PYTHON STRINGS & LISTS

- Python string is a built-in type text sequence.
- It is used to handle textual data in python.
- Creating Strings are simplest and easy to use in Python.
- We can simply create Python String by enclosing a text in single as well as
- double quotes.
- Python treat both single and double quotes statements same.

- Create >>>Str1=”hello” >>>Str2=”world”
- string operation Command Description
- 1.concatenation >>>str1+str2 “helloworld”
- 2.repetition >>>str1*2 ‘hellohello’
- #repeating the string
- 3.slice >>>str1[1] ‘e’
- #str1=”hello” print the 2nd character of hello ie…here [h refers to 0 and 1st character; e refers
  to 1 and 2nd character]
- 4.range slice >>>str2[2:] ‘rld’
- #str2[2:] means print the characters of string except first three
- 5.range slice >>>str2[0:1] ‘w’
- 6.membership[in] >>>’g’ in str1 True
- 7.membership[not in] >>>’h’ not in str2 False

String functions & methods

	Method	Description
1	Python String capitalize()	Converts first character to Capital Letter
2	Python String count()	returns occurrences of substring in string
3	Python String isalnum()	Checks Alphanumeric Character
4	Python String isalpha()	Checks if All Characters are Alphabets
5	Python String isdigit()	Checks Digit Characters
6	Python String islower()	Checks if all Alphabets in a String are Lowercase
7	Python String isspace()	Checks Whitespace Characters
8	Python String istitle()	Checks for Title cased String
9	Python String isupper()	returns if all characters are uppercase characters
10	Python String join()	Returns a Concatenated String

11	**Python String lower()**	**returns lowercased string**
12	Python String upper()	returns uppercased string
13	Python String swapcase()	swap uppercase characters to lowercase; vice versa
14	Python String partition()	Returns a Tuple
15	Python String replace()	Replaces Substring Inside
16	Python String split()	Splits String from Left
17	Python len()	Returns Length of an Object
18	Python max()	returns largest element
19	Python min()	returns smallest element

Python String capitalize()

Python String capitalize()

In Python, the capitalize() method converts first character of a string to uppercase letter and lowercases all other characters, if any.

The syntax of capitalize()is:

```
string.capitalize()
```

Capitalize a Sentence

```
string = "python is AWesome."
capitalized_string = string.capitalize()
print('Old String: ', string)
print('Capitalized String:', capitalized_string)
```

Out Put:

```
Old String:  python is AWesome.
Capitalized String: Python is awesome.
```

Python String count()

The string count() method returns the number of occurrences of a substring in the given string.

Substring - string whose count is to be found.

The syntax of count() method is:

string.count(substring, start=..., end=...)

Example : Count number of occurrences of a given substring

```
# define string
string = "Python is awesome, isn't it?"
substring = "is"
count = string.count(substring)
# print count
print("The count is:", count)
```

Output:

The count is: 2

Python String isalpha()

- **The isalpha() method returns True if all characters in the string are alphabets. If not, it returns False.**

Return Value from isalpha()

The isalpha() returns:

True if all characters in the string are alphabets (can be both lowercase and uppercase).

False if at least one character is not alphabet.

The syntax of isalpha() is: string.isalpha()

Example 1: Working of isalpha()

```
name = "Monica"
print(name.isalpha())
# contains whitespace
name = "Monica Geller"
print(name.isalpha())
 # contains number
name = "Mo3nicaGell22er"
print(name.isalpha())
```

Out Put:

True

False

False

Python String isdigit()

The isdigit() method returns True if all characters in a string are digits. If not, it returns False.

Return Value from isdigit()

The isdigit() returns:

True if all characters in the string are digits.

False if at least one character is not a digit.

Example : Working of isdigit()

```
s = "28212"
print(s.isdigit())
# contains alphabets and spaces
s = "Mo3 nicaG el l22er"
print(s.isdigit())
```

Output:

```
True
False
```

Python String islower()

The islower() method returns True if all alphabets in a string are lowercase alphabets. If the string contains at least one uppercase alphabet, it returns False.

The syntax of islower() is: string.islower()

The islower() method returns:

True if all alphabets that exist in the string are lowercase alphabets.
False if the string contains at least one uppercase alphabet.

Example 1: Return Value from islower()

```
s = 'this is good'
print(s.islower())
s = 'th!s is also good'
print(s.islower())
s = 'this is Not good'
print(s.islower())
```

Output:

```
True
True
False
```

Python String isspace()

Syntax:()

The isspace() method returns:

True if all characters in the string are whitespace characters

False if the string is empty or contains at least one non-printable() character

Example 1: Working of isspace()

```
s = '    \t'
print(s.isspace())
s = ' a '
print(s.isspace())
s = ''
print(s.isspace())
```

Output

```
True
False
False
```

Python String istitle()

The istitle() returns True if the string is a titlecased string. If not, it returns False.

The syntax of istitle() method is: string.istitle()

The istitle() method returns:

True if the string is a titlecased string

False if the string is not a titlecased string or an empty string

Example Program

```
s = 'Python Is Good.'
print(s.istitle())
s = 'Python is good'
print(s.istitle())
s = 'This Is @ Symbol.'
print(s.istitle())
s = '99 Is A Number'
print(s.istitle())
s = 'PYTHON'
print(s.istitle())
```

Output:

```
True
False
True
False
```

Python String upper()

The string upper() method converts all lowercase characters in a string into uppercase characters and returns it.

The syntax of upper() method is:

string.upper()

```
# example string
string = "this should be uppercase!"
print(string.upper())

# string with numbers
# all alphabets whould be lowercase
string = "Th!s Sh0uLd B3 uPp3rCas3!"
print(string.upper())
```

OUTPUT

```
THIS SHOULD BE UPPERCASE!
TH!S SH0ULD B3 UPP3RCAS3!
```

Python String partition()

The partition() method splits the string at the first occurrence of the argument string and returns a tuple containing the part the before separator, argument string and the part after the separator.

The syntax of partition() is: string.partition(separator)

The partition method returns a 3-tuple containing:

- the part before the separator, separator parameter, and the part after the separator if the separator parameter is found in the string
- string itself and two empty strings if the separator parameter is not found

program:

```
string = "Python is fun"
# 'is' separator is found
print(string.partition('is '))
# 'not' separator is not found
print(string.partition('not '))
string = "Python is fun, isn't it"
# splits at first occurence of 'is'
print(string.partition('is'))
```

output

```
('Python ', 'is ', 'fun')
('Python is fun', '', '')
('Python ', 'is', " fun, isn't it")
```

Python String join()

The join() is a string method which returns a string concatenated with the elements of an iterable.

The syntax of join() is:

string.join(iterable)

Example program

```
numList = ['1', '2', '3', '4']
seperator = ', '
print(seperator.join(numList))
numTuple = ('1', '2', '3', '4')
print(seperator.join(numTuple))
s1 = 'abc'
s2 = '123'
""" Each character of s2 is concatenated to the front of s1"""
print('s1.join(s2):', s1.join(s2))
""" Each character of s1 is concatenated to the front of s2"""
print('s2.join(s1):', s2.join(s1))
```

output:

```
1, 2, 3, 4
1, 2, 3, 4
s1.join(s2): 1abc2abc3
s2.join(s1): a123b123c
```

Python String replace()

The replace() method returns a copy of the string where all occurrences of a substring is replaced with another substring.

replace() parameters

The replace() method can take maximum of 3 parameters:

old - old substring you want to replace

new - new substring which would replace the old substring

count (optional) - the number of times you want to replace the old substring with the new substring

If count is not specified, replace() method replaces all occurrences of the old substring with the new substring.

Example Program

```
song = 'cold, cold heart'
print (song.replace('cold', 'hurt'))
song = 'Let it be, let it be, let it be, let it be'
'''only two occurences of 'let' is replaced'''
print(song.replace('let', "don't let", 2))
```

output:

```
hurt, hurt heart
Let it be, don't let it be, don't let it be, let it be
```

Python String split()

The split() method breaks up a string at the specified separator and returns a list of strings.

The syntax of split() is: str.split([separator [, maxsplit]])

split() Parameters

The split() method takes maximum of 2 parameters:

•**separator** (optional)- The is a delimiter. The string splits at the specified separator.

If the separator is not specified, any whitespace (space, newline etc.) string is a separator.

•**maxsplit** (optional) - The maxsplit defines the maximum number of splits.

The default value of maxsplit is -1, meaning, no limit on the number of splits.

Example Program

```
text= 'Love thy neighbor'
# splits at space
print(text.split())
grocery = 'Milk, Chicken, Bread'
# splits at ','
print(grocery.split(', '))
# Splitting at ':'
print(grocery.split(':'))
```

Output:

```
['Love', 'thy', 'neighbor']
['Milk', 'Chicken', 'Bread']
['Milk, Chicken, Bread']
```

Python String swapcase()

- **The string swapcase() method converts all uppercase characters to lowercase and all lowercase characters to uppercase characters of the given string, and returns it.**

The format of swapcase() method is:

string.swapcase()

Note: Not necessarily, string.swapcase().swapcase() == string

```
# example string
string = "THIS SHOULD ALL BE LOWERCASE."
print(string.swapcase())
string = "this should all be uppercase."
print(string.swapcase())
string = "ThIs ShOuLd Be MiXeD cAsEd."
print(string.swapcase())
```

Output:

```
this should all be lowercase.
THIS SHOULD ALL BE UPPERCASE.
tHiS sHoUlD bE mIxEd CaSeD.
```

Python String lower()

The string lower() method converts all uppercase characters in a string into lowercase characters and returns it.

The syntax of lower() method is:

string.lower()

```
# example string
string = "THIS SHOULD BE LOWERCASE!"
print(string.lower())
# string with numbers
# all alphabets whould be lowercase
string = "Th!s Sh0uLd B3 L0w3rCas3!"
print(string.lower())
```

ouput

```
this should be lowercase!
th!s sh0uld b3 l0w3rcas3!
```

Python Lists

- List is an ordered sequence of items.
- A list is a collection which is ordered and changeable.
- It is one of the most used datatype in Python and is very flexible.
- All the items in a list do not need to be of the same type.
- A list can be composed by storing a sequence of different type of values separated by
- commas.
- Python list is enclosed between square [] brackets and elements are stored in the index
- basis with starting index 0.

Example:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Output:

```
['apple', 'banana', 'cherry']
```

Characteristics of List.

- 1.Mutable
- 2.Linear Data structure
- 3.Mixed type Elements
- 4.Variable Length
- 5.Zero Based Indexing

	Create >>>List=[2,4,"grapes"]	Create >>>List2=[2,5,3,8,4]	
	List Operation	**Command**	**Description**
1	Replace	>>>list1[2]="mango"	(2,4,"mango")
2	insert	>>>list.insert(1,"orange")	(2,"orange",4,"mango")
3	sort	>>>list2.sort()	(2,3,4,5,8)

4	reverse	>>>list2.reverse()	(8,5,4,3,2)
5	append	>>>list2.append("guava")	(8,5,4,3,2,"guava")
6	delete	>>>del list2[5]	(8,5,4,3,2)
7	concatenation	>>>list1+list2	(2,"orange",4,"mango",8,5,4,3,2)
8	nested list example : >>>list3=[(2,1),(4,8),(5,6)]		
9	delete	>>>del list3[2]	[(2,1),(4,8)]
10	Pop	>>>list2.pop()	(8,5,4,3,2)
11	Pop with index	>>>list2.pop(1)	(8,4,3,2)

Method	Description
Python List append()	Add Single Element to The List
Python List extend()	Add Elements of a List to Another List
Python List insert()	Inserts Element to The List
Python List remove()	Removes Element from the List
Python List index()	returns smallest index of element in list
Python List count()	returns occurrences of element in a list
Python List pop()	Removes Element at Given Index
Python List reverse()	Reverses a List
Python List sort()	sorts elements of a list
Python List clear()	Removes all Items from the List
Python len()	Returns Length of an Object
Python max()	returns largest element
Python min()	returns smallest element

Thank you

Prepared by

Ramesh.Yajjala, Assistant Professor(c)

Dept of CSE,IIIT-Srikakulam, RGUKT-AP