CBSE: Class: XII Computer Science

UNIT-1 CHAPTER-1: String Functions

String functions: String functions are available in python standard module. These are always available to use. Import statement is not required for using these functions.

Sr. No	Function with Example

1	capitalize(void): This function Converts the first character of string to upper case and returns complete string with first letter upper case.
	
	Syntax: String.capitalize()
	
	Example
	str="this is string function"
	str.capitalize()
	print(str.capitalize())
	
	#or
	str="python is an oop"
	result=str.capitalize()
	print(result)
	
	Output: Python is an oop

2	casefold(void): This function Converts the all characters of string to lower case and returns complete string in lower case. Syntax: String.casefold() Example str="RAVI IS LEARNING PYTHON" print(str.casefold()) str="RAVI IS LEARNING JAVA" result=str.casefold() print(result) Output: ravi is learning python ravi is learning java
3	Center(): Python center() method alligns string to the center by filling paddings left and right of the string. This method takes two parameters, first is a width and second is a fillchar which is optional. The fillchar is a character which is used to fill left and right padding of the string. Syntax: string.center(width, fillchar) width= Padding space based on pixel fillchar=character to be filled Example: str = "Hello DPS Sonepat" str2 = str.center(30,'#') print("Old value:", str) print("New value:", str2)

	Output: Old value: Hello DPS Sonepat New value: ######Hello DPS Sonepat#######
4	**Count():** This function returns the number of occurrences of substring in the specified range. It takes three parameters, first is a substring, second a start index and third is last index of the range. Start and end both are optional whereas substring is required. Syntax: count(sub, start, end) ○ **sub** (required) Sub string ○ **start** (optional) starting position of main string ○ **end** (optional) end position of main string **Example:** str = "We are learning one of the best programming language" str2 = str.count('o') print("occurences1:", str2) str2 = str.count('e') print("occurences2:", str2) str2 = str.count('e',6) print("occurences3:", str2) str2 = str.count('e',1,6) print("occurences4:", str2) Output: occurences1: 3 occurences2: 7 occurences3: 5 occurences4: 2

5	encode(): This Python method encodes the string according to the provided encoding standard. By default Python strings are in unicode form but can be encoded to other standards also.
	
	Encoding is a process of converting text from one standard code to another.
	
	Syntax: encode(encoding="utf-8", errors="strict")
	Parameters
	encoding : encoding standard, default is UTF-8
	errors : errors mode to ignore or replace the error messages.
	
	Examples:
	str = "HELLO"
	encode = str.encode()
	# Displaying result
	print("Old value", str)
	print("Encoded value", encode)
	Output:
	Old value HELLO
	Encoded value b 'HELLO'
	--- 	
	str = "HËLLO"
	encode = str.encode()

	print("Old value", str) print("Encoded value", encode) Output: Old value HËLLO Encoded value b'H\xc3\x8bLLO' --- Encoding latin character into ascii, it throws an error. See the example below str = "HËLLO" encode = str.encode("ascii") print("Old value", str) print("Encoded value", encode) Output: UnicodeEncodeError: 'ascii' codec can't encode character '\xcb' in position 1: ordinal not in range(128) If we want to ignore errors, pass ignore as the second parameter. str = "HËLLO" encode = str.encode("ascii","ignore") print("Old value", str) print("Encoded value", encode) Output: Old value HËLLO Encoded value b'HLLO' Python String Encode() Method Example 5 It ignores error and replace character with ? mark. --- str = "HËLLO" encode = str.encode("ascii","replace") print("Old value", str) print("Encoded value", encode) Output: Old value HËLLO Encoded value b'H?LLO'
6	endswith() : Python **endswith()** method returns **true** if the string ends with the specified substring, otherwise returns **false**. Syntax: endswith(suffix, start, end) • **suffix** : a substring • **start** : start index of a range • **end** : last index of the range Example:1 str = "Hello this is javatpoint." isends = str.endswith(".") print(isends) Output: True

	Example 2 It returns false because string does not end with is. str = "Hello this is javatpoint." isends = str.endswith("is") print(isends) Output: False Example 3 Here, we are providing start index of the range from where method starts searching. str = "Hello this is javatpoint." isends = str.endswith("is",10) print(isends) Output: False Example 4 It returns true because third parameter stopped the method at index 13. str = "Hello this is javatpoint." isends = str.endswith("is",0,13) print(isends) Output:True
7	find() : Python find() method finds substring in the whole string and returns index of the first match. It **returns -1** if substring does not match. **Syntax:** find(sub, start, end) Parameters sub : substring start : start index a range end : last index of the range Return Type: If found, it returns index of the substring, otherwise -1 Example 1 An example of simple find method which takes only single parameter (a substring). str = "Welcome to the Python." str2 = str.find("the") print(str2) Output:11 Example 2 It returns -1 if not found any match, see the example. str = "Welcome to the Python." str2 = str.find("is") print(str2) Output:-1 Example 3 Let's specify other parameters too and makes search more customize. str = "Welcome to the Python." str2 = str.find("t") str3 = str.find("t",25) print(str2) print(str3)

	Output: 8 -1
8	Format(): Python format() method is used to perform format operations on string. While formatting string a delimiter {} (braces) is used to replace it with the value. This delimeter either can contain index or positional argument. Syntax: format(*args, **kwargs) Parameters *args : substring **kwargs : start index a range Return Type: It returns a formatted string. Example 1 An example of simple format method which format string using positional delimiter. str = "Java" str2 = "C#" str3 = "{} and {} both are programming languages".format(str,str2) print(str3) Output: Java and C# both are programming languages Example 2 The delimiter (braces) are using numerical index to replace and format string. str = "Java" str2 = "C#" str3 = "{1} and {0} both are programming languages".format(str,str2) print(str3) Output: C# and Java both are programming languages Example 3 Formatting numerical value in different-different number systems. See the below example. val = 10 print("decimal: {:d}".format(val)); # display decimal result print("hex: {:x}".format(val)); # display hexadecimal result print("octal: {:o}".format(val)); # display octal result print("binary: {:b}".format(val)); # display binary result Output: decimal: 10 hex: a octal: 12 binary: 1010 Example 4 Formating float and percentile in string is pretty easy. val = 100000000 print("decimal: {:,}".format(val)); # formatting float value print("decimal: {:.2}".format(56/9)); # formatting percentile value Output: decimal: 100,000,000 decimal: 6.2
9	index() : Python index() method is same as the find() method except it returns error on failure. This method returns index of first occurred substring and an error if there is no match found.

	Syntax: index(sub, start, end) sub : substring start : start index a range end : last index of the range Return Type: If found it returns an index of the substring, otherwise an error ValueError. Example 1 str = "Welcome to the Python." str2 = str.index("Py") print(str2) Output: 15 Example 2 An error is thrown if the substring is not found. str = "Welcome to the Python." str2 = str.index("ate") print(str2) Output: ValueError: substring not found Example 3 str = "Welcome to the Python." str2 = str.index("P",15,20) print("p is present at :",str2,"index") Output: P is present at : 15 index

10	Join(): Python join() method is used to concat a string with iterable object. It returns a new string which is the concatenation of the strings in iterable. It throws an exception TypeError if iterable contains any non-string value.It allows various iterables like: List, Tuple, String etc.
	
	Syntax: join(iterable)
	Parameters: iterable - iterable object like: List, Tuple, String etc.
	Return: It returns a new string or an exception TypeError if iterable contains any non-string value.
	Example 1
	A simple example which implements join() method with the List iterable, see the example below.
	str = ":" # string
	list = ['1','2','3'] # iterable
	str2 = str.join(list)
	print(str2)
	Output:1:2:3
	
	Example 2
	A list iterable join with empty string and produce a new string, see the example below
	str = "" # string
	list = ['P','y','t','h','o','n'] # iterable
	str2 = str.join(list)
	print(str2)
	Output: Python
	
	Example 3
	An example of join() method with Set iterable. Set contains unordered elements and produce different output each times. See the example below.
	str = "->" # string

	list = {'Java','C#','Python'} # iterable
	str2 = str.join(list)
	print(str2)
	Output:
	Java->Python->C#
	
	Example 4
	In case of dictionary, this method join keys only. Make sure keys are string, otherwise it throws an exception.
	dic = {'key1': 1, 'key2': 2}
	str = '&'
	str = str.join(dic)
	# Displaying result
	print(str)
	Output: key1&key2