

# Strings: string slices, immutability, string functions and methods, string module

# Strings

- String is a **sequence of characters**.
- String may contain **alphabets, numbers and special characters**.
- Usually strings are enclosed within a **single quotes and double quotes**.
- Strings is **immutable** in nature.
- **Example:**

a='hello world'

b="Python"

# Inbuilt String functions

- Python mainly contains 3 inbuilt string functions.
- They are
  - `len()`
  - `max()`
  - `min()`
- `len()`- Find out the length of characters in string
- `min()`- Smallest value in a string based on ASCII values
- `max()`- Largest value in a string based on ASCII values

# What is ASCII values

L.O : Explain the function of ASCII code.

## ASCII

### • HOW ASCII WORKS IN A COMPUTER SYSTEM?



Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z

Decimal	Character
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z

# Example for Inbuilt string functions

```
name=input("Enter Your name:")
```

```
print("Welcome",name)
```

```
print("Length of your name:",len(name))
```

```
print("Maximum value of character in your name",  
max(name))
```

```
print("Minimum value of character in your name",min(name))
```

# OUTPUT

Enter Your name:PRABHAKARAN

Welcome PRABHAKARAN

Length of your name: 11

Maximum value of character in your name R

Minimum value of character in your name A

# Strings Concatenation

- The **+** operator used for string concatenation.

## Example:

```
a="Hai"
```

```
b="how are you"
```

```
c=a+b
```

```
print(c)
```

```
Haihow are you
>>> a="Hai"
>>> b=" how are you"|
>>> c=a+b
>>> print(c)
Hai how are you
```

```
>>> a="Hai"
>>> b=' how are you'
>>> c=a+b
>>> print(c)
Hai how are you
```

# Operators on String

- The Concatenate strings with the “\*” operator can create multiple concatenated copies.
- Example:

```
>>> print("Python"*10)
```

```
PythonPythonPythonPythonPythonPython  
PythonPythonPythonPython
```

```
>>> print("Python"*10)
```

```
PythonPythonPythonPythonPythonPythonPythonPythonPythonPython
```

# String Slicing

- Slicing operation is used to return/select/slice the particular substring based on user requirements.
- A segment of string is called **slice**.
- **Syntax:** `string_variablename [ start:end ]`

# String Slice example

s="Hello"

```
>>> s="hello"
>>> s[1:4]
'ell'
>>> s[1:]
'ello'
>>> s[:]
'hello'
>>> s[1:100]
'ello'
>>> s[-1]
'o'
>>> s[:]
'hello'
>>> s[:-3]
'he'
```

H	e	l	l	o
0	1	2	3	4
-5	-4	-3	-2	-1

Python  
Programming

# Strings are immutable

- Strings are **immutable character sets**.
- Once a string is generated, **you cannot change any character within the string**.

```
>>> a="python program"
```

```
>>> a[0]
```

```
'p'
```

```
>>> a[0]="b"
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#16>", line 1, in <module>
```

```
    a[0]="b"
```

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

```
>>> a[0]
```

```
'p'
```

# String Comparision

- We can compare two strings using **comparision operators such as ==, !=, <, <=, >, >=**
- Python compares strings based on their corresponding ASCII values.

Python  
Programming

# Example of string comparision

```
str1="green"
```

```
str2="black"
```

```
print("Is both Equal:", str1==str2)
```

```
print("Is str1> str2:", str1>str2)
```

```
print("Is str1< str2:", str1<str2)
```

OUTPUT:

Is both Equal: False

Is str1> str2: True

Is str1< str2: False

# String formatting operator

- String formatting operator **%** is unique to strings.
- **Example:**

```
print("My name is %s and i secured %d  
marks in python" % ("Arbaz",92))
```

- **Output:**

My name is Arbaz and i secured 92 marks in python

# String functions and methods

<b>len()</b>	<b>min()</b>	<b>max()</b>	<b>isalnum()</b>	<b>isalpha()</b>
<b>isdigit()</b>	<b>islower()</b>	<b>isupper()</b>	<b>isspace()</b>	<b>isidentifier()</b>
<b>endswith()</b>	<b>startswith()</b>	<b>find()</b>	<b>count()</b>	<b>capitalize()</b>
<b>title()</b>	<b>lower()</b>	<b>upper()</b>	<b>swapcase()</b>	<b>replace()</b>
<b>center()</b>	<b>ljust()</b>	<b>rjust()</b>	<b>center()</b>	<b>rstrip()</b>
<b>rstrip()</b>	<b>strip()</b>			

## i) Converting string functions

<code>capitalize()</code>	Only First character capitalized
<code>lower()</code>	All character converted to lowercase
<code>upper()</code>	All character converted to uppercase
<code>title()</code>	First character capitalized in each word
<code>swapcase()</code>	Lower case letters are converted to Uppercase and Uppercase letters are converted to Lowercase
<code>replace(old,new)</code>	Replaces old string with nre string

## Program:

```
str=input("Enter any string:")  
print("String Capitalized:", str.capitalize())  
print("String lower case:", str.lower())  
print("String upper case:", str.upper())  
print("String title case:", str.title())  
print("String swap case:", str.swapcase())  
print("String replace case:",str.replace("python","python programming"))
```

## Output:

```
Enter any string: Welcome to python  
String Capitalized: Welcome to python  
String lower case: welcome to python  
String upper case: WELCOME TO PYTHON  
String title case: Welcome To Python  
String swap case: wELCOME TO PYTHON  
String replace case: Welcome to python programming
```

## ii) Formatting String functions

<code>center(width)</code>	Returns a string centered in a field of given width
<code>ljust(width)</code>	Returns a string left justified in a field of given width
<code>rjust(width)</code>	Returns a string right justified in a field of given width
<code>format(items)</code>	Formats a string

Programming

## Program:

```
a=input("Enter any string:")  
print("Center alignment:", a.center(20))  
print("Left alignment:", a.ljust(20))  
print("Right alignment:", a.rjust(20))
```

### Output:

```
Enter any string:welcome  
Center alignment:      welcome  
Left alignment: welcome  
Right alignment:      welcome
```

### iii) Removing whitespace characters

lstrip()

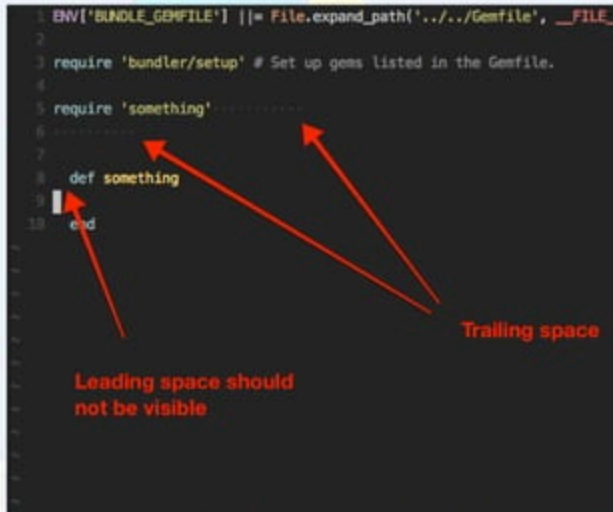
Returns a string with leading whitespace characters removed

rstrip()

Returns a string with trailing whitespace characters removed

strip()

Returns a string with leading and trailing whitespace characters removed



The screenshot shows a code editor with the following Python code:

```
1 ENV['BUNDLE_GEMFILE'] ||= File.expand_path('.././Gemfile', __FILE__)
2
3 require 'bundler/setup' # Set up gems listed in the Gemfile.
4
5 require 'something'
6
7
8 def something
9   end
10
```

Annotations with red arrows:

- An arrow points to the space before `def something` on line 8, with the text "Leading space should not be visible".
- An arrow points to the space after `require 'something'` on line 5, with the text "Trailing space".

# Program

```
a=input("Enter any string:")  
print("Left space trim:",a.lstrip())  
print("Right space trim:",a.rstrip())  
print("Left and right trim:",a.strip())
```

Output:

```
Enter any string:      welcome  
Left space trim: welcome  
Right space trim:      welcome  
Left and right trim: welcome
```

## iv) Testing String/Character

isalnum()	Returns true if all characters in string are alphanumeric and there is atleast one character
isalpha()	Returns true if all characters in string are alphabetic
isdigit()	Returns true if string contains only number character
islower()	Returns true if all characters in string are lowercase letters
isupper()	Returns true if all characters in string are uppercase letters
isspace()	Returns true if string contains only whitespace characters.

# Program

```
a=input("Enter any string:")  
print("Alphanumeric:",a.isalnum())  
print("Alphabetic:",a.isalpha())  
print("Digits:",a.isdigit())  
print("Lowecase:",a.islower())  
print("Upper:",a.isupper())
```

Output:

```
Enter any string:python  
Alphanumeric: True  
Alphabetic: True  
Digits: False  
Lowecase: True  
Upper: False
```

## v) Searching for substring

Endswith()	Returns true if the strings ends with the substring
Startswith()	Returns true if the strings starts with the substring
Find()	Returns the lowest index or -1 if substring not found
Count()	Returns the number of occurrences of substring

Programming

# Program

```
a=input("Enter any string:")  
print("Is string ends with thon:", a.endswith("thon"))  
print("Is string starts with good:", a.startswith("good"))  
print("Find:", a.find("ython"))  
print("Count:", a.count("o"))
```

## Output:

```
Enter any string : welcome to python  
Is string ends with thon: True  
Is string starts with good: False  
Find: 12  
Count: 3
```

# String Modules

- String module contains a number of functions to process standard Python strings
- **Mostly used string modules:**
  - `string.upper()`
  - `string.lower()`
  - `string.split()`
  - `string.join()`
  - `string.replace()`
  - `string.find()`
  - `string.count()`

# Example

```
import string
text="Monty Python Flying Circus"
print("Upper:", string.upper(text))
print("Lower:", string.lower(text))
print("Split:", string.split(text))
print("Join:", string.join(string.split(text),"+"))
print("Replace:", string.replace(text,"Python", "Java"))
print("Find:", string.find(text,"Python"))
print("Count", string.count(text,"n"))
```

# Output

Upper: "MONTY PYTHON FLYING CIRCUS"

Lower: "monty python flying circus"

Split: ['Monty', 'Python', 'Flying', 'Circus']

Join : Monty+Python+Flying+Circus

Replace: Monty Java Flying Circus

Find: 7

Count: 3