Collections

Python String

A string is a sequence of characters. A character is simply a symbol. For example, the English language has 26 characters.Computers do not deal with characters, they deal with numbers (binary). Even though you may see characters on your screen, internally it is stored and manipulated as a combination of 0's and 1's.

This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encoding used. In Python, a string is a sequence of Unicode characters. Unicode was introduced to include every character in all languages and bring uniformity in encoding. You can learn more about Unicode from here.

One of the most common data type of python is String. "str" is the built in string class of python. String liberals can be enclosed by single or double quotes.

Python accepts single ('), double (") and triple ('") quotes to denote 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.

String Quotes	Example
Single quotes	`'Welcome to Amravati'`
Double quotes	`"Welcome to Amravati "`

Triple quotes	`'Welcome to Amravati"'`
	
	`"“”Welcome to Amravati““"`

String Operators:

Operator	Description
+	The + operator concatenates strings. It returns a string consisting of the operands joined together
*	Repetition - Creates new strings, concatenating multiple copies of the same string.
In	`Python aso provdes a membershpoperator thatcan be used with srings. The in` operator returns True if the first operand is contained within the second and False

Example:

Program	Output
a=”Welcome b=”to ” c=" Amravati " msg=a+b+c print(msg)	Welcome to Amravati

Example:

Program	Output
a="CC" msg=a*3 print(msg)	CCCCCC

Example:

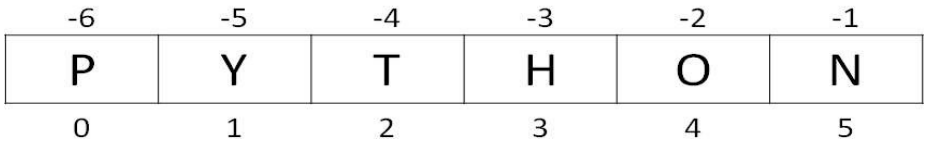
Program	Output
a=”Welcome to Amravati” msg=” Amravati” in a print(msg)	True

String Indexing:

In Python strings are ordered sequences of character data, and thus can be indexed in this way. Individual characters in a string can be accessed by specifying the string name followed by a number in square brackets.

Strinp indexing in Python is zero-based: the first character in the strinp has index 0, the next has index J, and so on. The index of the last character will be the length of the string minus one.

String indices can also be specified with negative numbers, in which case indexing occurs from the end of the string backward



Operator	Description
[1ndex]	Slice - Gives the character from the given index
[startindex endindex]	RangeSce-Givesthecharactersrornthegivenrange

Example:

Program

```
a=”Welcome to Amravati”
print(a[3])
```

Output

```
c
```

Example:

Program

```
a=”Welcome to Amravati”


print(a[5:])
print(a[:10])
print(a[5:12])
```

Output

```
me to Amra
Welcome to
me to A
```

String Functions:

Python provides many functions that are built-in that work with strinp.

Function	Description
len(object)	Returnsthelengthofastrng
str (object)	Returns a string representation of an object

Example:

Program

```
a=”Welcome to Amravati”
l=len(a)
print(”Length of String ”,l)
```

Output

```
Length of String  15
```

String Methods:

Python String provides many methods that are built-in that work with string.

Case Conversion:

Methods in this proup perform case conversion on the target strinp.

Function	Description
s.capitalize()	s.capitalize() returns a copy of s with the first character converted to uppercase and all other characters converted to lowercase.
s.lower()	s.lower() returns a copy of s with all alphabetic characters converted to lowercase.
s.swapcase()	s.swapcase() returns a copy of s with uppercase alphabetic characters converted to lowercase and vice versa
s.title()	s.title() returns a copy of s in which the first letter of each word is converted to uppercase and remaining letters are lowercase
s. upper()	s.upper() returns a copy of s with all alphabetic characters converted to uppercase.

Example:

Program	Output
a=”Welcome to Amravati” print(a.capitalize()) print(a.lower()) print(a.swapcase()) print(a.title()) **print(a .upper())**	#capitalize Welcome to Amravati #1ower welcome to Amravati #swapcase wELCOME TO aMRAVATI #title Welcome To Amravati #upper WELCOME TO AMRAVATI

Find and Replace:

These methods provide various means of searchinp the target strinp for a specified substrinp.

Function	Description
s.count(sub,start,end)	s.count(sub>) returns the number of non-overlapping occurrences of substring.
s.endswith(suffix,start,end)	s.endswith(suffixe) returns True if s ends with the specified <suffix> and False otherwise
s.find(sub,start,end)	.find() to see if a Python string contains a particular substring. s.find(sub>) returns the lowest index in s where substring <sub> ïs cou üô.
s. **índex(sub,** start, **end)**	This method is identical to .find(), except that it raises an exception if <sub> is not found rather than returning -J
s.rfind(sub,start,end)	s.rfind(sub>) returns the highest index in s where substring <sub> is found
s.rindex(sub,start,end)	This method is identical to .rfind(), except that it raises an exception if sub> is not found rather than returning -1
s.startswith(prefix,start,end)	.startswith() method, s.startswith(<su ix>) returns True if s starts with the specified <suñix> and False otherwise

Example:

Program	Output
a=”Welcome to Amravati” print(a.count('C')) print(a.endswith('C')) print(a.find('c')) print(a.index('A')) print(a.rfind('m')) print(a.rindex('A')) print(a.startswith('C'))	#count 1 #endswith False #find 4 #index 11 #rf1nd 12 #rindex 12 #startswith False

Character Classification:

Methods in this proup classify a strinp based on the characters it contains.

Function	Description
s.isalnum()	s.isalnum() returns True if s is nonempty and all its characters are alphanumeric (either a letter or a number), and False otherwise.
s.isalpha()	s.isalpha() returns True if s is nonempty and all its characters are alphabetic, and False otherwise
s.isdigit()	.isdigit() Python method to check if your string is made of only digits. s.digit() returns True if s is nonempty and all its characters are numeric digits, and False otherwise
s.isupper()	s.isupper() returns True if s is nonempty and all the alphabetic characters it contains are uppercase, and False otherwise.
s.islower()	s.islower() returns True if s is nonempty and all the alphabetic characters it contains are lowercase, and False otherwise.
s.isspace()	s.isspace() returns True if s is nonempty and all characters are whitespace characters, and False otherwise.

Example:

Program	Output
print(”CCIT”.isalnum())	#isalnum True
print(”CCIT”.isalpha())	#isalpha True
print(”CCIT”.isdigit())	#isdigit False
print(”CCIT”.isupper())	#isupper True
print(”CCIT”.islower())	#islower False
print(”CCIT”.isspace())	**#isspace** False

String Formatting:

Methods in this proup modify or enhance the format of a strinp

Function	Description
s.center(width,fill)	s.center(<width>) returns a string consisting of s centered in a field of width <width>
s.expandtabs(tabsize=8)	s.expandtabs() replaces each tab character ('\t') with spaces. By default, spaces are filled in assuming a tab stop at every eighth column
s.ljust(width,fill)	s.ljust(<width>) returns a string consisting of s left-justified in a field of width <width>.
s.lstrip(chars)	s.lstrip() returns a copy of s with any whitespace characters removed from the left end.
s.replace(old,new,count)	.replace() method. s.replace(<old>, <new>) returns a copy of s with all occurrences of substring <old> replaced by <new>
s.rjust(width,fill)	s.must(<width>) returns a string consisting of s right-justified in a field of width <width>.
s.rstrip(chars)	s.rstrip() returns a copy of s with any whitespace characters removed from the right end

s.strip(chars)	s.strip() is essentially equivalent to invoking s.lstrip() and s.rstrip() in succession. Without the <chars> argument, it removes leading and trailing whitespace
s.zfill(width)	s.zfill(<width>) returns a copy of s left-padded with '0' characters to the specified <width>

Example:

Program

```
a="CCIT"

print(a.center(10,”-”))

print(a.expandtabs(tabsize=8))

print(a.ljust(10,”-”))

print(a.lstrip('C'))

print(a.replace(a,”Python“))

print(a.rjust(10,”-“))

print(a.rstrip(”T”))

print(a.strip(“C”))

print(a.zfill(10))
```

Output

```
#center
 - --CCIT-- -

#expandtabs
       CCIT

#ljust
CCIT------

#lstrip
IT

#rep1ace
Python

#rjust
- - - - - -CCIT

#rstrip
CCI

#str1p
IT

#zfill
eeeeeeccT
```

Converting Between Strings and Lists:

Methods in this group convert between a string and some composite data type by either pasting objects together to make a string, or by breaking a string up into pieces.

Function	Description
s.join(iterable)	s.join(iterable>) returns the string that results from concatenating the objects in iterable > separated by s.
s.partition(<sep›)	s.partition(sep) splits s at the first occurrence of string <sep>. The return value is a three-part tuple consisting of: • The portion of s preceding <sep> • <sep> itself • The portion of s following <sep>
s.rpartition(‹sep›)	s.rpartition(sep>) functions exactly like s.partition(<sep>), except that s is split at the last occurrence of <sep> instead of the first occurrence.
s.rsplit(sep=None, maxsplit=-1)	Without arguments, s.rsplit() splits s into substrings delimited by any sequence of whitespace and returns the substrings as a listIf <sep> is specified, it is used as the delimiter for splitting
s.split(sep=None, maxsplit=-1)	s.split() behaves exactly like s.rsplit(), except that if maxsplit is specified, splits are counted from the left end of s rather than the right end

Example:

Program

```
print(”*".join(['Python','Flask',”Django”]))

msg=”Welcome to Amravati”

print(msg.partition(” ”))

print(msg.rpartition(" “))

print(msg.rsplit(” ”))

print(msg.split(” ”))
```

Output

```
#join
Python*Flask*Django

#partition
('Welcome', ' ', 'to Amravati
')

#rpartition
('Welcome to', ' ',  Amravati')

#rsplit
['Welcome', 'to', ' Amravati ']

#split
['Welcome', 'to', ' Amravati ']
```