RAJALAKSHMI ENGINEERING COLLEGE

OCS1903 - Programming using Python

Strings

Week-5

CONTENTS

- String-Data Structure
- String Creation
- String Indexing
- String Operators
- String Functions
- String Methods
- Problem Solving

String

Eg :"Vijaya Kumar"
'rec'

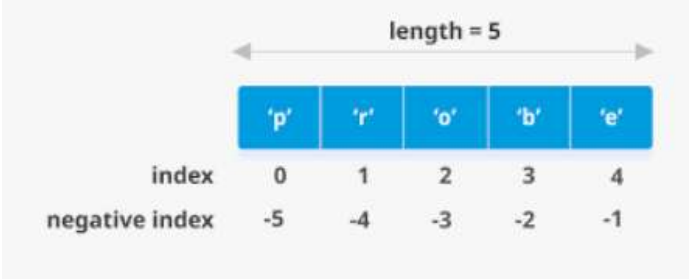
- String is defined as a sequence of characters.
- String can contain alphabets, numbers and special characters.
- String is either enclosed within " "**(double quotes) or ' '(single quotes)**
- Strings are **immutable in** nature-characters of a string cannot be modified.
- Characters of a String can be accessed by their index.

String Creation

```
>>> str1="vijaya kumar"
>>> str1
'vijaya kumar'
>>> str1='recvijay'
>>> str1
'recvijay'
>>> str1="\"Hello Vijay\""
>>> str1
'"Hello Vijay"'
>>> print(str1)
"Hello Vijay"
>>>
```

```
>>> str2=str("Hello")
>>> str2
'Hello'
>>>
```

String Indexing




```
>>> str1="\"Hello Vijay\""
>>> str1[0]
'"'
>>> str1[1]
'H'
>>> str1[-1]
'"'
>>> str1[-2]
'y'
```

Strings are Immutable

- Since Strings are immutable , characters cannot be modified

Example | OUTPUT

```
>>> str1
'"Hello Vijay"'
>>> str1[0]='R'
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    str1[0]='R'
TypeError: 'str' object does not support item assignment
```

- Slicing [::] (i.e) list[start:stop:step]
- Concatenation = +
- Repetition= *
- Membership = in
- Identity = is

String Operators

Slicing

```
>>> str1="Vijay"
>>> str1
'Vijay'
>>> str1[::-1]
'yajiV'
>>> str1[::]
'Vijay'
>>> str1[1:3]
'ij'
```

Concatenation

```
>>> str1='Vijay'
>>> str2='Kumar'
>>> str1=str1+str2
>>> str1
'VijayKumar'
```

Membership and identity

```
>>> str1
'recvijayrecvijay'
>>> 'r' in str1
True
>>> 'z' in str1
False
>>> str1 is str2
False
>>> str1 is not str2
True
```

Repetition

```
>>> str1=str1*2
>>> str1
'recvijayrecvijay'
```

‘s’ is different from ‘S’????

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z

Decimal	Character
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z

ASCII VALUE

Each and every character in a string is associated with a decimal value(alphabets, integer and special characters)

String :Built-in Methods

Case conversion methods	Find and Replace Methods
Character classification methods	String Formatting Methods

Case conversion methods

Methods in this group perform case conversion on the target string.

s.capitalize() returns a copy of s with the first character converted to uppercase and all other characters converted to lowercase

s.lower() returns a copy of s with all alphabetic characters converted to lowercase

s.swapcase() returns a copy of s with uppercase alphabetic characters converted to lowercase and vice versa

s.upper() returns a copy of s with all alphabetic characters converted to uppercase

Case conversion methods

Methods in this group perform case conversion on the target string.

```
>>> s="hi How are u"
>>> s.capitalize()
'Hi how are u'
>>> s.lower()
'hi how are u'
>>> s.upper()
'HI HOW ARE U'
>>> s.swapcase()
'HI hOW ARE U'
>>> s.title()
'Hi How Are U'
>>> s
'hi How are u'
```

Original string is not modified new strings are created

Find methods

These methods provide various means of searching the target string for a specified substring.

count(substring ,start , end):Counts occurrences of a substring in the target string

endswith (substring ,start , end) :Determines whether the target string ends with a given substring.

find (substring ,start , end): Searches the target string for a given substring.

rfind (substring ,start , end) :Searches the target string for a given substring starting at the end.

startswith(substring ,start , end): Determines whether the target string starts with a given substring.

Find methods

These methods provide various means of searching the target string for a specified substring.

```
>>> s
'gee hee lee bee'
>>> s.count('ee')
4
>>> s.startswith('gee')
True
>>> s.endswith('bee')
True
>>> s.find('ee')
1
>>> s.rfind('ee')
13
```

Replace method

s.replace(old , new , count): Replaces occurrences of a substring within a string and returns a new string.

```
>>> s
'gee hee lee bee'
>>> s1=s.replace('ee', 'hh', 2)
>>> s1
'ghh hhh lee bee'
>>> s2=s.replace('ee', 'hh')
>>> s2
'ghh hhh lhh bhh'
```

Character classification methods

Methods in this group classify a string based on the characters it contains.

isalnum(): Determines whether the target string consists of alphanumeric characters.

isalpha():Determines whether the target string consists of alphabetic characters.

isdigit():Determines whether the target string consists of digit characters.

islower():Determines whether the target string's alphabetic characters are lowercase.

isspace():Determines whether the target string consists of whitespace characters.

isupper():Determines whether the target string's alphabetic characters are uppercase.

istitle():Determines whether the target string is title cased.

center():Method will center align the string, using a specified character (space is default) as the fill character.

Character classification methods

Methods in this group classify a string based on the characters it contains.

```
>>> s="Vijay 143"
>>> s.isalnum()
False
>>> s="Vijay143"
>>> s.isalnum()
True
>>> s.isalpha()
False
>>> s="143"
>>> s.isdigit()
True
>>> s="1 4 3"
>>> s.isspace()
False
>>> s=" "
>>> s.isspace()
True
```

```
>>> s="dfg"
>>> s.islower()
True
>>> s="QWERTY"
>>> s.isupper()
True
```

```
print("*", "abcde".center(7), '*', sep='')
* abcde *
```

String formatting methods

Methods in this group modify or enhance the format of a string.

Refer the below url,

https://realpython.com/python-strings/#specifying-a-stride-in-a-string-slice

Conversion between string and list

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.

join(<iterable>)
Concatenates strings from an iterable.

```
>>> l1=["Vijay", "Kumar"]
>>> s="".join(l1)
>>> s
'VijayKumar'
```

split(delimeter)
Splits a string into a list of substrings.

```
>>> l1=["Vijay", "Kumar"]
>>> s="".join(l1)
>>> s
'VijayKumar'
>>> s="CSE IT ECE EEE MECH"
>>> s.split()
['CSE', 'IT', 'ECE', 'EEE', 'MECH']
>>> s="CSE>IT>ECE>EEE"
>>> s.split(">")
['CSE', 'IT', 'ECE', 'EEE']
```

Build in functions

Function	Description
`chr()`	Converts an integer to a character
`ord()`	Converts a character to an integer
`len()`	Returns the length of a string
`str()`	Returns a string representation of an object

```
>>> chr(65)
'A'
>>> ord('A')
65
```

Example 1

Write a Python program that takes a complete sentence as an input and find whether each word is a palindrome or not.

Input:
"My mom and dad speak malayalam"
Output:
My: not palindrome
Mom: palindrome
and: not palindrome
speak: not palindrome
malayalam : palindrome

```
sent=input("Enter the sentence:")
l1=sent.split()
print(l1)
for s in l1:
    temp=s[::-1]
    if(s==temp):
        print( s,":is a palindrome")
    else:
        print(s, ":is not a palindrome")
```

Example 2

Write a Python program that takes a String as an input and find whether the word is a palindrome or not after removing maximum of one character.

Input:
"abca"
Output:
Palindrome

Input:
"abc"
Output:
Not Palindrome

```
s = input()
left, right = 0, len(s) - 1
f=True
while left < right:
    if s[left] != s[right]:
        one, two = s[left:right], s[left + 1:right + 1]
        f = one == one[::-1] or two == two[::-1]
        break
    left, right = left + 1, right - 1
if(f):
    print("Palindrome")
else:
    print("Not Palindrome")
```

Example 3

Robot Return to Origin

There is a robot starting at the position (0, 0), the origin, on a 2D plane. Given a sequence of its moves, judge if this robot ends up at (0, 0) after it completes its moves.

You are given a string moves that represents the move sequence of the robot where moves[i] represents its ith move. Valid moves are 'R' (right), 'L' (left), 'U' (up), and 'D' (down).

Return true if the robot returns to the origin after it finishes all of its moves, or false otherwise.

Note: The way that the robot is "facing" is irrelevant. 'R' will always make the robot move to the right once, 'L' will always make it move left, etc. Also, assume that the magnitude of the robot's movement is the same for each move.

Example 3(COND)

Robot Return to Origin

Example-1:
Input: moves = "UD"
Output: true
Explanation: The robot moves up once, and then down once. All moves have the same magnitude, so it ended up at the origin where it started. Therefore, we return true.

Example-2:
Input: moves = “LL"
Output: false
Explanation: The robot moves left twice. It ends up two "moves" to the left of the origin. We return false because it is not at the origin at the end of its moves.

```
moves = input()
print(moves.count('L') == moves.count('R') and moves.count('U') == moves.count('D'))
```

Example 4

Rotate String

- Given two strings s and goal, PRINT true if and only if s can become goal after some number of shifts on s.
- A shift on s consists of moving the leftmost character of s to the rightmost position.
- For example, if s = "abcde", then it will be "bcdea" after one shift.
- INPUT: s = "abcde", goal = "cdeab“
- OUTPUT: true
- INPUT: s = "abcde", goal = "abced"
- OUTPUT: false

```
A = input()
B = input()
print(len(A) == len(B) and B in A + A)
```

Example 5

Valid Anagram

- Given two strings s and t, Print true if t is an anagram of s, and false otherwise.
- An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly

- INPUT: s = "anagram", t = "nagaram“
- OUTPUT: true
- INPUT: s = "rat", t = "car“
- OUTPUT: false

```
s = input()
t = input()
count=0
for i in 'abcdefghijklmnopqrstuvwxzy':
    if s.count(i)!=t.count(i):
        count=1
print(count==0)
```


THANK
YOU