



Python Strings

Strings in Python



What's in it for you?

- ▶ **What is a String?**
- ▶ **Creating String in Python**
- ▶ **String Methods**
- ▶ **String Indexing**
- ▶ **Hands on Demo**

► What is a String?

A String is a sequence of characters



In Python, we have `str` as data type for string

▶ String Methods

Strings can be defined by surrounding characters inside a single quote or double-quotes

Using Single Quotes
`str1 = 'Hello INDIA'`

Using Double Quotes
`str2 = "Hello INDIA"`

Triple-quotes are generally used for multiline string or docstrings

```
str3 = '''Human psychology is  
an interesting subject'''
```

▶ Creating String in Python

```
In [1]: ▶ str1 = 'Hi Sam'  
print(str1)
```

Hi Sam

```
In [2]: ▶ type(str1)
```

Out[2]: str

```
In [4]: ▶ str2 = "Hello Angela!!"  
print(str2)
```

Hello Angela!!

```
In [6]: ▶ str3 = '''Hi everyone  
Today we will learn about  
Strings in Python'''  
print(str3)
```

Hi everyone
Today we will learn about
Strings in Python

Python Slicing



python

```
elif _operation == "MIRROR":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif _operation == "MIRROR_X":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects
print("Selected" + str(modifier_ob))
#mirror_ob.select = 0
bpy.context.scene.objects
print("Selected" + str(modifier_ob))
```


What is slicing in Python?



SLICING IS THE EXTRACTION OF A PART OF A STRING.

SYNTAX:

Object[start : stop]

► String Indexing

Reverse Index

[-7] [-6] [-5] [-4] [-3] [-2] [-1]

H	E	A	L	T	H	Y
---	---	---	---	---	---	---

[0] [1] [2] [3] [4] [5] [6]

Forward Index

```
[77]: course = "Hello MAC108"
print(course[0])
print(course[1])
print(course[2])
print(course[3])
print(course[4])
print(course[5])
print(course[6])
print(course[7])
print(course[8])
print(course[9])
print(course[10])
print(course[11])
```

H
e
l
l
o

M
A
C
1
0
8

```
[78]: course[0:]
```

```
[78]: 'Hello MAC108'
```

```
[80]: course[0:12]
```

```
[80]: 'Hello MAC108'
```

```
[82]: course[5]
```

```
[82]: ' '
```

```
[83]: course[0:2]
```

```
[83]: 'He'
```

```
[84]: course[4:9]
```

```
[84]: 'o MAC'
```

```
[97]: course = "Hello MAC108"  
course[-12:-1]
```

```
[97]: 'Hello MAC10'
```

```
[98]: course[-11:-1]
```

```
[98]: 'ello MAC10'
```

```
[99]: course[-5:-1]
```

```
[99]: 'AC10'
```

```
[100]: course[-9:-3]
```

```
[100]: 'lo MAC'
```


SYNTAX:

Object[start : stop : step]

```
[102]: course = "Hello MAC108"
```

```
[103]: course[0:12:2]
```

```
[103]: 'Hl0MC0'
```

```
[104]: course[0:12:3]
```

```
[104]: 'HlM1'
```

```
[105]: course[0:12:1]
```

```
[105]: 'Hello MAC108'
```

```
[106]: course[2:12:2]
```

```
[106]: 'l0MC0'
```

```
[107]: course[1:12:3]
```

```
[107]: 'eoA0'
```

```
[113]: mycourse = slice(1, 9, 1)
print(course)
print(course[mycourse])
```

```
Hello MAC108
ello MAC
```

```
[111]: mycourse = slice(1, 9, 2)
print(course)
print(course[mycourse])
```

```
Hello MAC108
el A
```

```
[112]: mycourse = slice(3, 11, 3)
print(course)
print(course[mycourse])
```

```
Hello MAC108
lM1
```

String Formatting in Python



python



1. Using `%` operator (Old Style Formatting):

```
[6]: name = "Python"  
age = 32  
greeting = "Hi, %s! You are %d years old." % (name, age)  
print(greeting)
```

Hi, Python! You are 32 years old.

- `%s` is a placeholder for a string.
- `%d` is a placeholder for an integer.

2. Using `str.format()` method:

```
[8]: name = "Python"
age = 32
greeting = "Hi, {}! You are {} years old.".format(name, age)
print(greeting)
```

Hi, Python! You are 32 years old.

- `{}` is a placeholder for values.

In this case, `{}` is used for string substitution, and the values inside the `format` method (in this case, `name` and `age`) will replace the corresponding placeholders. The resulting `formatted_string` will be "Hi, Python! You are 32 years old."

3. Using f-strings (Python 3.6 and later):

```
[9]: name = "Python"  
age = 32  
greeting = f"Hi, {name}! You are {age} years old."  
print(greeting)
```

Hi, Python! You are 32 years old.

- `f"..."` denotes an f-string, a feature introduced in Python 3.6.
- `{name}` and `{age}` are placeholders for values.

In this case, the f-string allows you to embed expressions inside string literals, and the expressions within curly braces `{ }` are evaluated and replaced with their values. The resulting `formatted_string` will be "Hi, Python! You are 32 years old."

4. Using `join` method:

```
[10]: name = "Python"
      age = 32
      greeting = " ".join(["Hi,", name + "!", "You are", str(age), "years old."])
      print(greeting)
```

Hi, Python! You are 32 years old.

- `join` is a method used to concatenate the elements of an iterable, such as a list, into a single string.
- `name`, `str(age)`, and the string literals are elements of the list that will be joined.

In this case, the list `["Hi,", name + "!", "You are", str(age), "years old."]` is joined with spaces in between each element. The resulting `formatted_string` will be "Hi, Python! You are 32 years old."

5. Using `f-string` with expressions (Python 3.8 and later):

```
[12]: name = "Python"
      age = 32
      greeting = f"Hi, {name.upper()}! You are {age} years old."
      print(greeting)
```

Hi, PYTHON! You are 32 years old.

- `f"..."` denotes an f-string.
- `{name.upper()}` is an expression inside the f-string, converting the string in `name` to uppercase.
- `{age}` is another expression, representing the value of the `age` variable.

The resulting `greeting` will be "Hi, PYTHON! You are 32 years old." The f-string allows for expressions within curly braces `{ }` to be evaluated and inserted into the string.

Change and Delete String Characters

[14]:

```
name = "python"  
name[0] = "f"  
print(name)
```



```
-----  
TypeError                                Traceback (most recent call last)  
Cell In[14], line 2  
      1 name = "python"  
----> 2 name[0] = "f"  
      3 print(name)  
  
TypeError: 'str' object does not support item assignment
```

[15]:

```
name = "python"  
del name  
print(name)
```



```
-----  
NameError                                Traceback (most recent call last)  
Cell In[15], line 3  
      1 name = "python"  
      2 del name  
----> 3 print(name)  
  
NameError: name 'name' is not defined
```

Python String Operations

```
[17]: # Concatenation:
name1 = "Python"
name2 = " Programming"
result = name1 + name2
print(result)
```

Python Programming

```
[18]: name3 = 5
sum = name1 + name3
print(sum)
```

TypeError

Traceback (most recent call last)

Cell In[18], line 2

```
1 name3 = 5
----> 2 sum = name1 + name3
      3 print(sum)
```

TypeError: can only concatenate str (not "int") to str

Python String Operations

```
[19]: name3 = str(5)
      sum = name1 + name3
      print(sum)
```

Python5

```
[20]: name3 = 5
      sum = name1 + str(name3)
      print(sum)
```

Python5

```
[23]: name1 = "Python"
      name2 = "Programming"
      result = name1 + " " + name2
      print(result)
```

Python Programming

Python String Operations

```
[25]: name1 = "Python"  
      new_name1 = name1 * 3  
      print(new_name1)
```

PythonPythonPython

```
[26]: name1 = "Python"  
      new_name1 = (name1 + " ") * 3  
      print(new_name1)
```

Python Python Python

Python String Operations

```
[28]: # Length:  
text = "Python Programming"  
length = len(text)  
print(length)
```

18

```
[29]: # Lowercase and Uppercase:  
text = "Python"  
lowercase_text = text.lower()  
uppercase_text = text.upper()  
print(lowercase_text, uppercase_text)
```

python PYTHON

Python String Operations

```
[30]: # Replace:  
sentence = "I like Java"  
updated_sentence = sentence.replace("Java", "Python")  
print(updated_sentence)
```

I like Python

```
[31]: # Split:  
sentence = "Python is fun"  
words = sentence.split()  
print(words)
```

['Python', 'is', 'fun']

Python String Operations

```
[33]: # Stripping:  
whitespace_text = "    Python    "  
stripped_text = whitespace_text.strip()  
print(whitespace_text)  
print(stripped_text)
```

```
    Python  
Python
```

```
[34]: # Checking Substring:  
text = "Python is easy"  
contains_easy = "easy" in text  
print(contains_easy)
```

```
True
```





