Unit II

CHAPTER 8

STRINGS AND STRING MANIPULATION

Learning Objectives

After completion of this chapter, the student will be able to

- Know how to process text.
- Understanding various string functions in Python.
- Know how to format Strings.
- Know about String Slicing.
- Know about Strings application in real world.

EGT61Y

8.1 Introduction

String is a data type in python, which is used to handle array of characters. String is a sequence of Unicode characters that may be a combination of letters, numbers, or special symbols enclosed within single, double or even triple quotes.

Example

'Welcome to learning Python'

"Welcome to learning Python"

''' "Welcome to learning Python" '''

In python, strings are immutable, it means, once you define a string, it cannot be changed during execution.

8.2 Creating Strings

As we learnt already, a string in Python can be created using single or double or even triple quotes. String in single quotes cannot hold any other single quoted character in it, because the compiler will not recognize where to start and end the string. To overcome this problem, you have to use double quotes. Strings which contains double quotes should be define within triple quotes. Defining strings within triple quotes also allows creation of ***multiline*** strings.

Example

```
#A string defined within single quotes
>>> print (‘Greater Chennai Corporation’)
        Greater Chennai Corporation

#single quoted string defined within single quotes
>>> print ('Greater Chennai Corporation's student')
        SyntaxError: invalid syntax

#A string defined within double quotes
>>>print (“Computer Science”)
        Computer Science

#double quoted string defined within double quotes
>>> print (''' "Computer Science" ''')
        "Computer Science"

#single and double quoted multiline string defined within triple quotes
>>> print (''' "Strings are immutable in 'Python',
                which means you can't make any changes
                once you declared" ''')

        "Strings are immutable in 'Python',
        which means you can't make any changes once you declared"
```

8.3 Accessing characters in a String

Once you define a string, python allocate an index value for its each character. These index values are otherwise called as subscript which are used to access and manipulate the strings. The subscript can be positive or negative integer numbers.

The positive subscript **0** is assigned to the first character and **n-1** to the last character, where n is the number of characters in the string. The negative index assigned from the last character to the first character in reverse order begins with **-1**.

Example

String	S	C	H	O	O	L
Positive subscript	0	1	2	3	4	5
Negative subscript	-6	-5	-4	-3	-2	-1

Example 1 : Program to access each character with its positive subscript of a giving string

```
str1 = input ("Enter a string: ")
index=0
for i in str1:
        print ("Subscript[",index,"] : ", i)
        index + = 1
```

Output

```
        Enter a string: welcome
        Subscript [ 0 ] :  w
        Subscript [ 1 ] :  e
        Subscript [ 2 ] :  l
        Subscript [ 3 ] :  c
        Subscript [ 4 ] :  o
        Subscript [ 5 ] :  m
        Subscript [ 6 ] :  e
```

Example 2 : Program to access each character with its negative subscript of a giving string

```
str1 = input ("Enter a string: ")
index=-1
while index >= -(len(str1)):
        print ("Subscript[",index,"] : " + str1[index])
        index += -1
```

Output

```
        Enter a string: welcome
        Subscript [ -1 ] : e
        Subscript [ -2 ] : m
        Subscript [ -3 ] : o
        Subscript [ -4 ] : c
        Subscript [ -5 ] : l
        Subscript [ -6 ] : e
        Subscript [ -7 ] : w
```

8.4 Modifying and Deleting Strings

As you already learnt, strings in python are immutable. That means, once you define a string modifications or deletion is not allowed. If you want to modify the string, a new string value can be assign to the existing string variable.

Example

```
>>> str1="How are you"
>>> str1[0]="A"
Traceback (most recent call last):
            File "<pyshell#1>", line 1, in <module>
            str1[0]="A"
      TypeError: 'str' object does not support item assignment
```

In the above example, string variable str1 has been assigned with the string "How are you" in statement 1. In the next statement, we try to update the first character of the string with character 'A'. But python will not allow the update and it shows a TypeError.

To overcome this problem, you can define a new string value to the existing string variable. Python completely overwrite new string on the existing string.

Example

```
>>> str1="How are you"
>>> print (str1)
      How are you
>>> str1="How about you"
>>> print (str1)
      How about you
```

Usually python does not support any modification in its strings. But, it provides a function replace() to change all occurrences of a particular character in a string.

General formate of replace function:

replace("char1", "char2")

The replace function replaces all occurrences of char1 with char2.

Example

```
>>> str1="How are you"
>>> print (str1)
      How are you
>>> print (str1.replace("o", "e"))
      Hew are yeu
```

Similar as modification, python will not allow deleting a particular character in a string. Whereas you can remove entire string variable using **del** command.

Example 3: Code lines to delete a particular character in a string:

```
>>> str1="How are you"
>>> del str1[2]
Traceback (most recent call last):
        File "<pyshell#7>", line 1, in <module>
                del str1[2]
        TypeError: 'str' object doesn't support item deletion
```

Example 4: Code lines to delete a string variable

```
>>> str1="How about you"
>>> print (str1)
        How about you
>>> del str1
>>> print (str1)
Traceback (most recent call last):
        File "<pyshell#14>", line 1, in <module>
                print (str1)
        NameError: name 'str1' is not defined
```

8.5 String Operators

Python provides the following operators for string operations. These operators are useful to manipulate string.

(i) Concatenation (+)

Joining of two or more strings is called as Concatenation. The plus (+) operator is used to concatenate strings in python.

Example

```
>>> "welcome" + "Python"
        'welcomePython'
```

(ii) Append (+ =)

Adding more strings at the end of an existing string is known as append. The operator += is used to append a new string with an existing string.

Example

```
>>> str1="Welcome to "
>>> str1+="Learn Python"
>>> print (str1)
        Welcome to Learn Python
```

(iii) Repeating (*)

The multiplication operator (*) is used to display a string in multiple number of times.

Example

```
>>> str1="Welcome "
>>> print (str1*4)
        Welcome Welcome Welcome Welcome
```

(iv) String slicing

Slice is a substring of a main string. A substring can be taken from the original string by using [ ] operator and index or subscript values. Thus, [ ] is also known as slicing operator. Using slice operator, you have to slice one or more substrings from a main string.

General format of slice operation:

str[start:end]

Where ***start*** is the beginning index and ***end*** is the last index value of a character in the string. Python takes the end value less than one from the actual index specified. For example, if you want to slice first 4 characters from a string, you have to specify it as 0 to 5. Because, python consider only the end value as n-1.

Example I : slice a single character from a string

```
>>> str1="THIRUKKURAL"
>>> print (str1[0])
        T
```

Example II : slice a substring from index 0 to 4

```
>>> print (str1[0:5])
        THIRU
```

Example III : slice a substring using index 0 to 4 but without specifying the beginning index.

```
>>> print (str1[:5])
        THIRU
```

Example IV : slice a substring using index 0 to 4 but without specifying the end index.

```
>>> print (str1[6:])
        KURAL
```

Example V : Program to slice substrings using for loop

```
str1="COMPUTER"
index=0
for i in str1:
        print (str1[:index+1])
        index+=1
```

Output

```
        C
        C O
        C O M
        C O M P
        C O M P U
        C O M P U T
        C O M P U T E
        C O M P U T E R
```

(v) Stride when slicing string

When the slicing operation, you can specify a third argument as the stride, which refers to the number of characters to move forward after the first character is retrieved from the string. The default value of stride is 1.

Example

```
>>> str1 = "Welcome to learn Python"
>>> print (str1[10:16])
        learn
>>> print (str1[10:16:4])
        r
>>> print (str1[10:16:2])
        er
>>> print (str1[::3])
Wceoenyo
```

Note: Remember that, python takes the last value as n-1

You can also use negative value as stride (third argument). If you specify a negative value, it prints in reverse order.

Example

```
>>> str1 = "Welcome to learn Python"
>>> print(str1[::-2])
        nhy re teolW
```

8.6 String Formatting Operators

The string formatting operator is one of the most exciting feature of python. The formatting operator % is used to construct strings, replacing parts of the strings with the data stored in variables.

Syntax:
("String to be display with %val1 and %val2" %(val1, val2))

Example

```
name = "Rajarajan"
mark = 98
print ("Name: %s and Marks: %d" %(name,mark))
```

Output

```
Name: Rajarajan and Marks: 98
```

8.7 Formatting characters

Format characters	USAGE
%c	Character
%d (or) %i	Signed decimal integer
%s	String
%u	Unsigned decimal integer
%o	Octal integer
%x or %X	Hexadecimal integer (lower case x refers a-f; upper case X refers A-F)
%e or %E	Exponential notation
%f	Floating point numbers
%g or %G	Short numbers in floating point or exponential notation.

Escape sequence in python

Escape sequences starts with a backslash and it can be interpreted differently. When you have use single quote to represent a string, all the single quotes inside the string must be escaped. Similar is the case with double quotes.

Example

```
# String within triple quotes to display a string with single quote
>>> print ('''They said, "What's there?"''')
        They said, "What's there?"

# String within single quotes to display a string with single quote using escape sequence
>>> print ('They said, "What\'s there?"')
        They said, "What's there?"

# String within double quotes to display a string with single quote using escape sequence
>>> print ("They said, \"What's there?\"")
        He said, "What's there?"
```

Escape sequences supported by python

Escape Sequence	DESCRIPTION
\newline	Backslash and newline ignored
\\	Backslash
\'	Single quote
\"	Double quote
\a	ASCII Bell
\b	ASCII Backspace
\f	ASCII Form feed
\n	ASCII Linefeed
\r	ASCII Carriage Return
\t	ASCII Horizontal Tab
\v	ASCII Vertical Tab
\ooo	Character with octal value ooo
\xHH	Character with hexadecimal value HH

8.8 The format( ) function

The format( ) function used with strings is very versatile and powerful function used for formatting strings. The curly braces { } are used as placeholders or replacement fields which get replaced along with format( ) function.

Example

```
num1=int (input("Number 1: "))
num2=int (input("Number 2: "))
print ("The sum of { } and { } is { }".format(num1, num2,(num1+num2)))
```

Out Put

```
Number 1: 34
Number 2: 54
The sum of 34 and 54 is 88
```

8.9 Built-in String functions

Python supports the following built-in functions to manipulate string.

Syntax	Description	Example

len(str)	Returns the length (no of characters) of the string.	>>> A="Corporation"
		>>> print(len(A))
		11

capitalize( )	Used to capitalize the first character of the string	>>> city="chennai"
		>>> print(city.capitalize())
		Chennai

center(width, fillchar)	Returns a string with the original string centered to a total of width columns and filled with fillchar in columns that do not have characters	>>> str1="Welcome"
		>>> print(str1.center(15,'*'))
		********Welcome*********

find(sub[, start[, end]])	The function is used to search the first occurrence of the sub string in the given string. It returns the index at which the substring starts. It returns -1 if the substring does not occur in the string.	>>>str1='mammals' >>>str1.find('ma') ***0*** ***On omitting the start parameters, the function starts the search from the beginning.*** >>>str1.find('ma',2) 3 >>>str1.find('ma',2,4) -1 ***Displays -1 because the substring could not be found between the index 2 and 4-1.*** >>>str1.find('ma',2,5) 3

Syntax	Description	Example
isalnum( )	Returns True if the string contains only letters and digit. It returns False. If the string contains any special character like _, @, #, *, etc.	>>>str1='Save Earth' >>>str1.isalnum() ***False*** The function returns False as space is an alphanumeric character. >>>'Save1Earth'.isalnum() ***True***
isalpha( )	Returns True if the string contains only letters. Otherwise return False.	>>>'Click123'.isalpha() ***False*** >>>'python'.isalpha( ) ***True***
isdigit( )	Returns True if the string contains only numbers. Otherwise it returns False.	>>> str1='Save Earth' >>>print(str1.isdigit( )) ***False***
lower( )	Returns the exact copy of the string with all the letters in lowercase.	>>>str1='SAVE EARTH' >>>print(str1.lower()) ***save earth***
islower( )	Returns True if the string is in lowercase.	>>> str1='welcome' >>>print (str1.islower( )) ***True***
isupper( )	Returns True if the string is in uppercase.	>>> str1='welcome' >>>print (str1.isupper( )) ***False***
upper( )	Returns the exact copy of the string with all letters in uppercase.	>>> str1='welcome' >>>print (str.upper( )) ***WELCOME***

title( )	Returns a string in title case	>>> str1='education department'
		>>> print(str1.title())
		Education Department

swapcase( )	It will change case of every character to its opposite case vice-versa.	>>> str1="tAmiL NaDu"
		>>> print(str1.swapcase())
		TaMIl nAdU

Syntax	Description	Example

count(str, beg, end)	Returns the number of substrings occurs within the given range. Remember that substring may be a single character. Range (beg and end) arguments are optional. If it is not given, python searched in whole string. Search is case sensitive.	>>> str1="Raja Raja Chozhan"
		>>> print(str1.count('Raja'))
		2
		>>> print(str1.count('r'))
		0
		>>> prin***t(str1.count('R'))***
		2
		>>> print(str1.count('a'))
		5
		>>> print(str1.count('a',0,5))
		2
		>>> print(str1.count('a',11))
		1

ord(char)	Returns the ASCII code of the character.	>>> ch = 'A'
		>>> print(ord(ch))
		65
		>>> print(***ord('B'))***
		66

chr(ASII)	Returns the character represented by a ASCII.	>>> ch=97
		>>> print(chr(ch))
		a
		>>> print(chr(87))
		W

8.10 Membership Operators

The '**in**' and '**not in**' operators can be used with strings to determine whether a string is present in another string. Therefore, these operators are called as Membership Operators.

Example

```
str1=input ("Enter a string: ")
str2="chennai"
if str2 in str1:
        print ("Found")
else:
        print ("Not Found")
```

Output : 1

```
Enter a string: Chennai G HSS, Saidapet
Found
```

Output : 2

```
Enter a string: Govt G HSS, Ashok Nagar
Not Found
```

Example 8.11 Programs using Strings :

Example 8.11.1 : Program to check whether the given string is palindrome or not

```
str1 = input ("Enter a string: ")
str2 = ' '
index=-1
for i in str1:
          str2 += str1[index]
          index -= 1
print ("The given string = { } \n The Reversed string = { }".format(str1, str2))
if (str1==str2):
          print ("Hence, the given string is Palindrome")
else:
          print ("Hence, the given is not a palindrome")
```

Output : 1

```
Enter a string: malayalam
The given string = malayalam
The Reversed string = malayalam
Hence, the given string is Palindrome
```

Output : 2

```
Enter a string: welcome
The given string = welcome
The Reversed string = emoclew
Hence, the given string is not a palindrome
```

Example 8.11.2 : Program to display the following pattern

```
*
* *
* * *
* * * *
* * * * *
str1=' * '
i=1
while i<=5:
          print (str1*i)
          i+=1
```

Output

```
*
* *
* * *
* * * *
* * * * *
```

Example 8.11.3 : Program to display the number of vowels and consonants in the given string

```
str1=input ("Enter a string: ")
str2="aAeEiIoOuU"
v,c=0,0
for i in str1:
        if i in str2:
                v+=1
        else:
                c+=1
print ("The given string contains { } vowels and { } consonants".format(v,c))
```

Output

```
        Enter a string: Tamilnadu School Education
        The given string contains 11 vowels and 15 consonants
```

Example 8.11.4 : Program to create an Abecedarian series. (Abecedarian refers list of elements appear in alphabetical order)

```
str1="ABCDEFGH"
str2="ate"
for i in str1:
        print ((i+str2),end='\t')
```

Output

```
        Aate    Bate    Cate    Date    Eate    Fate    Gate    Hate
```

Example 8.11.5 : Program that accept a string from the user and display the same after removing vowels from it

```
def rem_vowels(s):
        temp_str=''
        for i in s:
                if i in "aAeEiIoOuU":
                        pass
                else:
                        temp_str+=i
        print ("The string without vowels: ", temp_str)
str1= input ("Enter a String: ")
rem_vowels (str1)
```

Output

```
        Enter a String: Mathematical fundations of Computer Science
        The string without vowels:  Mthmtcl fndtns f Cmptr Scnc
```

Example 8.11.6 : Program that count the occurrences of a character in a string

```
def count(s, c):
        c1=0
        for i in s:
                if i == c:
                        c1+=1
        return c1
str1=input ("Enter a String: ")
ch=input ("Enter a character to be searched: ")
cnt=count (str1, ch)
print ("The given character {} is occurs {} times in the given string".format(ch,cnt))
```

Out Put

```
        Enter a String: Software Engineering
        Enter a character to be searched: e
        The given character e is occurs 3 times in the given string
```

Points to remember:

- String is a data type in python.
- Strings are immutable, that means once you define string, it cannot be changed during execution.
- Defining strings within triple quotes also allows creation of multiline strings.
- In a String, python allocate an index value for its each character which is known as subscript.
- The subscript can be positive or negative integer numbers.
- Slice is a substring of a main string.
- Stride is a third argument in slicing operation.
- Escape sequences starts with a backslash and it can be interpreted differently.
- The format( ) function used with strings is very versatile and powerful function used for formatting strings.
- The ‘in’ and ‘not in’ operators can be used with strings to determine whether a string is present in another string.

Hands on Experience

1. Write a python program to find the length of a string.
2. Write a program to count the occurrences of each word in a given string.

3. Write a program to add a prefix text to all the lines in a string.
4. Write a program to print integers with '*' on the right of specified width.
5. Write a program to create a mirror image of the given string. For example, "wel" = "lew".
6. Write a program to removes all the occurrences of a give character in a string.
7. Write a program to append a string to another string without using += operator.
8. Write a program to swap two strings.
9. Write a program to replace a string with another string without using replace().
10. Write a program to count the number of characters, words and lines in a given string.

EWC13A

Evaluation

Part - I

Choose the best answer **(1 Mark)**

1. Which of the following is the output of the following python code?

   str1="TamilNadu"
   print(str1[::-1])

   (a) Tamilnadu (b) Tmlau
   (c) udanlimaT d) udaNlimaT

2. What will be the output of the following code?

   str1 = "Chennai Schools"
   str1[7] = "-"

   (a) Chennai-Schools (b) Chenna-School
   (c) Type error (D) Chennai

3. Which of the following operator is used for concatenation?

   (a) + (b) & (c) * d) =

4. Defining strings within triple quotes allows creating:

   (a) Single line Strings (b) Multiline Strings
   (c) Double line Strings (d) Multiple Strings

5. Strings in python:

   (a) Changeable (b) Mutable
   (c) Immutable (d) flexible

6. Which of the following is the slicing operator?

(a) { } (b) [ ] (c) < > (d) ( )

7. What is stride?

(a) index value of slide operation (b) first argument of slice operation

(c) second argument of slice operation (d) third argument of slice operation

8. Which of the following formatting character is used to print exponential notation in upper case?

(a) %e (b) %E (c) %g (d) %n

9. Which of the following is used as placeholders or replacement fields which get replaced along with format( ) function?

(a) { } (b) < > (c) ++ (d) ^^

10. The subscript of a string may be:

(a) Positive (b) Negative

(c) Both (a) and (b) (d) Either (a) or (b)

Part -II

Answer the following questions (2 Marks)

1. What is String?
2. Do you modify a string in Python?
3. How will you delete a string in Python?
4. What will be the output of the following python code?

```
str1 = "School"
print(str1*3)
```

5. What is slicing?

Part -III

Answer the following questions (3 Marks)

1. Write a Python program to display the given pattern

```
C O M P U T E R
C O M P U T E
C O M P U T
C O M P U
C O M P
C O M
C O
C
```

2. Write a short about the followings with suitable example:

   (a) capitalize( )   (b) swapcase( )

3. What will be the output of the given python program?

```
str1 = "welcome"
str2 = "to school"
str3=str1[:2]+str2[len(str2)-2:]
print(str3)
```

4. What is the use of format( )? Give an example.
5. Write a note about count( ) function in python.

Part -IV

Answer the following questions (5 Marks)

1. Explain about string operators in python with suitable example.

Reference Books

1. *https://docs.python.org/3/tutorial/index.html*
2. *https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list*
3. *Python programming using problem solving approach – Reema Thareja – Oxford University press.*
4. *Python Crash Course – Eric Matthes – No starch press, San Francisco.*