Topics:

- **Strings and its operations**
  - **Concatenating**
  - **Appending**
  - **Multiplying strings**
  - **Built-in String methods and functions**
  - **Slice Operation**
  - **Iterating String**
  - **String Module**

Strings

- In Python, string is a sequence of characters, where a character could be a letter,digit,whitespace or any other symbol
- **Create a string**
- Strings can be created by enclosing characters inside a single quote(‘), double quotes (“) and triple code (‘ ‘ ‘ or “ “ “)
- It must start and end with same type of quote
- Tripe quotes are used to span string across multiple lines
- Python has built-in string class named “str” that has many useful features
- **The index of the first character is 0 and the index of the last character is n-1, where n is the number of characters in the string.** Can be accessed using negative indices. Last character will start with -1 and traverses from right to left

```
#Syntax for Creating a sting
name="RGUKT"
location='''''Basar Temple'''''
Dist='Nirmal'
mandal=str("Basar,")
print(name, location, Dist,mandal)


word = 'Python Programming'
sentence = "Object Oriented Programming"
paragraph ='''''Python is a Object Oriented Programming Language'''''
feedback=str('It is a Biginner's Language')
print(word,sentence,paragraph,feedback)

my_string = 'Hello'
print(my_string)
my_string1 = input("Enter a string")
print(my_string1)
my_string2 = """Hello, welcome to the world of Python"""
print(my_string2)
```

- **access characters in a string**
- We can access individual characters using indexing,using the subscript([ ]) operator and a range of characters using slicing. Index starts from 0. Trying to access a character out of index range(below 0 or above n-1) will raise an IndexError.
- The index must be an integer. We can't use float or other types, this will result into TypeError.
- Python also allows negative indexing for its sequences.

Example

```
str = 'program'
i=2
print('str = ', str)
#first character
print('str[0] = ', str[0])
#last character
print('str[-1] = ', str[-1])
#slicing 2nd to 5th character
print('str[1:5] = ', str[1:5])
print( str[i])
print(str[i*2+2])
```

- **change or delete a string**
- Strings are immutable. This means that elements of a string cannot be changed once it has been assigned. We can simply reassign different strings to the same name.
- We cannot delete or remove characters from a string. But deleting the string entirely is possible using the keyword del.
- **Example**

```
my_string = 'perl'
my_string = 'Python'
print(my_string)
del my_string
print(my_string)
```

- **Concatenation of Two or More Strings**
- Joining of two or more strings into a single one is called concatenation. The + operator does this in Python.
- The * operator can be used to repeat the string for a given number of times.
- **Example**

```
# using +
str1 = 'Hello'
str2 ='World!'
str3=str1+str2
print("The Concatenated string is:", str3)
# using *
str="Hello hw R U"
print('str * 3 =', str * 3)
```

- Append mean to add something at the end. In Python you can add one string at the end of another string using the += operator

```
str="Hello, "
name=input("\n Enter Your name:")
str+=name
str+=" Welcome to Python Programming"
print(str)
```

- The **str( )** function is used to convert values of any other type into string type. This helps the programmer to concatenate a string with any other data which is otherwise not allowed

```
str1="Hello"
var=7
str2=str1+var
print("str2",str2)


str1="hello"
var=7
str2=str1+str(var)
print("str2",str2)
```

- The print statement prints one or more literals or values in a new line. If you don't want to print on a new line then, add end statement with a separator like whitespace, comma etc.

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

```
print("Hello",end='')
print("world")
```

Strings are Immutable

- Python strings are immutable which means that once created they cannot be changed. Whenever you try to modify an existing string variable, a new string is created
- Every object in python is stored in memory. You can find out whether two variables are referring to the same object or not by using id( ).
- The id( ) returns the memory address of that object. As both str1 and str2 points to same memory location, they both point to the same object

```
# Program to demonstrate id() function

str1="Hello"
print("str1 is :",str1)
print("Id of str1 is:",id(str1))

str2="World"
print("str2 is :",str2)
print("id of str2 is:",id(str2))
str1+=str2
print("str1 after concatenation is:",str1)
print("id of str1:",id(str1))
str3=str1
print("str3=",str3)
print("id of str3 is:",id(str3))
```

```
str="hai"
str[0]='o'
print(str)


str1="Hai"
new_str="O"
print("old string",str1)
print("New string", new_str)
```

String Formatting Operator

- One of Python's coolest features is the string format operator %. This operator is unique to strings and makes up for the pack of having functions from C's printf( ) family.
- The % operator takes a format string on the left(that has %d,%s, etc) and the corresponding values in a tuple on the right
- The format operator,% allows users o construct strings, replacing parts of the strings with the data stored in variables.
- **Syntax:** `"<FORMAT>"%(<VALUES>)`

- **# Program to use format sequences while printing a string**

```
Emp_id="sbi1204"
Emp_Age= 35
print("Emp_id=%s and Emp_Age=%d" %(Emp_id,Emp_Age))
print("Emp_id=%s and Emp_Age=%d" %("David",42))
```

```
print('%d %f %s'%(10,'s',20.23))
print('%d %f %s'%(10,20.23))
```

```
Traceback (most recent call last):
  File "C:\Users\Administrator\Desktop\string.py", line 109, in <module>
    print('%d %f %s'%(10,'s',20.23))
TypeError: must be real number, not str
>>>
============ RESTART: C:\Users\Administrator\Desktop\string.py ============
Traceback (most recent call last):
  File "C:\Users\Administrator\Desktop\string.py", line 110, in <module>
    print('%d %f %s'%(10,20.23))
TypeError: not enough arguments for format string
>>>
```

Note: In the first case type of arguments doesn't match, in the second case number of arguments doesn't match

```
print("My name is %s and weight is %d kg!"%('student',45))
```

Formatting Symbols

Format Symbol	Purpose
%c	Character
%d or %i	Signed Decimal Integer
%s	String
%u	Unsigned Decimal Integer
%o	Octal Integer
%x or %X	Hexadecimal Integer (where x for lower case characters a-f and X for upper case characters A-F)
%e or %E	Exponential Notation
%f	Floating Point Number
%g or %G	Short numbers in floating point or exponential notation

```
# Program to display powers of a number without using formatting characters

i=1
print("i\t i**2\t i**3\t i**4\t i**5\t i**6\t i**7\t i**8\t i**9\t i**10")
while i<=10:
    print(i,'\t',i**2,'\t',i**3,'\t',i**5,'\t',i**6,'\t',i**7,'\t'
          ,i**8,'\t',i**9,'\t',i**10,)
    i+=1


# Program to display powers of a number using formatting characters

i=1
print("%-4s %-5s %-6s %-8s %-13s %-15s %-17s %-19s %-21s %-23s"
      %('i','i**2','i**3','i**4','i**5','i**6','i**7','i**8','i**9','i**10'))
while i<=10:
    print("%-4d %-5d %-6d %-8d %-13d %-15d %-17d %-19d %-21d %-23d"
          %(i,i**2,i**3,i**4,i**5,i**6,i**7,i**8,i**9,i**10))
    i+=1
```

Note: - after each % in the conversion string indicates left justification

The numerical values specify the minimum length

%-10d means it is a left justified number that is at least 10 characters wide

Built-in String Methods and Functions

Function	Description	Example
capitalize( )	This function is used to capitalize first letter of the string	str="hello" print(str.capitalize( ))
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	str = "hello" print(str.center(10,'*'))
count(str,beg, end)	Counts number of times str occurs in a string. You can specify beg as 0 and end as the length of the message to search the entire string or use any other value to just search a part of the string	str='he' message="helloworldhelloworld" print(message.count(str, 0,len(message))

Function	Description	Example
endswith(suffix,beg,end)	Checks if string ends with suffix;returns True if so and False otherwise.You can either set beg=0 and end equal to the length of the message to search entire string or use any other value to search a part of it	message="she is my best friend" print(message.endswith("end",0,len(message))
startswith(prefix, beg,end)	Checks if string starts with prefix;if so, it returns True and False otherwise. You can either set beg=0 and end equal to the length of the message to search entire string or use any other value to search a part of it	Str="The world is beautiful" Print(str.startwith("Th", 0,len(str)))

Function	Description	Example
find(str,beg,end)	Checks if str is present in string. If found it returns the position at which str occurs in string,otherwise returns -1.You can either set beg=0 and end equal to the length of the message to search entire string or use any other value to search a part of it	message="she is my best friend" print(message.find("my", 0,len(message)))
index(str,beg,end)	Same as find but raises an exception if str is not found	message="she is my best friend" print(message.index("mine",0,len(message)))

Function	Description	Example
index(str,beg, end)	Same as find but raises an exception if str is not found	message="she is my best friend" print(message.index(" mine",0,len(message)))
rfind(str,beg,e nd)	Same as find but starts searching from the end	str="Is this your bag?" print(str.rfind("is",0,len (str)))
rindex(str,beg, end)	Same as rindex but start searching from the end and raises an exception if str is not found	str="Is this your bag?" print(str.rindex("you",0 ,len(str)))
isalnum( )	Returns True if string has atleast 1 chracter and every character is either a number or an alphabet and False otherwise	message="JamesBond0 07" print(message.isalnum( ))

Function	Description	Example
isalpha( )	Returns True if string has atleast 1 character and every character is an alphabet and False othewise	message="JamesBond007" print(message.isalpha( ))
isdigit()	Returns True if string contains only digits and False othewise	message="007" print(message.isdigit())
islower()	Returns True if string has atleast 1 character and every character is a lowercase alphabet and False otherwise	message="Hello" print(message.islower())
isspace()	Returns True if string contains only whitespace characters and False otherwise	message=" " print(message.isspace())

Function	Description	Example
isupper()	Returns True if string has atleast 1 character and every character is an upper case alphabet and False otherwise	message="HELLO" print(message.isupper()))
len(string)	Returns the length of the string	str="Hello" print(len(str))
ljust(width[,fillchar])	Returns a string left-justified to a total of width columns. Columns without characters are padded with the character specified in the fillchar argument	str="Hello" print(str.ljust(10,'*')

Function	Description	Example
rjust(width[,fillchar])	Returns a string right-justified to a total of width columns. Columns without characters are padded with the character specified in the fillchar argument	str="Hello" print(str.rjust(10,'*')
zfill(width)	Returns string left padded with zeros to a total of width characters. It is used with numbers and also retains its sign(+or-)	str="1234" print(str.zfill(10))
lower()	Converts all characters in the string into lowercase	str="Hello" print(str.lower())

Function	Description	Example
upper( )	Converts all characters in the string into uppercase	str="Hello" print(str.upper( ))
lstrip( )	Removes all leading whitespace in string	str="Hello" print(str.lstrip( ))
rstrip( )	Removes all trailing whitespace in string	str=" Hello " print(str.lstrip( ))
strip()	Removes all leading and trailing whitespace in string	str="Hello " print(str.strip())
max(str)	Returns the highest alphabetical character(having highest ASCII value) from the string str	str="hello friendz" print(max(str))
min(str)	Returns the lowest alphabetical character(lowest ASCII value) from the string str)	str="hello friendz" print(min(str))

Function	Description	Example
replace(old,new [,max])	Replaces all or max(if given) occurrences of old in string with new	str="hello hello hello" print(str.replace("he", "FO"))
title( )	Returns string in title case	str="The world of Python" print(str.title( ))
swapcase( )	Toggles the case of every character(uppercase character becomes lowercase and vice versa)	str="The world is Beautiful" print(str.swapcase())
split(delim)	Returns a list of substrings separated by the specified delimiter. If no delimiter is specified then by default it splits strings on all whitespace characters	str="abc,def,ghi,jkl" print(str.split(','))

Function	Description	Example
join(list)	It is just the opposite of split. The function joins a list of strings using the delimiter with which the function is involved	print('-'.join(['abc','def','ghi','jkl']))
isidentifier( )	Returns True if the string is a valid identifier	str="hello" print(str.isidentifier())
enumerate(str)	Returns an enumerate object that lists the index and value of all the characters in the string as pairs	str="Hello World" print(list(enumerate(str)))

- Try out over string **s="hello this is my class my "**

format( ) function

- format() function used with strings is a powerful function used for formatting strings
- Format strings have curly braces{} as placeholders or replacement fields which gets replaced

```
# program to demosntrate format() function

str1="{},{} and {}".format('sun','Mon','Sat')
print("\n The default sequence of argument is:" +str1)
str2="{1},{0} and{2}".format('sun','mon','sat')
print("\n the positional sequence of arguments(1,0 and 2)is:"+str2)
str3="{c},{b} and {a}".format(a='sun',b='mon',c='sat')
print("\n The keyword sequence of arguments is:"+str3)
```

String Operations and Functions

- **Concatenation**

  -Strings can be concatenated with '+' operator

  "Hello" + "World" will result in HelloWorld

- **Repetition**

  - Repeated concatenation of string can be done using asterisk operator " * "

  "Hello" * 3 will result in "HelloHelloHello"

- **Indexing**

  - "Python" [0] will result in "P"

- **Slicing**

  - Substrings are created using two ndices in a square bracket separated by a ' :'

  - "Python"[2:4]will result in "th"

- **Size**

  - prints length of string

  len("python") will result in 6

Slice Operation

- A substring of a string is called a slice. The slice operation is used to refer the sub-parts of sequences and strings
- You can take substring from the original string by using [ ] operator
- Index starts with 0 and ends with n-1, Negative index starts with -1 to –n
- We can access a range of items in a list by using the slicing operator(colon)
- **Syntax: a[start:end:step]**
- a[start:end] # items from start to end-1
- a[start:]#items from start to the end of the list
- a[:end] #items from start to end-1
- a[:] # all items
- a[start:end:step]# from start to end-1, incremented index by step
- a[-3] #last three items in the list
- a[:-3] # entire list except the last three items
- Normally it considers Left----Right but in case of Negative Step Count it considere Right ---- Left

```
# Program to demonstrate slice operation on string objects
str='PYTHON'
print("str[1:5]=",str[1:5])#Characters starting at index 1
                  #and extending up to but not including index 5
print("str[:6]=",str[:6])#defaults to the start of the string
print("str[1:]=",str[1:])#defaults to the end of the string
print("str[:]=",str[:])#defaults to the entire string
print("str[1:20]=",str[1:20])#an index that is too big is
                  #truncted down to length of the string

str="PYTHON"
print("str[-1]=",str[-1])#last character is accessed
print("str[-6]=",str[-6])#first character is accessed
print("str[-2:]=",str[-2:])#second last and the last characters are accessed
print("str[:-2]=",str[:-2])#all characters upto but not including second last character
print("str[-5:-2]=",str[-5:-2])#characters from second upto second last are accessed

str="Welcome to the world of Python"
print("str[2:10]=",str[2:10])
print("str[2:10:1]=",str[2:10:1])
print("str[2:10:2]=",str[2:10:2])# skips every alternate character
print("str[2:13:4]=",str[2:13:4])# skips every fourth character
print("str[::3]=",str[::3])
print("str[::-1]=",str[::-1])
print("str[::-3]=",str[::-3])
```

in And not in operators

- in and not in operators can be used with strings to determine whether a string is present in another string
- Used to check whether a character is present in a word or not

```
print('u' in 'stars')
print('t' in 'stars')
print('v' not in 'success')
print('s' not in 'success')
print('py' in 'python')
print('pr' in 'python')

str1 ='Welcome to world of Python'
str2='the'
if str2 in str1:
    print("Found")
else:
    print("Not Found")
```

Comparing Strings

- Python allows to compare strings using relational or comparison operators
- The ASCII values of A-Z is 65-90 and a-z is 97-122

Operator	Description	Example
==	If two strings are equal, it returns True	'AbC'=='AbC'
!= or <>	If two strings are not equal, it returns True	'AbC'!='Abc' 'abc<>'ABC'
>	If the fist string is greater than the second, it returns True	'abc'>'Abc'
<	If the second string is greater than the first, it returns True	'abc'<'abc'
>=	If the first string is greater than or equal to the second, it returns True	'aBC'>='ABC'
<=	If the second string is grater than or equal to the first, it returns True	'ABc<='Abc'

Iterating String

- String is a sequence type(sequence of characters)

```
str="Welcome to the World of Python"
for i in str:
    print(i,end=' ')


msg="Welcome to the World of Python"
index=0
while index<len(msg):
    letter=msg[index]
    print(letter,end=' ')
    index +=1
print('\n',len(msg))
```

The String Module

- The string module consists of a number of useful constants, classes and functions . These functions are used to manipulate strings
- String constants defined in the string module are:

string.ascii_letters	**Combination of ascii_lowercase and ascii_uppercase constants**
string.ascii_lowecase	Refers to all lowercase letters from a-z
string.ascii_uppercase	Refers to all uppercase letters from A-Z
string.digits	Refers to digits from 0-9
string.hexdigits	Refers to hexadecimal digits,0-9,a-f,and A-F
string.lowercase	A string that has all the characters that are considered lowercase letters
string.uppercase	A string that has all the characters that are considered uppercase letters
string.octdigits	Refers to octal digits 0-7
string.printable	String of printable characters which includes digits,letters,punctuation and whitespaces

- To see the contents of the string module, use the dir( ) with the module name as an argument

```
import string
print(dir(string))
```

```
==
['Formatter', 'Template', '_ChainMap', '_TemplateMetaclass', '__all__', '__
builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', '_re', '_string', 'ascii_letters', 'ascii_lowerc
ase', 'ascii_uppercase', 'capwords', 'digits', 'hexdigits', 'octdigits', 'p
rintable', 'punctuation', 'whitespace']
>>>
```

```
str="hello"
print(dir(str))
```

```
# program to dispaly the type of items in the string module
import string
print(string.digits)
print(string.ascii_letters)
print(string.ascii_lowercase)


 str="Python"
 print(help(str.isalpha))
```

```
Help on built-in function isalpha:

isalpha() method of builtins.str instance
    Return True if the string is an alphabetic string, False otherwise.

    A string is alphabetic if all characters in the string are alphabetic a
nd there
    is at least one character in the string.

None
```

For Details Contact Me @ :

9247448766

ravikanth27787@gmail.com

python

Programming Language