai