10. Python - String

Table of Contents

10. Python - String

1. Gentle reminder

- We already learnt first Hello world program in python.
- In that program we just print a group of characters by using print(p) function.
- That group of characters are called as a string.

Program Name printing Welcome to python programming demo1.py

```
print("Hello world")
```

Output

```
Hello world
```

2. What is a string?

2.1. Definition 1

- A group of characters enclosed within single or double or triple quotes is called as string.

2.2. Definition 2

- We can say string is a sequential collection of characters.

2.3. String is more popular

- In any kind of programming language, mostly usage data type is string.

3. Creating string

Syntax 1 With single quotes

```
name1 = 'Daniel'
```

Syntax 2 With double quotes

```
name2 = "Daniel"
```

Syntax 3 With triple single quotes

```
name3 = '''Daniel'''
```

Syntax 4 With triple double quotes

```
name4 = """Daniel"""
```

Program Name

Creating string by using all possibilities
demo2.py

```
name1= 'Daniel'
name2 = "Prasad"
name3 = '''Mouli'''
name4 = """Veeru"""

print(name1, "name is created by using single quotes")
print(name2, "name is created by using double quotes")
print(name3, "name is created by using triple single quotes")
print(name4, "name is created by using triple double quotes")
```

output

```
Daniel name is created by using single quotes
Prasad name is created by using double quotes
Mouli name is created by using triple single quotes
Veeru name is created by using triple double quotes
```

Make a note

- ✓ Generally, to create a string mostly used syntax is double quotes syntax.

3.1. When should we go for triple single and triple double quotes?

- ✓ If you want to create multiple lines of string, then triple single or triple double quotes are the best to use.

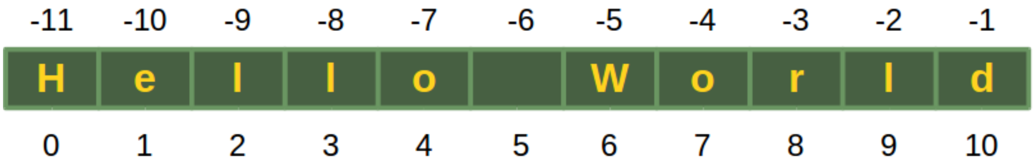
Program Name printing Employee information
demo3.py

```
loc1 = '''TCS company
          White Field
          Bangalore'''

loc2 = """TCS company
          Bangalore
          ITPL tech park"""

print(loc1)
print(loc2)
```

output

```
TCS company
White Field
Bangalore

TCS company
Bangalore
ITPL  tech park
```

Diagram representation



-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

Program Name

Accessing string by using index
demo4.py

```
wish = "Hello World"

print(wish[0])
print(wish[1])
```

Output

```
H
e
```

Program Name

Accessing string by using slicing
demo5.py

```
wish = "Hello World"

print(wish[0:7])
```

Output

```
Hello W
```

Program Name

Accessing string by using for loop
demo6.py

```
wish = "Hello World"

for char in wish:
        print(char)
```

Output

```
H
e
l
l
o

W
o
r
l
d
```

4. Mutable

- ✓ Once if we create an object then the state of existing object can be change/modify/update.
- ✓ This behaviour is called as mutability.

5. Immutable

- ✓ Once if we create an object then the state of existing object cannot be change/modify/update.
- ✓ This behaviour is called as immutability.

6. Strings are immutable

- ✓ String having immutable nature.
- ✓ Once we create a string object then we cannot change or modify the existing object.

Program Name: Printing name and first index in string
demo7.py

```
name = "Daniel"
print(name)
print(name[0])
```

output

```
Daniel
D
```

Program Name string having immutable nature
demo8.py

```
name = "Daniel"

print(name)
print(name[0])
name[0]="X"
```

output

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

7. Mathematical operators on string objects

- ✓ We can perform two mathematical operators on string.
- ✓ Those operators are,
    - ○ Addition (+) operator.
    - ○ Multiplication (*) operator.

7.1. Addition (+) operator with string

- ✓ The + operator works like concatenation or joins the strings.

Program — + works as concatenation operator

Name — demo9.py

```
a = "Python"
b = "Programming"
print(a+b)
```

output

```
PythonProgramming
```

7.2. Multiplication (*) operator with string

- ✓ This operator works with string to do repetition.

Program — * operator works as repetition in strings

Name — demo10.py

```
course="Python"
print(course*3)
```

output

```
PythonPythonPython
```

8. Length of the string

- ✓ We can find number of characters in string by using len() function

Program Name

length of the string
demo11.py

```
course = "Python"
print(len(course))
```

Output

```
6
```

9. Membership operators (in, not in)

Definition 1

- ✓ We can check, if a string or character is a member of string or not by using in and not in operators

Definition 2

- ✓ We can check, if string is a substring of main string or not by using in and not in operators.

9.1. in operator

- ✓ in operator returns True, if the string or character found in the main string.

Program Name in operator demo12.py

```
print('p' in 'python')
print('z' in 'python')
print('on' in 'python')
print('pa' in 'python')
```

output

```
True
False
True
False
```

9.2. not in operator

- ✓ The not in operator returns opposite result of in operator.
- ✓ not in operator returns True, if the string or character not found in the main string.

Program
Name

not in operator
demo13.py

```
print('b' not in 'apple')
```

output

```
True
```

10. Methods in str class

- ✓ As discussed, str is a predefined class.
- ✓ So, str class can contain methods because methods can be created inside of class only.
- ✓ We can check these methods by using dir(parameter1) predefined function.

- ✓ So, internally str class contains two types of methods,
  - ○ With underscore symbol methods.
    - ▪ We no need to focus
  - ○ Without underscore symbol methods.
    - ▪ We need to focus much on these

Program Name

Printing str class methods by using dir(str) function
demo14.py

```
print(dir(str))
```

output

```
[

__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__',
'__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', 'capitalize',
```

Important methods

'count', 'upper', 'lower', 'replace', 'split', 'strip',

]

Important point

- ✓ As per object-oriented principle,
    - If we want to access instance methods, then we should access by using object name.
- ✓ So, all str class methods we can access by using str object

Important methods in str class

- ✓ upper()
- ✓ lower()
- ✓ strip()
- ✓ count(p)
- ✓ replace(p1, p2)
- ✓ split(p)

11.1. upper() method

- ✓ upper() is a pre-defined method in str class
- ✓ This method we should access by using string object.
- ✓ This method converts lower case letters into upper case letters

Program Name

Converting from lowercase to uppercase
demo15.py

```
name = "daniel"
print("Before converting: ", name)
print("After converting: ", name.upper())
```

Output

```
Before converting:  daniel
After converting:  DANIEL
```

11.2. lower() method

- ✓ lower() is a pre-defined method in str class
- ✓ This method we should access by using string object.
- ✓ This method converts upper case letters into lower case letters

Program Converting from uppercase to lowercase

Name demo16.py

```
name = "DANIEL"
print("Before converting: ", name)
print("After converting: ", name.lower())
```

Output

```
Before converting:  DANIEL
After converting:  daniel
```

11.3. strip() method

- ✓ strip() is a pre-defined method in str class
- ✓ This method we should access by using string object.
- ✓ This method removes left and right side spaces of string

Make a note

- ✓ This method will not remove the spaces which are available middle of string object.

Program Name

removing spaces in starting and ending of the string
demo17.py

```
course = "Python            "
print("with spaces course length is: ",len(course))
x = course.strip()
print("after removing spaces, course length is: ",len(x))
```

Output

```
with spaces course length is:  18
after removing spaces, course length is:  6
```

11.4. count(p) method

- ✓ count(p) is a pre-defined method in str class
- ✓ This method we should access by using string object.
- ✓ By using count() method we can find the number of occurrences of substring present in the string

Program Name counting sub string by using count() method
demo18.py

```
s="Python programming language, Python is easy"
print(s.count("Python"))
print(s.count("Hello"))
```

output

```
2
0
```

11.5. replace(p1, p2) method

- ✓ replace(p1, p2) is a pre-defined method in str class
- ✓ This method we should access by using string object.
- ✓ We can replace old string with new string by using replace(p1, p2) method.

Program replacing string by using replace method
Name demo19.py

```
s1 = "Java programming language"
s2 = s1.replace("Java", "Python")

print(s1)
print(s2)
```

output

```
Java programming language
Python programming language
```

Replace method returns new string object

- ✓ As we know string is immutable, so replace(p1, p2) method never perform changes on the existing string object.
- ✓ replace(p1, p2) method creates new string object, we can check this by using id(p) function

Program Name
replacing string by using replace(p1, p2) method
demo20.py

```
s1 = "Java programming language"
s2 = s1.replace("Java", "Python")

print(s1)
print(s2)

print(id(s1))
print(id(s2))
```

output

```
Java programming language
Python programming language
49044256
48674600
```

String objects are immutable, Is replace(p1, p2) method will modify the string objects?

- ✓ Once we create a string object, we cannot change or modify the existing string object.
- ✓ This behaviour is called as immutability.
- ✓ If we are trying to change or modify the existing string object, then with those changes a new string object will be created.
- ✓ So, replace(p1, p2) method will create new string object with the modifications.

Splitting of Strings:

11.6. split(p) method

- ✓ split(p) is a pre-defined method in str class
- ✓ This method we should access by using string object.
- ✓ The default separator is space.
- ✓ We can split the given string according to specified separator by using split(p) method.
- ✓ split(p) method returns list.

Program Name splitting string by using split() method
demo21.py

```
s = "Python programming language"
n = s.split()

print("Before splitting:", s)
print("After splitting: ", n)
```

output

```
Before splitting: Python programming language
After splitting: ['Python', 'programming', 'language']
```

Program splitting string by using split(p) method

Name demo22.py

```
s = "This is, Python programming, language "
n = s.split(",")

print("Before splitting:", s)
print("After splitting: ", n)
```

output

```
Before splitting: This is, Python programming, language
After splitting:  ['This is', ' Python programming', ' language']
```