PYTHON PROGRAMMING LANGUAGE CS226 Semester-II

Unit-1

Strings:

A string is a group of characters or sequence of characters.

Create and initializing a string

Strings can be created by enclosing characters inside a single quote or double quotes. Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

Example 1:

```
>>'Hello'

'Hello'



>>"Hello"
'Hello'



>>'''Hello'''

'Hello'



>>>myfirst="SaveEarth"
>>>print

(myfirst) Save

Earth

>>>print("A friend in need isafriendindeed") A

 friend in need is a friend indeed


>>>my_string1 = raw_input("Enter a string")
```

```
>>>print(my_string1)


>>>my_string2 = """Hello, welcome to the world of Python"""

>>>print(my_string2)
```

Access elements in a string:

Individual characters in a string are accessed using the subscript[] operator. The expression in brackets is called an **index**. The index of the first character is 0 and that of the last character is n-1 where n is the number of characters in the string. Trying to access a character out of index range will raise an IndexError. Python allows negative indexing for its sequences.

String A	H	E	L	L	O
Positive Index	0	1	2	3	4
Negative Index	-5	-4	-3	-2	-1

Example 2:

A= “HELLO”

To access the first character of the string

```
>>>print

A[0] H
```

To access the fourth character of the string

```
>>>print

A[3] L
```

To access the last character of the string

```
>>print A[-1]

   O
```

To access the third last character of the string

```
>>printA[2]

L

>>print A[3]

L
```

Both indexing elements referring same elements.

Important points about accessing elements in the strings using subscripts

- Positive subscript helps in accessing the string from the beginning
- Negative subscript helps in accessing the string from the end.
- Subscript 0 or –ve n(where n is length of the string) displays the first element. Example : A[0] or A[-5+willdisplay“H”
- Subscript 1 or –ve (n-1) displays the second element.

Strings are immutable

Strings are immutable means that the contents of the string cannot be changed after it is

created. Let us understand the concept of immutability with help of an example.

Example 3

```
>>>str='honesty'

>>>str[2]='p'

TypeError: 'str' object does not support item assignment
```

Python does not allow the programmer to change a character in a string. As shown in the above example, strhasthevalue“honesty”.Anattempttoreplace“n”inthestringby“p”displays a TypeError.

Traversing a string

Traversing a string means accessing all the elements of the string one after the other by using the subscript. A string can be traversed using: for loop or while loop.

String traversal using for loop	String traversal using while loop
Program: A="Welcome" for i in A: print (i) Output: W e l c o m e	**Program:** A="Welcome" i=0 while (i<len(A)): print (A[i]) i=i+1 Output: W e l c o m e

Strings Operations

Operator	Description	Example
+ (Concatenation or Addition)	The + operator joins the text on both sides of the operator	Example4: >>> "Save"+"Earth" Save Earth Example 5:

		str1 = 'Hello'
		
		str2 ='World!'
		
		# using +
		
		print('str1 + str2 = ', str1 + str2)

*** (Repetition)**	The * operator repeats the string on the left hand side times the value on right hand side.	Example 6: >>>3*"SaveEarth" "Save Earth Save Earth Save Earth" Example 7: str= "Hello" print (3*str) Output: HelloHelloHello

in (Membership)	The operator displays 1 if the string contains the given character or the sequence of characters.	>>>A="Save Earth"
		
		>>> "S" in A
		
		True
		
		>>> "Save" in A
		
		True
		
		>> "SE" in A
		
		False

not in	The operator displays 1 if the string does not contain the given character or the sequence of characters. (working of this operator is the reverse of in operator discussed above)	>>>"SE" not in "Save Earth" True >>>"Save" not in "Save Earth" False
range (start, stop[, step])	This function is already discussed in previous chapter.	
Slice[n:m]	The Slice[n : m] operator extracts sub	>>>A="Save Earth"

	parts from the strings.	>>> print A[1:3] av The print statement prints the substring starting from subscript 1 and ending at subscript 3 but not including subscript 3

Slice Operation:

A substring of a string is called a slice. The slice operation is used to extract sub-parts of strings.

Example 8: Program to demonstrate slice operation on strings.

```
x=“PYTHON”
print (x[1:5])
# output:
YTHO print
(x[:6])

# output:

PYTHON print

(x[1:])

# output:

YTHON print

(x[:])

# output:

PYTHON print

(x[1:20])

#           output:

YTHON     print

(x[-1])

#  output:  N

print   (x[-6])

#  output:  P
```

```
print  (x[-2:])

#      output:

ON


print (x[:-2])

# output:

PYTH print

(x[-5:-2]) #

output: YTH
```

Slicing string stride:

In the slice operation, we 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 9:

```
name=“WelcometotheworldofPython” print
(name[2:10])

print (name[2:10:1])

print (name[2:10:2])

print (name[2:13:4])
```

Output:

```
lcome

to

lcome

to loet

le
```

Example 10:

```
test=“WelcometotheworldofPython” print

(test[::3])
```

Output:

```
WceohwloPh
```

Example 11:

st="WelcometotheworldofPython" print

(st[::-1])

Output:
nohtyP fo dlrow eh tot emocleW

Example 12:

A="WelcometotheworldofPython" print

(A[::-3])

Output:
nt   r ttml

Strings using relations operators:

Symbol/Operator	Description	Example
<	Less than	>>> "Hello"< "Goodbye" False >>>'Goodbye'< 'Hello' True
>	Greater than	>>>"Hello"> "Goodbye" True >>>'Goodbye'> 'Hello' False

<=	less than equal to	>>>"Hello"<= "Goodbye" False >>>'Goodbye' <= 'Hello' True
>=	greater than equal to	>>>"Hello">= "Goodbye" True >>>'Goodbye' >= 'Hello' False
! =	not equal to	>>>"Hello"!= "HELLO" True >>> "Hello" != "Hello" False
==	equal to	>>>"Hello" == "Hello" True >>>"Hello" == "Good Bye" False

Escape Sequence:

An escape sequences is nothing but a special character that has a specific function.

Escape sequence	Description	Example
\n	New line	>>> print "Hot\nCold" Hot Cold
\t	Tab space	>>>print "Hot\tCold" Hot Cold

\	Backslash	>>>print(“\”) \
\’	Single quote	>>>print(“\’ ”) ‘
\”	Double quote	>>>print ((“\””) “

Raw String:

If you want to specify a string that should not handle any escape sequences and want to display exactly as specified, then you need to specify that string as a raw string.

A Raw string is specified by prefixing r or R to the string.

Example 13:

```
>>>print(R“What\’s yourname?”)
What\’syourname?
```

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.

%s string conversion via str() prior to formatting

%d signed decimal integer

%f floating point real number

Example 14:

print "My name is %s and weight is %d kg!" % ('student', 45.256)

Output

My name is student and weight is 45 kg!

String Output formatting:

Sometimes we would like to format our output to make it look attractive. This can be done by using the str.format() method. This method is visible to any string object.

Example 15:

x = 5; y = 10

print ('The value of x is {} and y is

{}'.format(x,y)) Output:- The value of x is 5

and y is 10

Here the curly braces {} are used as placeholders. We can specify the order in which it is printed by using numbers (tuple index).

Example 15:

print('I love {0} and

{1}'.format('bread','butter')) # Output: I love

bread and butter

print('I love {1} and

{0}'.format('bread','butter')) # Output: I love

butter and bread

We can even use keyword arguments to format the string.

print('Hello {name}, {greeting}'.format(greeting = 'Goodmorning', name =

'John')) # Output: Hello John, Goodmorning

String Functions and Methods:

Syntax	Description	Example
len()	Returns the length of the string Syntax: len(string_variable)	>>>A="Save Earth" >>> print len(A) 10
capitalize()	Returns the exact copy of the string with the first letter in upper case Syntax: string_variable.capitalize( )	>>>str="welcome" >>>print str.capitalize() Welcome
find(sub[, start[, end]])	The function is used to search the first occurrence of the substring in the given string. It returns the index at which the substring starts. It returns -1 if the substring does occur in the string. Syntax: string_variable. find(substring,start,end)	>>>str='mammals' >>>str.find('ma') 0 On omitting the start parameters, the function starts the search from the beginning. >>>str.find('ma',2) 3 >>>str.find('ma',2,4) -1 Displays -1 because the substring could not be found between the index 2 and 4-1 >>>str.find('ma',2,5) 3

isalnum()	Returns True if the string contains only letters and digit. It returns False ,If the string contains any special character like _ , @,#,* etc. syntax:string_variable.isalnum( )	>>>str='Save Earth' >>>str.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.
	syntax: string_variable.isalpha()
		>>> 'Click123'.isalpha()
		
		False
		
		>>> 'python'.isalpha()
		
		True

isdigit()	Returns True if the string contains only numbers. Otherwise it returns False. Syntax: string_variable.isdigit()	>>>str="SAVE EARTH" >>>print str.isdigit() False Message= "007" print (Message.isdigit()) Output: True
lower()	Returns the exact copy of the string with all the letters in lowercase. syntax: string_variable.lower()	>>> str = "SAVE EARTH" >>>print str.lower() save earth
islower()	Returns True if the string is in lowercase syntax: string_variable.islower()	str="save earth" >>>print str.islower() True
isupper()	Returns True if the string is in uppercase.	str="save earth"

	syntax:string_variable.isupper()	>>> print str.isupper() False
upper()	Returns the exact copy of the string with all letters in uppercase. syntax: string_variable.upper()	>>> str= “welcome” >>>print str.upper() WELCOME
strip()	Removes all leading and trailing whitespaces in a string. syntax:string_variable.strip()	>>>str=” hello “ >>>str.strip() Hello

lstrip()	Returns the string after removing the space(s) on the left of the string.
	syntax:string.lstrip()
		>>> print str
		
		Save Earth
		
		>>>str.lstrip()
		
		'Save Earth'
		
		>>>str='Teach India Movement'
		
		>>> print str.lstrip("T")
		
		each India Movement
		
		>>> print str.lstrip("Te")
		
		ach India Movement
		
		>>> print str.lstrip("Pt")
		
		Teach India Movement
		
		If a string is passed as argument to the lstrip() function, it removes those characters from the left of the string

rstrip()	Returns the string after removing the space(s) on the right of the string. syntax:string_variable.rstrip()	>>>str='Teach India Movement‟ >>> print str.rstrip() Teach India Movement

isspace()	Returns True if the string contains only white spaces and False even if it contains one character. syntax:string_variable.isspace( )	>>> str=' '
		>>> print str.isspace()
		True
		>>> str='p'
		>>> print str.isspace()
		False

istitle()	Returns True if the string is title cased. Otherwise returns False syntax: string_variable.istitle()	>>> str='The Green Revolution'
		>>> str.istitle()
		True
		>>> str='The green revolution'
		>>> str.istitle()
		False

replace(old, new)	The function replaces all the occurrences of the old string with the new string syntax:string_variable.replace(old,new)	>>>str=‟hello”
		>>> print str.replace('l','%')
		he%%o
		>>> print str.replace('l','%%')
		he%%%%o

join (list)	Returns a string in which the string elements have been joined by a separator.
	Syntax: string_variable.join(list)
		>>> str1=('jan', 'feb' ,'mar')
		>>>str=‟&”
		>>> str.join(str1)
		'jan&feb&mar

swapcase()	Returns the string with case changes
	Syntax:string_variable.swapcase()
		>>> str='UPPER'
		>>> print str.swapcase()
		upper
		>>> str='lower'
		>>> print str.swapcase()

		LOWER

split([sep[, maxsplit]])	The function splits the string into substrings using the separator. The second argument is optional and its default value is zero. If an integer value N is given for the second argument, the string is split in N+1 strings.
	
	Syntax: string_variable.split(substring,value)
		>>>str='The$earth$is$what$we$all $have$in$common.'
		
		>>> str.split($,3)
		
		SyntaxError: invalid syntax
		
		>>> str.split('$',3)
		
		['The', 'earth', 'is', 'what$we$all$have$in$common.']
		>>> str.split('$')
		
		['The', 'earth', 'is', 'what', 'we', 'all', 'have', 'in', 'common.']
		
		>>> str.split('e')
		
		['Th', ' Gr', '', 'n R', 'volution']
		
		>>> str.split('e',2)
		
		['Th', ' Gr', 'en Revolution']

partition(sep)	The function partitions the strings at the first occurrence of separator, and returns the strings partition in three parts i.e. before the separator, the separator itself, and the part after the separator. If the separator is not found, returns the string itself, followed by two empty strings
	syntax:string_variable.partition (substring)
		>>>str='The Green Revolution'
		
		>>> str.partition('Rev')
		
		('The Green ', 'Rev', 'olution')
		
		>>> str.partition('pe')
		
		('The Green Revolution', '', '')
		
		>>> str.partition('e')
		
		('Th', 'e', ' Green Revolution')

count(str,beg,end)	Count number of times str occurs in a string. You can specify be 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. syntax: string_variable.count(str,beg,en	>>>str="he" >>>message="helloworldhellohello" >>>message.count(str,0,len(message)) 3

	d)	
find(str,beg,end)	Checks if str is present in the string. If found it returns the position at which str occurs in a 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. Syntax:string_variable.find(str, beg,end)	>>>message= "she is my best friend" >>>message.find("my",0,len(message)) 7 >>>message.find("no",0,len(message)) -1
index(str,beg,end)	Same as find but raises an exception if str is not found Syntax:string_variable.index(substring, beg, end)	>>>message= "she is my best friend" >>>message.index("mine",0,len(message)) valueError: substring not found
rfind(str,beg,end)	Same as find but starts searching from the end. Syntax: string_variable.rfind(str,beg,end)	>>>str= "Is this your bag?" >>>str.rfind("is",0,len(str)) 5
rindex(str,beg,end)	Same as index but start searching from the end and raises an exception if str is not found. Syntax: string_variable.rindex(str,beg,end)	>>>str= "Is this your bag?" >>>str.rindex("you",0,len(str)) 8
encode(encoding,errors)	This function is used to encode the string using the provided encoding. Syntax: string_variable.encode(encoding, errors)	>>str = "this is string example. ....wow!!!"; >>str.encode('base64','strict') dGhpcyBpcyBzdHJpbmcgZXhhbXBsZS4uLi53b3chISE=

decode(encoding,errors)	This function is used decode the encoded string. syntax: string_variable.decode(encoding,errors)	>>str = "this is string example. ....wow!!!"; >>str=str.encode('base64','strict') >>str.decode('base64','strict') this is string example. ....wow!!!" ;

String Constants:

The string module consists of a number of useful constants, classes and functions. These functions are used to manipulate strings.

String constants some constants defined in the string module are:

1) **string.ascii_letters:** The command/function displays a string containing uppercase characters and lowercase characters.

Example 16:

```
>>>import string

>>>print (string.ascii_letters)

'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRST

UVWXYZ'
```

2) **string.ascii_uppercase:** The command/function displays a string containing uppercase characters.

Example 17:

```
>>>import string

>>>print (string.ascii_uppercase)

'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

3) **string.ascii_lowercase:** The command displays a string containing all lowercase characters.

Example 18:

```
>>>import string

>>>print (string.ascii_lowercase)

'abcdefghijklmnopqrstuvwxyz'
```

4) **string.ascii_digits:** The command displays a string containing digits.

Example 19:

```
>>>import string

>>>print

(string.ascii_digits)

'0123456789'
```

5) **string.hexdigits:** The command displays a string containing hexadecimal characters.

Example 20:

```
>>>import string

>>>print (string.hexdigits)

'0123456789abcdefABCDEF'
```

6) string.octdigits: The command displays a string containing octal characters.

Example 21:

```
>>>import string

>>>print

(string.octdigits)

'01234567'
```

7) string.punctuations: The command displays a string containing all the punctuation characters.

Example 22:

```
>>>import string

>>>print (string.punctuations)

'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}-

'
```

8) string.whitespace: The command displays a string containing all ASCII characters that are considered whitespace. This includes the characters space, tab, linefeed, return, formfeed, and vertical tab.

Example 23:

```
>>>import string

>>>print

(string.whitespace)

'\t\n\x0b\x0c\r’
```

9) string.printable: The command displays a string containing all characters which are considered printable like letters, digits, punctuations and whitespaces.

Example 23:

```
>>>import string

>>>print (string.printable)

'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!

"#$%&\'()*+,-
./:;<=>?@[\\]^_`{|}- \t\n\r\x0b\x0c'
```

Exercise:

1. Write a Python program to copy the string using for loop(using range and without using range function).
2. Write a program to print the following pattern. A
   AB
   ABC
   ABCD
   E
   ABCDEF
3. Writeapythonthattakesuser'snameandPANcardnumberasinput.Validatetheinformationusing is X function and print the details.
4. Write a program that encrypts a message by adding a key value to every character. Hint: say, if key=3, then add 3 to every character
5. Write a program to that uses split() to split multiline string.
6. Write a program that accepts a string from user and redisplays the same string after removing vowels from it.
7. Write a program that finds whether a given character is present in a string or not. In case it is present it prints the index at which it is present. Do not use built-in find functions to search the character.
8. Write a program that finds a character in a string from ending of string to beginning that like rfind function.
9. Write a python program that count the occurrences of a character in a string and also counting from the specified location . Do not use string count function.
10. Write a program to reverse of string.
11. Write a program to find whether the given string is palindrome string or not.

12. Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.
    Sample String : 'restart' Expected Result : 'resta$t

13. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

14. Write a Python program to remove the characters which have odd index values of a given string.
15. Write a Python script that takes input from the user and displays that input back in upper and lower cases.
    Sample Output: What's your favourite language?
    english My favourite language is ENGLISH
    My favourite language is english
16. Write a Python function to reverse a string if it's length is a multiple of 4.
17. Write a Python program to count and display the vowels of a given text.
18. Write a Python program to lowercase first n characters in a string.
19. Write a Python program to reverse words in a string.
20. Write a Python program to count occurrences of a substring in a string.

Assignment –I

1. Write any 20 Functions from strings with syntax and example?
2. Write short notes on String Constants?