PYTHON

String : A string is a sequence of characters enclosed in ‘ ’(or)“ ”(or)’’’ ’’’.A character is anything you can type from the keyboard.

- Strings are immutable.

- An empty string is a string that has 0 characters.

Creating a string : Strings in python can be created using single or double or triple quotes.

- Single and double quotes works same but triple quotes are used for writing string in multiple lines and print output without using escape sequences.

Example:

```
name=”python”
type=”programming”
desc=’’’python is
        one of the best
        programming language presently’’’
```

Strings are Immutable : Once a string is defined it cannot be modified as they are immutable.

Example:

```
n=”python”
n[1]=”J”
print n
```

- This will show an error as we tried to insert value of J at index [1] but this is not possible as strings are immutable.

- To print single and double quotes on the screen we can use two methods:

    1)By using the escape characters.

2) By using mix quote, but while using this we have to make sure that to print single quotes we should use double quotes as delimiters and to print double quotes use single quotes as delimiters.

Example:

```
print ("python,\"The best language\"")

print ("Datascience,'uses python'")
```

Accessing Strings : Characters in a string can be accessed using indexing. The value should be written in square brackets.

• Basically Indexing starts with 0.If the value provided is larger than the length of string it will raise an exception as IndexError.

• Index must be an integer not floating point value.

• Python also supports negative indexing which is we can access from right side of the string by using negative values i.e,[-1].

• To access more than one character in a string we can use slicing (:).

Example:

```
A = 'python'

print(A)

#first character

print(A[0])

#last character

print(A[-4])

#slicing 2nd to 5th character

print(A[1:5])

#slicing 3th to 1nd last character

print(A[3:-1])
```

Output:

python

p

t

ytho

ho

· If we try to access index out of range then it will raise an error.

Change(or)Delete a string : As the string are immutable we cannot make changes in the string and delete a character from the string, But we delete entire string by using **del().**

Example:

```
str=”programming”
del str[4]
print  (str)
del(str)
print (str)
```

· When tried to delete str[4] it will show an error as that method is not possible.

· We can delete the entire string by using del(str).

· While attempting to print str at the last it show error because str is already deleted. It shows **NameError : name “str” is not defined.**

Basic Operations : The operations that can be performed on string are as follows:

Concatenation : Concatenation is the process of combining two or more string and making it as a one string.

· The + operator is used for concatenation and another way to concatenate is writing two string together.

Example:

```
name="python "
str="Welcome to python"
print (name + str)
str1 = 'Hello ''World!'
print (str1)
```

· It will provide the output as **python Welcome to python**

Hello World

· To concatenate strings in multiple lines we use parenthesis.

```
A=("Build "
    "Career with "
    "Python")
print (A)
```

· This is will print output as **Build Career with Python** even though it is in multiple line by using parenthesis.

Repetition : Repetition is the process of repeating a string to the no.of times specified.

· The * operator is used for repetition purpose.

Example:

```
name="python "
print(name*3)
```

· It will give output as **python python python**

Iterating String : This will help to go to every character in the string by using a loop condition (for loop).

· It is also used for testing purpose whether the character present in the string if yes how many times it is repeated.

Example:

```
count = 0
for letter in 'Python is a programming language':
if(letter == 'l'):
count += 1
print(count,'letters found')
```

• In the example it is going to check for ‘l’ whether it is present in the provided string and print how many times it is repeated in the string.

Membership : These are used to test whether the substring exist in the provided string are not by using **in** and **not in** statements.

• If the string exist it will give output as True otherwise the output will be False.

Example:

```
str='datascience is growing'
str1='growing'
print (str1 in str)
print (str1 not in str)
```

• In the example two strings are defined where we applied membership to second string with first string as it is present it will display True .

String Methods : The methods that are used to perform different operations on string is called string methods.

Different string methods are :

1)startswith():This will check whether the string startswith the specified prefix or not.

• If condition is satified it will return True otherwise it will return False.

Syntax:

```
startswith(“substring”,begin,ending)
```

・Here we can specify where the search has to begin and where it has to end by specifying the index value at beginning and ending.

Example:

```
str='datascience is growing'

str1='growing'

print(str.startswith(str1))

print(str.startswith(str1,15))
```

・In example it will check whether str startwith str1.In first print it will be False as it is not starting with str1.But, in the second print it will be True because here we gave the index value where it has to begin the checking.

2) endswith():This will check whether the string endswith the specified suffix or not.

・ If condition is satisfied it will return True otherwise it will return False.

Syntax:

```
endswith(“substring”,begin,ending)
```

・Here we can specify where the search has to begin and where it has to end by specifying the index value at beginning and ending.

Example:

```
str='datascience is growing'

str1='growing'

print(str.endswith(str1,15))

print(str.endswith(str1))
```

・In example it will check whether str endswith str1.In both the conditions it is True because it endswith str1 in both the cases.

3)islower():This will check whether all string in lower case or not. If all are in lower case it will return True otherwise it will return False.

Syntax:

islower(“substring”)

Example:

```
str1="growing"
islower(str1)
```

- In example as all the letters are in lower case it will return True.

4)isupper():This will check whether all string in upper case or not. If all are in upper case it will return True otherwise it will return False.

Syntax:

isupper(“substring”)

Example:

```
str1="growing"
isupper(str1)
```

- In example as all the letters are in lower case it will return False

5)lower():This is used to convert all the letters in a string in lower case letters and returns a new string with all lowercase letters.

Syntax:

lower()

Example:

```
str="DIGITALLYNC"
a='DigiTAllyNc'
str1=str.lower()
str2=a.lower()
```

```
print (str1)

print(srt2)
```

· Here it will str is completely in upper case and str2 in mixed it is converted to lower by using the lower method and even it have some uppercase and some lowercase letter by using lower it will completely convert it into the lower case form.

6)upper():This is used to convert all the letters in a string in upper case letters and returns a new string with all uppercase letters.

Syntax:

```
upper()
```

Example:

```
str="digitallync"

a='DigiTAllyNc'

str1=str.upper()

str2=a.upper()

print (str1)

print(srt2)
```

· Here it will str is completely in lowercase and str2 in mixed and it is converted to uppercase by using the upper method and even it have some uppercase and some lowercase letter by using upper it will completely convert it into the uppercase form.

7)swapcase(): It will be used for swapping the letters case that is uppercase to lowercase and lower case to uppercase.

Syntax:

```
swapcase()
```

Example:

```
str="digitallync"

a='DigiTAllyNc'
```

```
str1=str.swapcase()

str2=a.swapcase()

print (str1)

print(srt2)
```

• Here str is completely in lowercase so it is converted into uppercase but a is mixed with uppercase and lowercase then it will convert uppercase to lowercase and viceversa.

str2=’dIGItaLLYnC’.

8)title():It will convert the string into title case that is the first letter will be capital and remaining will be in lower case.

Syntax:

```
title()
```

Example:

```
str="digitallync"

a='digiTAllyNc'

str1=str.title()

str2=a.title()

print (str1)

print(srt2)
```

• Now it will convert the str to titlecase as “Digitallync” and there are not uppercase in the middle of str. But, a has mixed letters and it convert the first to uppercase and remaining to lower case as “Digitallync”.

9)len(): It is used to find the length of the string.

Syntax:

```
len(stringname)
```

Example:

```
str=”digitallync”
```

print (len(str))

- It will calculate the length by count how many letters in the string and return output as 11.

10)count():It will count the number of occurrence of a substring in a string. It take three arguments substring,beginning,ending.

- If beginning and ending id not specified it will take by default as 0 and length of string.

Syntax:

count(“substring”,begin,end)

Example:

```
str="hyderabad is captial of telangana"
print(count("s"))
print(count("s",15))
```

- It will check the substring and give the output as count. Here first print give as 1 and second will give as 0 because in second we specified starting index as 15 after that index there are no s so it is 0.

11)isalpha():This is used to check whether as characters in the string are alphabets or not.

- It will return True if all are alphabets otherwise False.

Synax:

str.isalpha()

Example:

```
A=”google123”
A.isalpha()
```

- It will return False here in the example there are number so it return false.

12)isalnum():This is used to check whether all characters in the string are combination of number and alphabets.

- It will return True if all are combination of alphabets and number and return False if special symbols are used.

Synax:

```
str.isalnum()
```

Example:

```
A="Hyeg1234@"
print(A.isalnum())
```

- It will return False here in the example there are number so it return false.

13)join():This is used to join a sequence of strings mentioned in its arguments with the string.

Syntax:

```
specialsymbol.join(str)
```

Example:

```
str=”hyderabad”
print “@”.join(str)
```

- It will join @ to hyderabad the output will be “h@y@d@e@r@a@b@a@d”.

14)strip():This methods is used to remove all leading and trailing characters and escape sequences form the string.

Syntax:

```
str.strip()
```

Example:

```
str="---Great Charminar---"
str1="\n\tHyderabad\t"
print(str.strip("-"))
```

print(str1.strip())

· This will give the output as **Great Charminar Hyderabad** as strip() will remove the escape sequences and leading and trailing characters.

15)lstrip(): This methods is used to remove all leading characters and escape sequences form left side of the string only.

Syntax:

str.lstrip()

Example:

str="---digital\nlync---"

print(str.lstrip("-"))

· This will give the output as **digital** and **lync--- in next line** as strip() will remove the escape sequences and trailing characters.

16)rstrip():This methods is used to remove all leading characters and escape sequences form right side of the string only.

Syntax:

str.rstrip()

Example:

str="---digital\nlync---"

print(str.rstrip("-"))

· This will give the output as **----digital** and **lync in next line** as strip() will remove the escape sequences and leading characters.

17)min("string"): This method is used to get the minimum value or minimum character from the string.

· This will return the minimum value in the string the order will be from uppercase to lower case.

Syntax:

min(str)

Example:

```
str="Hyderabad"
str1="betgdkZ"
print (min(str))
print (min(str1))
```

· In example **str** the minimum value by seeing is we take ‘a’ but the output will be ‘H’ because it will consider the Upper case letters and then the lower case letters. For str1 output will be **Z**.

18)max(“string”): This method is used to get the maximum value or maximum character from the string.

· This will return the maximum value in the string the order will be from uppercase to lower case.

Syntax:

```
max(str)
```

Example:

```
str="Hyderabad"
str1="betgdkZ"
print (max(str))
print (max(str1))
```

· In example **str** the maximum value output will be ‘y’ . For str1 output will be **t**.

19)replace():This is used to replace the substring in string with new string. This can be done by using the 3 arguments. Old substring which is going to be replaced ,New substring with which we have to replace, value for how many times to replace in the string.

Syntax:

```
str.replace(“oldstring”,”newstring”,value)
```

Example:

str=”If you trouble the trouble the trouble troubles you I am not the trouble I am the Truth”

str1=”trouble”

str2=”problem”

print(str.replace(str1,str2,3))

· The output will be replaced with the new substring i.e, here str1=”trouble” will be replaced with str2=”problem” and the count is 3.so it will change the value only three times.

“If you problem the problem the problem troubles you I am not the trouble I am the Truth”.

20)reversing(): This method is used to reverse the string and the can be done by using the in-built method reversed and by using slicing methods.

Syntax:

str="hyderabad"

print("".join(reversed(str)))

print (str[::-1])

· Both methods will give the same output as “dabaredyh”.

21)split():This method is used to split a string at a particular point in the string.

· It will split the word into 2 but will remove the specified value to split in condition.

Syntax:

str.split(“string or symbol where to split”)

Example:

Str=”digital_lync”

Str.split(‘_’)

· It will give the output as “digital”,”lync” and remove ”_” as here we splitted the word.

22)Zfill():It will check the value and fill with 0's according to the condition specified.

Syntax:

strname.zfill(condition)

Example:

num="456376"

print(num.zfill(6))

print(num.zfill(10))

print(num.zfill(4))

Output:

456376

0000456376

456376

- In 1st print as value and size of zfill is same so not change will be occurred.
- But 2nd print value will be 0000456376 as the size is larger and it is filled with 0's towards left side.
- 3rd print will give the value as 456376 even though the zfill size is smaller than the length of value.

String Slicing : This one of the best method supported by string in python as we can access the letters in the string by using the indexing.

- A method [Start:Stop:Jump] is what slicing main tells. By using this we can get access to any elements in the string or some range of elements.

Syntax:

str[start:stop:jump]

Example:

str="digital"

```
print str[0]=d

print str[0:5:2]=”dgt”

print str[0:-1]=”digita”

print str[::-1]=”latigid”
```

· Here we specified different ways of slicing a string and output accordingly.

String Formatting : String formatting in python can be done by using % and by using {}.

Formatting with % : This is similar to that of printf in c language.

%d-integer value

%f-floating value

%s-string

%x-hexadecimal

%o-octal

· These are different type of numbers that are used in formatting a string.

Example:

```
value="29"

print("Value as string = %s" %(value))

print("Value as raw data = %r" %(value))

value1=int(value)

print("Value as integer = %d" %(value1))

print("Value as float = %f" %(value1))

print("Value as hexadecimal  = %x" %(value1))

print("Value as octal = %o" %(value1))
```

· This will give the output as 29 for string, ’29’ for raw data.

· As %d indicates integer we have to convert the string value to integer by using int() and prints 29,for floating =29.000,for hexadecimal =1D,for octal value =35.

Formatting with {} : The format class in the string module allows you to create and customize your own string formatting behaviours using the same implementation of the built-in format() method. This the standard method which is mostly used in python.

Example:

```
day="Tuesday"

month="october"

year="2017"

print ("Today is {} of the month {} and year{}".format( day, month, year ))

print ("Today is {2} of the month {0} and year{1}".format( day, month, year ))

print ("Today is {} of the month {} and year{} where days are {days}".format( day, month, year, days = 366 ))

print ("Today is {} of the month {} and year {year} where days are {days}".format( day, month, year = 2018, days = 366))

print ("Today is {day} of the month {month} and year {year} where days are {days}".format( day = 4, month = "august", year = 2018, days = 366))
```

output:

```
Today is Tuesday of the month october and year2017

Today is 2017 of the month Tuesday and yearoctober

Today is Tuesday of the month october and year2017 where days are 366

Today is Tuesday of the month october and year 2018 where days are 366

Today is 4 of the month august and year 2018 where days are 366
```

· In example first print statement give the output in the normal format as specified.

· In 2nd print the order is changed as we specified the order how it has to print.

- Next we added an extra data as days by using the format as before and writing days in {} indicates it get data from days in .format().

- To change the values any element we can use {} and specifying the name of elements in {} and assigning the values to the element in .format().