



String

Agenda

1

Introduction to string

2

String slicing

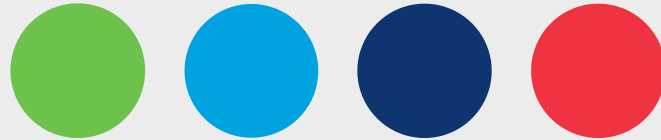
3

Formatting string

4

Built-in methods

Introduction to string



String (str)

- **str** is the built-in string class.
- str represents one or more sequence of characters.
- It is enclosed in either single or double quotes.
- It can contain alphabets, numbers, special characters and white spaces.

Program:

```
s1 = 'Wipro'
s2 = "Welcome to Wipro"
s3 = '@'
print(type(s1))
print(type(s2))
print(type(s3))
```

Output:

```
class <'str'>
class <'str'>
class <'str'>
```

String (str) continued..

Different ways of creating a string:

1. Using single quotes: `'Hi'`, `'Welcome to Wipro'`, `'@123'`
2. Using double quotes: `"Wipro Technologies"`, `"P#y$t @h*o%n"`, `"@"`
3. Using triple single or double quotes for multi line string:
`'''Python is an interpreted, high-level,
general-purpose programming language.
Created by Guido van Rossum and first released in 1991.'''`

String (str) continued..

- Strings are **immutable**.
- Any changes made to the string content is stored in a new location.
- The original content will remain unaffected.
- Execute the below programs and check whether is there any difference in the outputs.

Program:1

```
s1 = "hello"  
s1.upper()  
print(s1)
```

Program:2

```
s1 = "hello"  
s2 = s1.upper()  #s1 = s1.upper()  
print(s1,s2)     #print(s1)
```

String (str) continued..

- String provides index based access of each character.
- Both positive and negative indexing is supported.

```
s1 = 'Welcome'
```

0	1	2	3	4	5	6
W	e	l	c	o	m	e
-7	-6	-5	-4	-3	-2	-1

```
print(s1[0]) #prints W  
print(s1[-4]) #prints c
```

String (str) continued..

Using for loop with string:

Program:1

```
s1 = 'Wipro Technologies'
for letter in s1:
    print(letter,end=' ')
```

Output:

W i p r o T e c h n o l o g i e s

Program:2

```
s1 = 'Wipro Technologies'
for i in range(0,len(s1)):
    print(s1[i],end=' ')
```

len function gives the
count of characters
available in s1

String slicing



String slicing

- Slicing is extracting a part of the string using indexes.
- *Syntax* : `s1[start_index : end_index]`
- End_index is excluded.
- `s1 = "Wipro"`

Slicing	Result
<code>s1[0:2]</code> or <code>s1[:2]</code>	Wi
<code>s1[0:5]</code> or <code>s1[0:]</code> or <code>s1[:5]</code>	Wipro
<code>s1[-3:-1]</code>	pr
<code>s1[-4:]</code>	ipro

String slicing continued..

- Slicing operation returns a new string which is stored in a new location.

Program:

```
s1 = 'Wipro'  
print(s1,id(s1)) #prints Wipro and address of Wipro  
s2 = s1[1:5]  
print(s2,id(s2)) #prints ipro and address of ipro
```

Formatting string



Formatting string

- We can concatenate strings simply by using `+` operator.

Program:

```
s1 = 'Wipro'
```

```
s2 = 'Technologies'
```

```
print(s1+s2)           #WiproTechnologies
```

```
print(s1+" "+s2)       #Wipro Technologies
```

Formatting string continued..

- However concatenating a string with other data types like int or float results in **TypeError**.

Execute the code:

```
s1 = 'My employee no is '
```

```
num = 2233
```

```
print(s1+num) #Error in line 3
```

- **format()** method is used to overcome this problem.

Formatting string continued..

- Use placeholders { } in the string, later these placeholders can be replaced by values.

Program:

```
s1 = 'My employee no is {}'  
num = 2233  
print(s1.format(num))
```

Output:

My employee no is 2233

Formatting string continued..

Program:

```
name = 'Arun'
```

```
age = 25
```

```
s1 = 'My name is {} and I am {}'
```

```
print(s1.format(name,age))
```


Output:

```
My name is Arun and I am 25
```


Formatting string continued..

Program:

```
s1 = '''Welcome to {0} Technologies, {0} is an {1}
      multinational corporation.'''
p1 = 'Wipro'
p2 = 'Indian'
print(s1.format(p1,p2))
```



Output:

Welcome to Wipro Technologies, Wipro is an Indian multinational corporation.

Built-in methods



Built-in methods

Some important built-in string methods:

Method	Explanation
upper()	Converts the entire string to uppercase.
lower()	Converts the entire string to lowercase.
isupper()	Returns True if all the characters are in uppercase.
islower()	Returns True if all the characters are in lowercase.
isdigit()	Returns True if all the characters are digits.
isalpha()	Returns True if all the characters are alphabets.
capitalize()	Converts the first character to uppercase.
swapcase()	Toggles the cases. Upper to Lower and Lower to Upper.

Built-in methods continued..

Method	Explanation
<code>strip()</code>	Removes spaces from front and end.
<code>startswith(value)</code>	Returns True if the string starts with the specified value.
<code>count(value)</code>	Returns the number of times a specified value occurs in a string.
<code>replace(old_value, new_value)</code>	Returns a string where all occurrences of old_value is replaced with new_value.
<code>index(value)</code>	Searches the string for a specified value and returns the index of where it was found.
<code>split(separator)</code>	Splits the string at the specified separator, and returns a list.

Example:1

```
s1 = "wipro"  
print(s1.upper())  
s2 = "HELLO"  
print(s2.lower())  
s3 = "ABC"  
print(s3.isupper())  
s4 = 'xyz'  
print(s4.islower())
```

Output:

WIPRO

hello

True

True

Reminder...!!
None of the
methods will affect
the original content.

Changes are
always stored in a
new location.

That's why strings
are ***immutable***.

Example:2

```
s5 = 'hi hello how are you'
print(s5.count('h'))

s6 = 'admin'
print(s6.index('m'))

s7 = 'Python session'
li = s7.split(' ')
print(li)

s8 = 'chad@marcel@arun'
print(s8.replace('@','$'))
```

Output:

3

2

['Python', 'session']

chad\$marcel\$arun



Thank you