



## Python

### Experiment No. 4

**Aim:** To Study various string operations and methods in python.

#### Problem Statements:

1. Write a program to output middle three characters of an input string.
2. Given two strings, 'Rama' and 'Sita'. Write a program to create a new string by appending 'Sita' in the middle of the string "Rama".
3. Arrange string characters such that lowercase letters should come first.
4. Write a program to count occurrences of all characters within a string.
5. Count all letters, digits, and special symbols from a given string
6. Write a program to find the last position of a substring "Rama" in a given string
7. Calculate the sum and average of the digits present in a string
8. Removal all characters from a string except integers
9. Replace each special symbol with # in the following string:

#### Theory :

What is a String in Python?

A String is a data structure in Python that represents a sequence of characters. It is an immutable data type, meaning that once you have created a string, you cannot change it. Strings are used widely in many different applications, such as storing and manipulating text data, representing names, addresses, and other types of data that can be represented as text.

Python does not have a character data type, a single character is simply a string with a length of 1 (one).

#### **Creating a String in Python**

Strings in Python can be created using single quotes or double quotes or even triple quotes. Let us see how we can define a string in Python.

#### **Example:**

In this example, we will demonstrate different ways to create a Python String. We will create a string using single quotes (' '), double quotes (" "), and triple double quotes (""" """). The triple quotes can be used to declare multiline strings in Python.

Code:

```

String1 = 'Welcome to the Geeks World'

print("String with the use of Single Quotes: ")
print(String1)

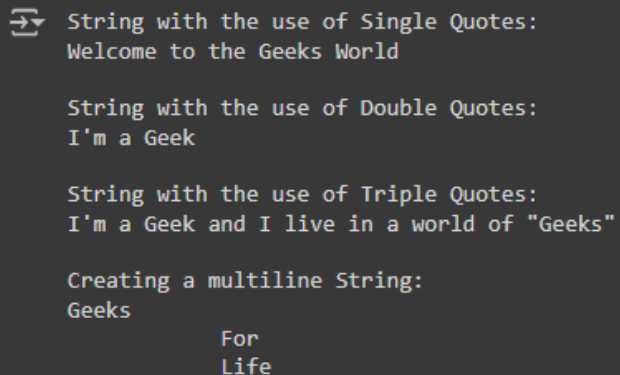
# Creating a String
# with double Quotes
String1 = "I'm a Geek"
print("\nString with the use of Double Quotes: ")
print(String1)

# Creating a String
# with triple Quotes
String1 = '''I'm a Geek and I live in a world of
"Geeks"'''
print("\nString with the use of Triple Quotes: ")
print(String1)

# Creating String with triple
# Quotes allows multiple lines
String1 = '''Geeks
              For
              Life'''
print("\nCreating a multiline String: ")
print(String1)

```

Output:



```

String with the use of Single Quotes:
Welcome to the Geeks World

String with the use of Double Quotes:
I'm a Geek

String with the use of Triple Quotes:
I'm a Geek and I live in a world of "Geeks"

Creating a multiline String:
Geeks
    For
    Life

```

## Accessing characters in Python String

In Python, individual characters of a String can be accessed by using the method of Indexing. Indexing allows negative address references to access characters from the back of the String, e.g. -1 refers to the last character, -2 refers to the second last character, and so on

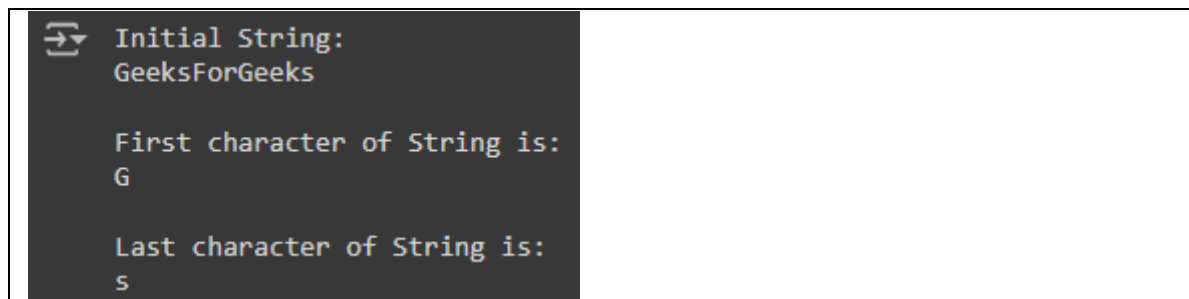
Code:

```
String1 = "GeeksForGeeks"
print("Initial String: ")
print(String1)

# Printing First character
print("\nFirst character of String is: ")
print(String1[0])

# Printing Last character
print("\nLast character of String is: ")
print(String1[-1])
```

Output:



```
⇒ Initial String:
   GeeksForGeeks

   First character of String is:
   G

   Last character of String is:
   s
```

## Join Two or More Strings

```
message1 = "Hello, "
message2 = "students"

# using + operator
result = message1 + message2

print(result)
```

## String Slicing

In Python, the String Slicing method is used to access a range of characters in the String. Slicing in a String is done by using a Slicing operator, i.e., a colon (:). One thing to keep in mind while using this method is that the string returned after slicing includes the character at the start index but not the character at the last index.

### **Example:**

In this example, we will use the string-slicing method to extract a substring of the original string. The [3:12] indicates that the string slicing will start from the 3rd index of the string to the 12th index, (12th character not including). We can also use negative indexing in string slicing.

### **Accessing characters in Python String**

```
String1 = "GeeksForGeeks"

print("Initial String: ")

print(String1)

# Printing First character

print("\nFirst character of String is: ")

print(String1[0])

# Printing Last character

print("\nLast character of String is: ")

print(String1[-1])
```

### **String Slicing**

In Python, the String Slicing method is used to access a range of characters in the String. Slicing in a String is done by using a Slicing operator, i.e., a colon (:). One thing to keep in mind while using this method is that the string returned after slicing includes the character at the start index but not the character at the last index.

Python slicing can be done in two ways:

- Using a slice() method
- Using the array slicing [:: ] method

Code:

```
# String slicing
String = 'ASTRING'
# Using slice constructor
s1 = slice(3)
s2 = slice(1, 5, 2)
s3 = slice(-1, -12, -2)
print("String slicing")
print(String[s1])
print(String[s2])
print(String[s3])
```

Output:

```
String slicing
AST
SR
GITA
```

### Example:

In this example, we will use the string-slicing method to extract a substring of the original string. The [3:12] indicates that the string slicing will start from the 3rd index of the string to the 12th index, (12th character not including). We can also use negative indexing in string slicing.

Code:

```
String1 = "GeeksForGeeks"
print("Initial String: ")
print(String1)
# Printing 3rd to 12th character
print("\nSlicing characters from 3-12: ")
print(String1[3:12])
# Printing characters between
# 3rd and 2nd last character
print("\nSlicing characters between " +
      "3rd and 2nd last character: ")
print(String1[3:-2])
```

Output:

---

- Chaitanya Shah



Initial String:  
GeeksForGeeks

Slicing characters from 3-12:  
ksForGeek

Slicing characters between 3rd and 2nd last character:  
ksForGee

## Reversing a Python String

By accessing characters from a string, we can also reverse strings in Python. We can Reverse a string by using String slicing method.

### Example:

In this example, we will reverse a string by accessing the index. We did not specify the first two parts of the slice indicating that we are considering the whole string, from the start index to the last index.

```
gfg = "geeksforgeeks"  
print(gfg[::-1])
```

## Python String Operations

There are many operations that can be performed with strings which makes it one of the most used data types in Python.

### 1. Compare Two Strings

We use the == operator to compare two strings. If two strings are equal, the operator returns True. Otherwise, it returns False. For example,

```
str1 = "Hello, world!"  
str2 = "I love Python."  
str3 = "Hello, world!"  
  
# compare str1 and str2  
print(str1 == str2)  
  
# compare str1 and str3  
print(str1 == str3)
```

## 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
```

### Format() Method:

we cannot combine strings and numbers. If we want to combine, we can use the format() method!

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders { } are:

```
age = 36

txt = "My name is John, and I am {}"

print(txt.format(age))
```

### Methods of Python String

Besides those mentioned above, there are various [string methods](#) present in Python. Here are some of those methods:

Methods	Description
<a href="#">upper()</a>	converts the string to uppercase
<a href="#">lower()</a>	converts the string to lowercase
<a href="#">partition()</a>	returns a tuple
<a href="#">replace()</a>	replaces substring inside
<a href="#">find()</a>	returns the index of first occurrence of substring
<a href="#">rstrip()</a>	removes trailing characters
<a href="#">split()</a>	splits string from left
<a href="#">startswith()</a>	checks if string starts with the specified string

<a href="#">isnumeric()</a>	checks numeric characters
<a href="#">index()</a>	returns index of substring

```
a="Hello, World!"
```

```
print(a.upper())
```

```
a = "Hello, World!"
```

```
print(a.lower())
```

```
a = "Hello, World!"
```

```
print(a.replace("H", "J"))
```

### 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?"
```

**Here is a list of all the escape sequences supported by Python.**



Escape Sequence	Description
<code>\\</code>	Backslash
<code>\'</code>	Single quote
<code>\"</code>	Double quote
<code>\a</code>	ASCII Bell
<code>\b</code>	ASCII Backspace
<code>\f</code>	ASCII Formfeed
<code>\n</code>	ASCII Linefeed
<code>\r</code>	ASCII Carriage Return
<code>\t</code>	ASCII Horizontal Tab
<code>\v</code>	ASCII Vertical Tab
<code>\ooo</code>	Character with octal value ooo
<code>\xHH</code>	Character with hexadecimal value HH

Q.1 Write a program to output middle three characters of an input string.

Code:

```
import math

str=input("Enter a string : ")
mid=len(str)/2
print(str)
if (mid%3 != 0):
    mid=math.floor(len(str)/2)
    print(str[mid-1:mid+2])
else:
    print("\nString is of Even length....Three Middle characters not possible")
```

Output:


```
➞ Enter a string : madam
madam
ada
```

Q.2 Given two strings, 'Rama' and 'Sita'. Write a program to create a new string by appending 'Sita' in the middle of the string "Rama".

Code:

```
str1='Rama'  
str2='Sita'  
  
mid=int(len(str1)/2)  
str=str1[:mid]+str2+str1[mid:]  
print(str)
```

Output:

 RaSitama

Q.3 Arrange string characters such that lowercase letters should come first.

Code:

```
str=input("Enter a String : ")
str_list=list(str)
l=int(len(str))

lower=''
upper=''

i=0
while (l!=0):
    if (str_list[i].islower()):
        lower+=str_list[i]
    else:
        upper+=str_list[i]
    i+=1
    l-=1

print("\nArranged String is :\n",lower+upper)
```

Output:

```
Enter a String : aBcDeFgHiJkLmNoPqRsTuVwXyZ

Arranged String is :
acegikmoqsuwyBDFHJLNPRTVXZ
```

Q.4 Write a program to count occurrences of all characters within a string.

Code:

```
str=input("Enter a string : ")

char_count = {}

for char in str:
    if char in char_count:
        char_count[char] += 1
    else:
        char_count[char] = 1
print()
for key, value in char_count.items():
    print("{} appears {} times".format(key,value))
```

Output:

```
➡ Enter a string : Mississippi

M appears 1 times
i appears 4 times
s appears 4 times
p appears 2 times
```

Q.5 Count all letters, digits, and special symbols from a given string

Code:

```
str=input("Enter a string : ")

upper=lower=num=spc=0

for char in str:
    if char.isupper():
        upper+=1
    elif char.islower():
        lower+=1
    elif char.isdigit():
        num+=1
    else:
        spc+=1

print("\nUppercase : {}\nLowercase : {}\nDigits : {}\nSpecial\nCharacter : {}".format(upper,lower,num,spc))
```

Output:

```
➞ Enter a string : A_xyz123@email.com

Uppercase : 1
Lowercase : 11
Digits : 3
Special Character : 3
```

Q.6 Write a program to find the last position of a substring "Rama" in a given string

Code:

```
string = input("Enter a string: ")
substring = "Rama"

last_position = -1
index = 0

while index < len(string):
    if string[index: index + len(substring)] == substring:
        last_position = index
        index += 1

if last_position != -1:
    print("Last position of substring is:", last_position)
else:
    print("Substring not found.")
```

Output:

```
➞ Enter a string: Rama is Rama
   Last position of substring is: 8
```

Q.7 Calculate the sum and average of the digits present in a string

Code:

```
n=int(input("Enter a number : "))

sum=count=0

while n!=0:
    sum+=n%10
    n=int(n/10)
    count+=1

print("\nSum : {}\nAverage : {}".format(sum,sum/count))
```

Output:

```
➞ Enter a number : 164582

Sum : 26
Average : 4.333333333333333
```



### Q.8 Removal all characters from a string except integers

Code:

```
str=input("Enter a string : ")

numStr=''
for char in str:
    if char.isdigit():
        numStr+=char

print("\nString after removing all characters except integers : 
"+numStr)
```

Output:

```
Enter a string : Python3

String after removing all characters except integers : 3
```

Q.9 Replace each special symbol with # in the following string:

Code:

```
str=input("Enter a string : ")

for char in str:
    if not char.isalnum():
        str=str.replace(char,'#')

print("\nString after replacing each special symbol with # : "+str)
```

Output:

```
⇒ Enter a string : a@b$c%d^e12&34
String after replacing each special symbol with # : a#b#c#d#e12#34
```

**Conclusion:** Thus studied data types in python.