BACHELOR OF COMPUTER APPLICATIONS

SEMESTER *5*

DCA3104

PYTHON PROGRAMMING

Unit 7

String Handling

Table of Contents

1. INTRODUCTION

In the previous section, we have discussed OOP (Object-Oriented Programming) in Python. Python is a dynamic, versatile, and open-source programming language. You can read a python program very easily. It can run on different software available, which makes it a portable language. Different methods and approaches can be implemented in Python to solve a problem. Object-Oriented Programming methods solve the problem by creating objects and classes. This method makes programming easy to understand. You can even share the classes that you have created.

STUDY NOTE

Unlike C++ and Java, Python Programming does not utilize braces to delimit codes. Indentation is compulsory in Python. If you select to import it from the _____ further package____, it provides you a witty error.

Now in this unit, we are going to tell you further about this programming language. String handling is a crucial topic in any programming language. As we already know that Python has a large standard library that contains numerous codes, modules, and functions that makes it easy to code in this language. Python has a built-in class called 'str' for strings that consist of several features. In pythons, strings can be closed by either single quotes or double-quotes. Most people commonly use single quotes for their convenience. However, use single quotes for closing your string when it contains double quotes in it. Similarly, use double quotes to contain your string when your string contains a single quote in it to give a clear code (e.g., "I don't need it").

A string can be a word or a sentence. It can also contain multiple lines, but a backlash should succeed each line to avoid a new line. Multiple line strings can be depicted by ''' or """. Strings in Python are known as immutable because they cannot be changed once they are created. New strings can be created using old strings. For instance, to make a string 'Hi there', we can add up two strings ('Hi' + 'there') to produce the desired string.

1.1 Learning Objectives

After studying the chapter, you will be able to:

- *Understand the syntax and structure of strings in Python.*
- *Explain the process of creating a string.*
- *Describe Indexing and Splitting of strings.*
- *Discuss reassigning strings in Python.*
- *Elaborate on the method of deleting the strings.*
- *Understand String Operators in Python.*
- *Explain the process of string formatting.*
- *Discuss different string methods.*

2. CONCEPTS OF STRINGS

A collection of characters enclosed by either single or double quotes is called a string. A character is a number, symbol, or letter; that is why strings are also considered arrays. Computers stores the values of strings in the form of binary numbers, therefore in a combination of 0 or 1. As we have already discussed, a python string can be expressed as 'hello' or "hello".

The string is a sequence of characters, and therefore there is no such limit to the characters in a string. We can determine the number of characters in a string by the len(string) function, which will be further discussed in the coming sections.

STUDY NOTE

A variable can be assigned to each string to make a code easier to write and read.

As we already know, Python's strings are immutable, so once they are formed, they cannot be changed. However, there are different ways to create new strings from the older ones discussed in separate sections one by one. These methods and functions are pre-built in Python's standard library to carry out these processes of creating new strings. We also have a wide range of operators to develop new strings explained in the coming sections.

We can access the characters in the strings using the square brackets [] syntax. Zero-based indexing is used in Python, therefore for the string 'good', s[1] is o. Python displays an error message if the index is out of bound.

SELF-ASSESSMENT QUESTIONS - 1

1. The data or text enclosed with single quote, double-quote or triple quote is known as _____.

2. The string which is having 0 characters is known as _____.

3. Strings are immutable. **[True/False]**

4. 'It's very good' is a valid string. **[True/False]**

5. Which of the following can be used in enclosing a string?
   A. Single quotes
   B. Double quotes
   C. Triple quotes
   D. All of the above

3. CREATING A STRING

As said before, strings in Python can be made by enclosing a sequence of characters in single quotes, double quotes, or triple quotes depending upon the string. E.g.

STUDY NOTE

Triple quotes can be used when multiple lines are present in the string.

```
#creating strings in python
#using single quotes
new_string = 'Hello'
print(new_string)

#using double quotes
new_string = "Hello"
print(new_string)

#using triple quotes
new_string = '''You are welcome in our house.'''
print(new_string)

#The output is as follows:
Hello
Hello
You are welcome in our house.
```

Activity I

Suppose you are a software engineer working in a company. You need to develop some programs based on Python Programming and explain the importance of Python programming to your team. Start with creating a program that prints a string containing a single quote. For example, a string which says, (Hey there, I haven't seen you in a very long time). Try with all the types of quotes, and then discuss the results with your team.

SELF-ASSESSMENT QUESTIONS - 2

6. Strings are the sequence of _____.

7. A string can be implemented as _____.

8. Strings are enclose by ( ). [True/False]

9. Multiple lines can be enclosed by triple quotes. [True/False]

10. Which is the correct string?
    A. “a”
    B. “You’re looking good”
    C. “hello”
    D. All of the above

4. INDEXING AND SPLITTING

Indexing means acknowledging an element within the string by its position from either left or right. Indexing can be done by either positive or negative numbers depending upon the position from where it is starting. We use the standard [] syntax to specify the position of an element within the string. In contrast, splitting or slicing means to get only a part of a subset from the present string. [ ] is known as the slicing operator. We have presented examples below for you to understand this concept more clearly.

First, let's see an example of indexing. Consider a string 'football' of 8 characters. As we know that Python allows zero-based indexing, so we can start indexing from 0 and will end on 7. For str[1], the value returned will be the second character of the string from the left, that is 'o'. Similarly, for str[6], the value returned by the string will be the sixth character from the left, that is 'l'. If you want to access characters in the string from the right side, you can use negative numbers. For the last element in the string 'football', str[-1] will be used. Similarly, for str[-6], the returned value will be 'o'. This is useful for programmers when they want to access a character or element from the string; they need not find the string's length, just use the indexing from the right.

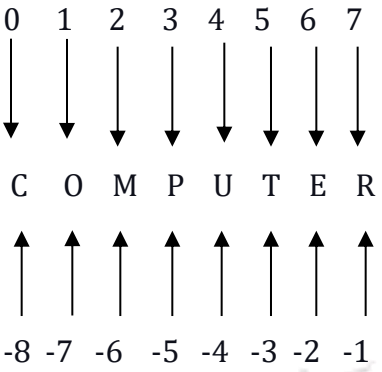
STUDY NOTE

To get a reverse string, we use str[::-1]. The slice will take the whole string, but it will start from the last element. For example, for the given string 'COMPUTER', the value of str[::-1] will be 'RETUPMOC'.

Note that if you try to use the index out of range or use a number that is not an integer, Python raises an IndexError. E.g., for the string 'football', str[8] will cause Python to raise an error because there is no eighth element, and str[2.5] will also raise an error, and Python will ask the programmer to use an integer and nothing else.

Let's come to the splitting of strings. As discussed already, splitting refers to accessing a subset or smaller string from the given string. The syntax for this operation takes the form of string[start:stop], where start refers to the index of the first character that is to be included and stop refers to the index of the last character or element to which the slicing operator has to take without including the last character.

Let's take an example to understand more clearly. Take a string 'COMPUTER' and index it from both the left and right sides.



Both the forward and reverse indices are given for the string 'COMPUTER'. So for str[1:4], the value will be 'OMP'. You can see in this particular example that we have not included the fourth value; the stop only indicates where the computer has to stop. Let's look at more examples.

For str[3:], the value will be 'PUTER'

For str[:5], the value will be 'COMPU'

For the negative splitting, see that we have started from -1 and stopped at -8. So for str[-5:-1], the string value will be 'PUTE'. Note that whenever we split or slice the string, the element that has to be included in the slice will have to be at the right side of the start value and similarly to the left of the stop. So if you try for str[-1:-5], no values will be included in the sliced string because there is nothing to be included to the right of index -2.

Activity II

Create two non-empty strings with n number of elements. Modify the program to rotate all the letters in the string by:

1. Left by two digits
2. Right by two digits

Discuss the approach used and the output of the program.

SELF-ASSESSMENT QUESTIONS - 3

11. Getting a subset of the string is called _____.

12. The method to refer each element in a string by its position is called ____.

13. Indexing can be done only from the left side. [True/False]

14. You can reverse a string by using the slice operator. [True/False]

15. Which will give an empty string?
    A. str[4:6]
    B. str[:3]
    C. str[-2:-4]
    D. str[1:]

5. REASSIGNING STRINGS

The strings in Python are immutable; that is, they cannot be changed once created. But there are ways to update it. The easiest thing you can do is assign a completely different value to the existing string. This is most useful as in Python; item assignment is not supported in strings. Let's take a look at the following example.

```
#creating a string
str = 'PYTHON'
str[1] = 'y'
print(str)
```

For this output, Python will show an error because the item assignment is not valid in Python. Now reassign the existing string to a completely new value that you want in the output.

```
#printing the existing string
str = 'PYTHON'
print(str)
#reassigning a new value to the existing string
str = 'python'
print(str)
#The output will be as follows :
PYTHON
python
```

Python programming language also has an in-built function in its standard library where you can use it to replace the string's value with what you want and however many times you want. The syntax for the replace() function is shown below.

string.replace(old, new, count)

The old parameter refers to the old string that has to be replaced. The new parameter refers to the new string that has to replace the old one. The count parameter shows how many times the old one

STUDY NOTE

Python allows you to allocate the same value in many variables in one statement. It will also allow you to allocate values to many variables at once.

has to be replaced with the new one. Look at the example shown below:

```
#using the replace function
str = "HELLO HELLOHELLOHELLO"
print(str)
#printing the replaced HELLO with hello
print(str.replace("HELLO", "hello"))
#printing the replaced HELLO with hello 3 times only
print(str.replace("HELLO", "hello", 3))
The output is given below.
HELLO HELLOHELLOHELLO
hellohellohellohello
hellohellohelloHELLO
```

We hope that the above example has made the concept of reassigning strings clear to you.

SELF-ASSESSMENT QUESTIONS - 4

16. To replace the value within the string, you can use this function. _____.

17. Item assignment in strings is not valid in Python. **[True/False]**

18. The strings in Python can be changed once they are created. **[True/False]**

19. Reassigning a string to a variable is not possible. **[True/False]**

20. Which is the valid statement?
    A. x= HELLO
    B. x=”HELLO”
    C. x : HELLO
    D. x== HELLO

6. DELETING THE STRINGS

The strings cannot be changed once they are created, so for convenience, if a string is no longer required, you can delete that string with the del keyword. Take a look at the example given below:

```
#we cannot delete a single element from the string
str = 'BASKETBALL'
del str[4]
```

The above program's output will display Python's error because the elements cannot be deleted from the string. Let's take a look at another example.

STUDY NOTE

We cannot change or delete the character directly from an existing string, but we can take the help of slicing and the operators to get the desired result.

```
#deleting the whole string using the del keyword
Str1 = "BASKETBALL"
del str1
print(str1)
```

Again, the output will show that there is no such string named str because we have deleted it already before the printing function.

SELF-ASSESSMENT QUESTIONS - 5

21. The keyword used to delete a string is _____.

22. You can delete an element in the string in Python. **[True/False]**

23. The eliminate() function is used to delete the string. **[True/False]**

24. You cannot delete a string once it is formed. **[True/False]**

25. Which is the correct syntax?
    A. del str
    B. del str[4]
    C. del (str)
    D. del str[1:4]

7. STRING OPERATORS

Strings operators in Python refer to the different types of string operations applied to code's string type of variables. When used in any code, these operators make it much easier to write and read. Pythons allow several operators that are presented below:

- Concatenate operator : "+"
- String repetition operator: "*"
- Assignment operator: "=."
- String slicing operator: "[]"
- String comparison operator: "==" & "!="
- Membership operator: "in" & "not in"
- String formatting operator: "%"
- Escape sequence operator: "\"

Concatenation of strings using "+"

Concatenation refers to adding or joining two or more strings with the "+" operator's help. Take a look at the code given below.

```
#joining of two strings
str1 = ' hello'
str2 = 'there'
combined_string = str1 + str2
print(combined_string)
#The output of the above code given by Python is presented below.
hellothere
```

Repetition of strings using "*"

A string can be repeated any number of times by using the "*" operator. Look at the code given below to understand the operator clearly.

```
#repetition of strings
str = 'hey'
```

```
print(str*2)
print(str*3)
print(str*4)
#The output is as follows.
heyhey
heyheyhey
heyheyheyhey
```

Assignment of strings using "="

To write a code more easily and keep the code clear, programmers often use this operator to assign the string to a variable. To get a more clear picture, go through the following examples.

STUDY NOTE

The placeholders can be empty or contain names or numbers following the values that have to be inserted.

```
#assigning the strings to the variable
var1='PYTHON'
var2='JAVA'
var3='PYTHON'
print(var1)
print(var2)
print(var3)

#The output is given below.
PYTHON
JAVA
PYTHON
```

Slicing the strings using "[ ]"

We have already discussed the splitting or slicing of strings in the previous sections. This operator is used to take out the elements from the given string using the slicing operator.

```
#use of slicing operator
str = 'HELLO'
print(str[1:4])
```

```
#The output of the above code is given below.
ELL
```

Comparison of the strings using “==” & “!=”

Equals to is presented by "==" and not equals to is presented by "!=". when two strings are compared using the equals to the operator, then the boolean value TRUE is returned if both the strings compare the same and FALSE is returned if they are not same. Similarly, when two strings are compared using the not equals to the operator, then the boolean value TRUE is returned if they are not the same, and FALSE is returned if they are the same.

```
#comaparing two strings
str1 = ‘hello’
str2 = ‘hello there’
str3 = ‘hello there’
str4 = ‘hi’
print(str1 == str4)
print(str2 == str3)
print(str2 != str3)
print(str2 != str4)
The output is given below.
FALSE
TRUE
FALSE
TRUE
```

Membership operator “in” & “not in”

These two operators are used to see whether a particular element is present in the string or not. For instance, 'x' in a string returns the boolean value TRUE if x is indeed present in the string and FALSE if not. Similarly, 'x' not in a string returns the boolean value TRUE if x is not present in the string and FALSE if it is present.

```
#using the membership operator
str = 'hellothere'
print('x' in str)
print('t' in str)
print('o' in str)
print('there' not in str)
print('hi' not in str)
print('llo' not in str)
#The output for the following code is given below.
FALSE
TRUE
TRUE
FALSE
TRUE
FALSE
```

String formatting using "%"

The format of the string is specified using this "%" operator. Some commonly used string formatting specifiers are given below:

- %d for signed decimal numbers
- %u for unsigned decimal numbers
- %c for characters
- %f for floating-point real numbers
- %s for strings

Look at the code given below.

```
#usig the string formatting specifier
str = 'WORLD'
str1 = 'HELLO %s' %(str)
print(str1)
The output will be :
HELLO WORLD
```

Escape sequence operator "\"

This operator is very useful when you have to add a non-allowed element in a string. The non-allowed operator is preceded by the backslash or the escape operator "\".

```
#not using the escape sequence operator
str = "Have you visited "India" yet?"
print(str)
```

This code will generate an error.

```
#using the escape sequence operator
str = "Have you visited \"India\" yet?"
print(str)

#The following output will be generated.

Have you visited "India" yet?
```

SELF-ASSESSMENT QUESTIONS - 6

26. The joining of two strings is called _____.

27. Escape sequence character is used when you want to insert a non-allowed character. [True/False]

28. Assignment of the strings is done using the operator "==". **[True/False]**

29. Repetition of the string can be done by "*". **[True/False]**

30. Which of the format specifiers is wrong?
    A. %d
    B. %u
    C. %s
    D. %g

8. STRING FORMATTING

Python uses C-style string formatting to create new formatted strings. As we have already learned the old way of string formatting using the "%" operator, we can now move forward to a new and different string formatting method.

8.1. format() METHOD

The format() method is the new method to create formatted strings in Python. It formats the given values and them in the string to the specified position. This format() method only returns the formatted values in the output. The syntax for the format() method is given below:

STUDY NOTE

For example, you want to insert double quotes in a string enclosed by double quotes, then by the use of a backslash; we could easily add the element in the string.

str.format(val1, val2…)

In this, the val1 and val2 represent the values to be formatted and added in the string. It can represent the element's position that has to be formatted and can be from any data type. The placeholders {} are present in the string in which the values have to be inserted. Look at the given code below to understand in a better way.

```
#using the format() method
str1 = "My name is {}. "
str2 = "I am {} years old. "
str3 = "My name is {fname} and I am {age} years older."
print(str1.format("Mary"))
print(str2.format(19))
print(str3.format(fname = "Mary", age = 19))
#The output will be as follows.
My name is Mary.
I am 19 years old.
My name is Mry, and I am 19 years old
```

SELF-ASSESSMENT QUESTIONS - 7

31. This method is used when we have to create formatted strings in Python _____.

32. "&" operator is used for the string formatting. **[True/False]**

33. Placeholders should not be empty. **[True/False]**
34. C-style string formatting is used by Python. **[True/False]**

35. Which one of these is the correct syntax?
    A. string.format()
    B. string.format(val1, val2…….)
    C. format(string)
    D. format(string(val1, val2…))

9. DIFFERENT STRING METHODS

Python programming language has a wide range of built-in methods that can be used whenever necessary on the code. A string object calls the methods according to their properties.

9.1. capitalize()

This python method is called when you have to convert the first character of the string to a capital letter or uppercase. The syntax of this method is given below:

str.capitalize()

No parameters are used in this method. In case a number is present as the first character of the string, then no change happens when this method is called. Look through the given code:

```
#using capitalize() method
str = 'nice to meet you.'
print(str.capitalize())
#The output is as follows.
Nice to meet you.
```

9.2. casefold()

You can call this method to convert the strings into lower case. The syntax of this method is given below.

str.casefold()

STUDY NOTE

All the methods, when used in a code, return new values. No change in the original string is done.

No parameters are used in this method. Look through the given code:

```
#using casefold() method
str = 'Nice To Meet You.'
print(str.casefold())
#The output is as follows.
nice to meet you.
```

9.3. Center(Width ,Fillchar)

This method is used to return a centred string. The syntax of this method is given below.

str.center(width ,fillchar)

In this, the width is the required length according to the programmer's input and the fillchar (optional) is the character used to fill in space. The default option for fillchar is space, that is " ".Look through the given code in which “0” is used as the padding character:

```
#using center(width ,fillchar) method
str = ‘python’
print(str.center(10 ,”0”))
#The output is as follows.
00python00
```

9.4. count(string,begin,end)

This method is called when you want the number of times a particular element is used or occurred in the string. The syntax of this method is given below.

str.count(string,begin,end)

The string here is the one in which we have to count the values. Begin specifies the position to start from, but the default is set to the start. End refers to position till we want to search, and the default value for the end is the last of the string. Look through the given code:

```
#using count(string,begin,end) method
str = ‘There are you, I was looking for you.’
print(str.count(“you”))
#The output is as follows.
2
```

9.5.join(seq)

This method is called when you have to join or concatenate all the iterables together. There must be a string given to separate the iterables. The syntax of this method is given below.

str.join(seq)

In this,seq refers to the sequence in which the strings have to be joined. For example:

```
#using join(seq) method
tuple = (“Hello”, “my”, “friends”)
a = “x”.join(tuple)
print(a)
#The output is:
Helloxmyxfriends
```

9.6. Len(String)

In Python, whenever we have to find out the length of the string, we call this len() method to know the value. The syntax of this method is given below.

```
len(string)
Look through the given code:
#using len(str)method
str = ‘I love Python’
print(len(str))
#The output is as follows.
13
```

9.7. Lower()

This method is called when you have to convert the whole string into a lower case. The syntax of this method is given below.

str.lower()

No parameters are used in this method. Look through the given code:

```
#using lower() method
str = 'HI THERE'
print(str. lower())
#The output is as follows.
hi there
```

9.8. partition()

This method is called to divide the string into three parts and gives a tuple in return. The syntax of this method is given below.

str.partition()

The value that is inserted in the bracket decides the splitting point. The first part is before the given value, the second part is the value itself, and the third part is after the value. Look through the given code:

```
#using partition() method
str = 'I loved the show'
print(str.partition('the'))
#The output is as follows.
("I loved", "the", "show")
```

Activity III

Create a string in a program. Make only half of the letters in the string in uppercase. Let the other half letters remain in the lowercase. Create this program using at least two different approaches. Discuss the results.

SELF-ASSESSMENT QUESTIONS - 8

36. This method returns a centered string _____.

37. casefold() converts the strings into lower case. **[True/False]**

38. lower() converts the first character of the string into lower case. **[True/False]**

39. partition() divides the string into as many as parts you want. **[True/False]**

40. Which one of these is the correct syntax?
    A. count(element)
    B. count.string(element)
    C. element.count(string,begin,end)
    D. str.count(string,begin,end)

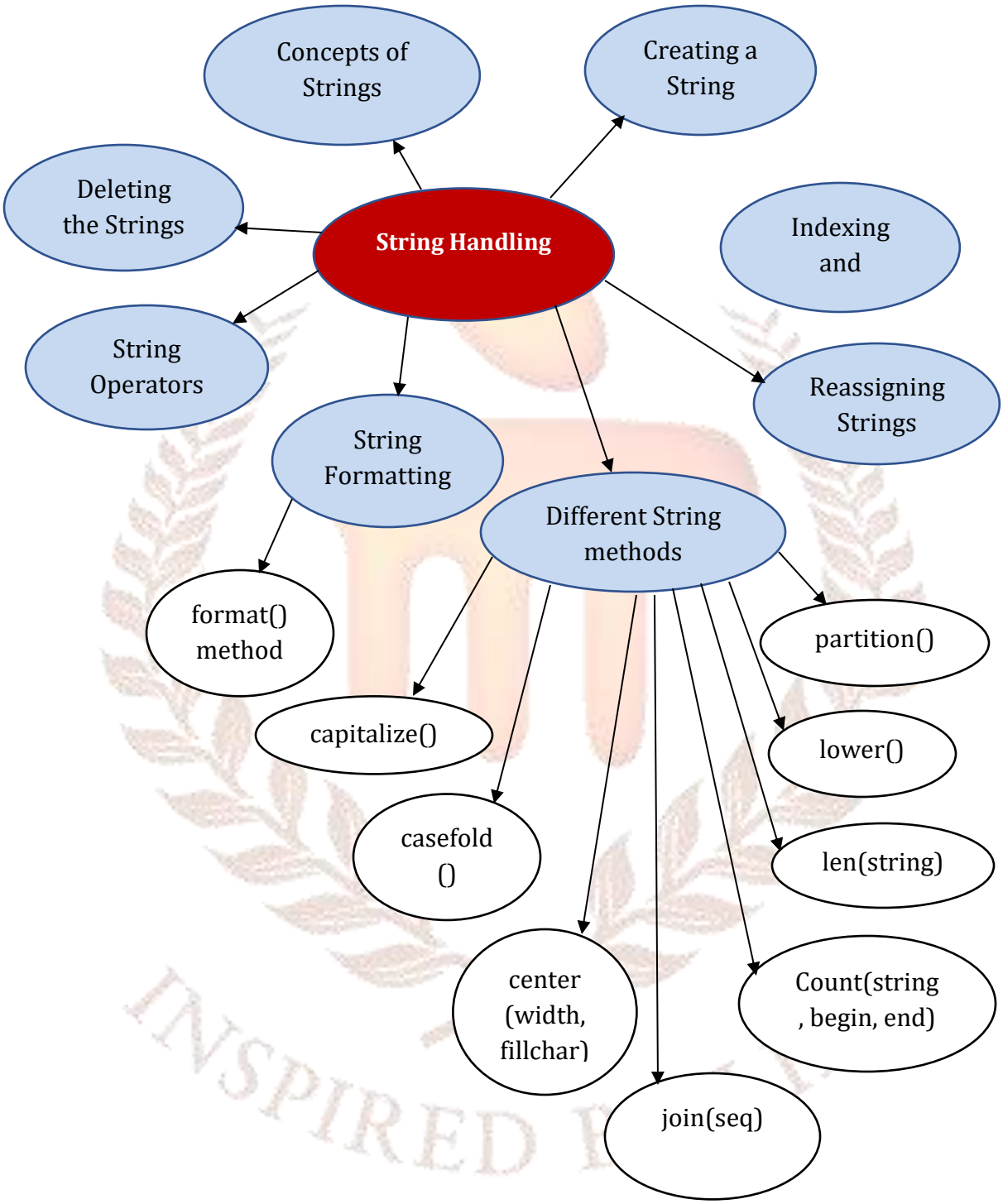


Fig 1: Conceptual map

10. SUMMARY

- A sequence of characters enclosed by either single or double quotes is called a string
- Strings in Python are known as immutable because they cannot be changed once they are created.
- Triple quotes can be used when multiple lines are present in the string.
- Indexing refers to acknowledging an element within the string by its position from either left or right.
- Item assignment is not supported in strings.
- Splitting or slicing means to get only a part of a subset from the present string.
- Assigning a completely different value to the existing string is called reassigning a string.
- If a string is no longer required, then you can delete that string with the del keyword.
- Strings operators in Python refer to the different types of string operations applied to code's string type of variables.
- Concatenation refers to adding or joining two or more strings with the "+" operator's help.
- A string can be repeated any number of times by using the "*" operator.
- The assignment operator is used to assign the string to a variable.
- The placeholders can be empty or contain names or numbers following the values that have to be inserted.
- The format of the string is specified using this "%" operator.
- The format() method is the new method to create formatted strings in Python.
- The escape sequence operator is used to add a non-allowed element in a string.
- capitalize() converts the first character of the string to a capital letter or in uppercase.
- casefold()converts the strings into lower case.
- center(width ,fillchar) returns a centered string.
- join(seq) joins or concatenates all the iterables together.
- len(string)finds out the length of the string.
- lower() converts the whole string into a lower case.
- partition() divides the string into three parts and gives a tuple in return.

11. GLOSSARY

- **String:** Arrays representing unicode characters.
- **Variable:** Reserved memory location to store strings or any other values.
- **Slicing:** To obtain a slice in a sequence.
- **Indexing:** The method to refer to each element in a string by its position.
- **Concatenate:** Addition of two or more strings together.
- **Comparison:** checking if two strings are the same or not.
- **Boolean Value:**Either True or False
- **Operator:** Special symbols that carry out arithmetic or logical operations in Python.
- **Placeholder:** A container in which content can be placed.

12. CASE STUDY

A CASE STUDY ON PYTHON PROGRAMMING

Python is an object-oriented, interpreted language with an organized built-in data types. It is executed in C language in an object-oriented fashion. The design is an appropriate model for language execution. The Python interpreter performs by reading a line typed at the keyboard or loading a source file, translating it into an abstract syntax tree, integrating the tree into bytecode, and implementing the bytecode.

PARSING:

The parsing process is a standardized process. The foremost idea is to first translate the input characters into a more conceptual representation like name: x, integer: 7. Theconceptual characters are referred to as tokens. The tokenization process is completely defined in the language reference.

For example:

```
while(x <= 3):
f(x)
might be tokenized as
keyword: while
left-paren
name: x
leq
int: 3
right-paren
colon

indent
name: f
left-paren
name: x
right-paren
```

Then the tokens are collected into statements, class definitions, expressions, function definitions, etc.

***Source**: alumini.media.mit.edu*

DISCUSSION QUESTIONS:

1. From the above study, determine how the Python interpreter performs?
2. As a software programmer, explain the Python fundamentals in Data Science and do you think Python Programming is beneficial in technological companies?

13. TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Q1. Write a program to find the length of the string "python" without using the len function.

Q2. Write a program to check if the letter 'o' is present in the word 'World'.

Q3. Specify the type of value that will be returned by the centre (width, fillchar) method in Python.

Q4. What do you mean by string slice?

Q5. Elaborate the statement – “Python strings are immutable”.

LONG ANSWER QUESTIONS

Q1. In the given string “BANANA”, count the occurrences of all characters within a string.

Q2. Reverse and print the string “DEVELOPMENT.”

Q3. Explain the use of the format() method by an example.

Q4. Join the list of strings separated by "*."

Q5. Convert a string into uppercase.

14. ANSWERS

SELF ASSESSMENT QUESTIONS

1. String
2. Empty String
3. True
4. False
5. D. All of the above
6. ASCII codes
7. Array data structure
8. False
9. True
10. A and C
11. Splitting
12. Indexing
13. False
14. True
15. C. str[-2:-4]
16. replace()
17. TRUE
18. FALSE
19. FALSE
20. B. x = “HELLO”
21. Del
22. FALSE
23. FALSE
24. FALSE
25. A. del str
26. Concatenation
27. TRUE
28. FALSE

29. TRUE
30. D. %g
31. format()
32. FALSE
33. FALSE
34. TRUE
35. B. string.format(val1, val2…….)
36. center(width ,fillchar)
37. TRUE
38. FALSE
39. FALSE
40. D. str.count(string,begin,end)

TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Answer 1:

```
s = "python"
count = 0
for char in s:
   count += 1
print("Length of the string is:", count)
```

Answer 2:

```
str = “World”

print(“o” in str)
```

Answer 3: The method centre( ) returns centred in a string of width specified by the programmer. The padding will be done by the character specified by the programmer.

Answer 4: A subset or a slice of a string is known as a string slice.

Answer 5: In Python, once the strings are created, then they cannot be changed. The new operation has to be performed, or methods are called to change something and form new strings.

LONG ANSWER QUESTIONS

Answer 1:

```
str = “BANANA”
print(str.count(“B”))
print(str.count(“A”))
print(str.count(“N”))
```

Answer 2:

```
str1 = str1[::-1]
print("Reversed String is:", str1)
```

Answer 3:

```
string = “The { } was in { }”
print(string.format(question, spanish))
```

Answer 4:

```
list = (“I”, “love”, “Python”)
a = “*”.join(list)
print(a)
```

Answer 5:

```
str = ‘what a extravagent party!’
print(str.capitalize())
```

15. SUGGESTED BOOKS AND E-REFERENCES

BOOKS:

- Paul Barry (2016), Head First Python: A Brain-Friendly Guide, 1st edition, O'Reilly Publisher.
- Wes McKinney (2017), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Ipython, 1st edition, O'Reilly Publisher.
- Swaroop C H (2013), A Byte of Python, Kindle edition, ebshelf Inc.
- Charles Severance (2016), Python for Everybody: Exploring Data in Python 3, Kindle edition, Amazon Digital Services.

REFERENCES:

- The Python Standard Library, viewed March 27, 2021

  <https://docs.python.org/3/library/string.html>

- Python String Methods, viewed March 27, 2021

  <https://www.w3schools.com/python/python_ref_string.asp>

- Python String formatting, viewed March 27, 2021

  <https://realpython.com/python-string-formatting/>

- Python String Operators, viewed March 28. 2021

  <https://realpython.com/lessons/string-operators/>