1. String Declaration
2. Indexing and Slicing
3. String Operators
4. Escape Sequence in Python

Python – String

In Python, string is an immutable sequence data type. It is the sequence of characters wrapped inside single, double, or triple quotes.

The followings are valid string literals in Python.

```
'This is a string in Python' # string in single quotes
"This is a string in Python" # string in double quotes
'''This is a string in Python''' # string in triple quotes
"""This is a string in Python""" # string in triple double-quotes
```

A string literal can be assigned to a variable, as shown below.

```
str1='This is a string in Python'
print(str1)
str2="This is a string in Python"
print(str2)
```

Multi-line strings must be embed in triple quotes, as shown below.

```
str1='''This is
the first
Multi-line string.
'''
print(str1)

str2="""This is
the second
Multi-line
string."""
print(str2)
```

```
#Using single quotes
str1 = 'Hello Python'
print(str1)
#Using double quotes
str2 = "Hello Python"
print(str2)

#Using triple quotes
str3 = '''Triple quotes are generally used for
    represent the multiline or
    docstring'''
print(str3)
```

If a string literal required to embed double quotes as part of a string then, it should be put in single quotes. Likewise, if a string includes a single quote as a part of a string then, it should be written in double quotes.

```
str1='Welcome to "Python Tutorial" on TutorialsTeacher'
print(str1)

str2="Welcome to 'Python Tutorial' on TutorialsTeacher"
print(str2)
```

Output:

```
Welcome to "Python Tutorial" from TutorialsTeacher
Welcome to 'Python Tutorial' on TutorialsTeacher
```

Strings indexing and splitting

Like other languages, the indexing of the Python strings starts from 0. For example, The string "HELLO" is indexed as given in the below figure.

str = "HELLO"

H	E	L	L	O
0	1	2	3	4

str[0] = 'H'

str[1] = 'E'

str[2] = 'L'

str[3] = 'L'

str[4] = 'O'

Consider the following example:

```
str = "HELLO"
print(str[0])
print(str[1])
print(str[2])
print(str[3])
print(str[4])
# It returns the IndexError because 6th index doesn't exist
print(str[6])
```

Output:

```
H
E
L
L
O
IndexError: string index out of range
```

As shown in Python, the slice operator [] is used to access the individual characters of the string. However, we can use the : (colon) operator in Python to access the substring from the given string. Consider the following example.

str = "HELLO"

H	E	L	L	O
0	1	2	3	4

str[0] = 'H'

str[1] = 'E'

str[2] = 'L'

str[3] = 'L'

str[4] = 'O'

str[:] = 'HELLO'

str[0:] = 'HELLO'

str[:5] = 'HELLO'

str[:3] = 'HEL'

str[0:2] = 'HE'

str[1:4] = 'ELL'

str = “HELLO”

H	E	L	L	O
-5	-4	-3	-2	-1

str[-1] = ‘O’

str[-2] = ‘L’

str[-3] = ‘L’

str[-4] = ‘E’

str[-5] = ‘H’

str[-3:-1] = ‘LL’

str[-4:-1] = ‘ELL’

str[-5:-3] = ‘HE’

str[-4:] = ‘ELLO’

str[::-1] = ‘OLLEH’

Reassigning Strings

Updating the content of the strings is as easy as assigning it to a new string. The string object doesn't support item assignment i.e., A string can only be replaced with new string since its content cannot be partially replaced. Strings are immutable in Python.

Consider the following example.

Example 1

```
str = "HELLO"
str[0] = "h"
print(str)
```

Output:

```
Traceback (most recent call last):
  File "12.py", line 2, in <module>
    str[0] = "h";
TypeError: 'str' object does not support item assignment
```

Deleting the String

As we know that strings are immutable. We cannot delete or remove the characters from the string. But we can delete the entire string using the **del** keyword.

```
str = "JAVATPOINT"
del str[1]
```

Output:

```
TypeError: 'str' object doesn't support item deletion
```

Now we are deleting entire string.

```
str1 = "JAVATPOINT"
del str1
print(str1)
```

Output:

```
NameError: name 'str1' is not defined
```

String Operators

Operator	Description
+	It is known as concatenation operator used to join the strings given either side of the operator.
*	It is known as repetition operator. It concatenates the multiple copies of the same string.
[]	It is known as slice operator. It is used to access the sub-strings of a particular string.
[:]	It is known as range slice operator. It is used to access the characters from the specified range.
in	It is known as membership operator. It returns if a particular sub-string is present in the specified string.
not in	It is also a membership operator and does the exact reverse of in. It returns true if a particular substring is not present in the specified string.
r/R	It is used to specify the raw string. Raw strings are used in the cases where we need to print the actual meaning of escape characters such as "C://python". To define any string as a raw string, the character r or R is followed by the string.

```
str = "Hello"
str1 = " world"
print(str*3) # prints HelloHelloHello
print(str+str1)# prints Hello world
print(str[4]) # prints o
print(str[2:4]); # prints ll
print('w' in str) # prints false as w is not present in str
print('wo' not in str1) # prints false as wo is present in str1.
print(r'C://python37') # prints C://python37 as it is written
```

Output:

```
HelloHelloHello
Hello world
o
ll
False
False
C://python37
```

Python String Length

In Python, we use the `len()` method to find the length of a string. For example,

```
greet = 'Hello'

# count length of greet string
print(len(greet))

# Output: 5
```

String Membership Test

We can test if a substring exists within a string or not, using the keyword `in`.

```
print('a' in 'program') # True
print('at' not in 'battle') False
```

Escape Sequences in Python

The escape sequence is used to escape some of the characters present inside a string.

Suppose we need to include both double quote and single quote inside a string,

```
example = "He said, "What's there?""

print(example) # throws error
```

Since strings are represented by single or double quotes, the compiler will treat `"He said, "` as the string. Hence, the above code will cause an error.

To solve this issue, we use the escape character `\` in Python.

```
# escape double quotes
example = "He said, \"What's there?\""

# escape single quotes
example = 'He said, "What\'s there?"'

print(example)

# Output: He said, "What's there?"
```

The list of an escape sequence is given below:

Sr.	Escape Sequence	Description	Example

1.	\newline	It ignores the new line.	`print("Python1 \`
			`Python2 \`
			`Python3")`
			
			Output:
			
			`Python1 Python2 Python3`

2.	\\	Backslash	`print("\\")`
			
			Output:
			
			`\`

3.	\'	Single Quotes	`print('\'')`
			
			Output:
			
			`'`

\''	Double Quotes	print("\"") **Output:**"

\b	ASCII Backspace(BS)	print("Hello \b World") **Output:** Hello World

\n	ASCII Linefeed	print("Hello \n World!") **Output:** Hello World!

\t	ASCII Horizontal Tab	print("Hello \t World!") **Output:** Hello World!