Strings

January 25, 2023

1 Python - Strings

- String are collection of characters
- A python string is any text that is enclosed in either single quotes or double quotes or triple quote (for multilined string)
- Strings in python are indexed, which means they have an index value for each character starting from 0
- Characters in the string can be accessed using these indexes
- Unlike lists, Python strings are immutable, which means we cannot modify the content of the string, once it is created.
- strings can be accepted as inputs from user using str(input()) or input() functions

[5]:
```
string = "hello world" # length
print(len(string))
print(max(string)) # UNICODE code point values
print(min(string))
```

```
11
w
```

[ ]:
```
# a - z --> 97 - 122
# A - Z --> 65 - 90
```

[6]:
```
print(ord('a'))
```

```
97
```

[10]:
```
if 'a' > 'A': # 97 > 10
    print('Yes')
else:
    print('No')
```

```
Yes
```

[11]:
```
print(chr(97))
```

```
a
```

[12]:
```
print(chr(65)) # code point value and want to find character use chr
```

```
A
```

[13]:
```
print(ord('A')) # character and want to find out number use ord
```

```
65
```

[17]:
```
# strings are immutable
s = 'python'
# print(s[1])
s[i] = 'y'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [17], in <cell line: 4>()
      2 s = 'python'
      3 # print(s[1])
----> 4 s[i] = 'y'

NameError: name 'i' is not defined
```

[15]:
```
lst = [10, 20, 30]
lst[1] = 50
print(lst)
```

```
[10, 50, 30]
```

[20]:
```
lst = [10, 20, 30, 40, 50, 60]
string = 'python'
print(len(lst), len(string))
print(lst[2], string[2])
print(lst[-1], string[-1])
```

```
6 6
30 t
60 n
```

[22]:
```
lst = [10, 20, 30, 40, 50, 60]
# element based access
for i in lst:
    print(i, end = ' ')
# index based access
for i in range(len(lst)):
    print(lst[i], end = ' ')
```

```
10 20 30 40 50 60 10 20 30 40 50 60
```

[23]:
```
string = 'python'
# ind     012345
# character based access
```

```
for i in string:
    print(i)
```

```
p
y
t
h
o
n
```

[24]:
```
string = 'python'
# ind     012345
# index based access
for i in range(len(string)):
    print(string[i])
```

```
p
y
t
h
o
n
```

[28]:
```
lst = [10, 20, 30, 40, 50, 60] # 10 30 50
string = 'python'
print(lst[1:4], string[1:4])
print(lst[::2], string[::2])
```

```
[20, 30, 40] yth
[10, 30, 50] pto
```

[70]:
```
# Count how many uppercase, lowercase letters and digits and
# special characters are there in the given string
s = "This is Python 3.10.4 Wishing you GOOD Day"
up = lw = di = sp = 0
for i in s:
    if i.isupper():
        up += 1
    elif i.islower():
        lw += 1
    elif i.isdigit():
        di += 1
    else:
        sp += 1
print(up, lw, di, sp)
print(len(s), up+lw+di+sp)
```

```
8 21 4 9
42 42
```

[39]:
```
x = 'x'
print(chr(ord(x) - 32))
```

```
X
```

[42]:
```
s = 'abcdef'
new_s = ''
for i in s:
    new_s += chr(ord(i) - 32) # ABCDEF
print(new_s)
```

```
ABCDEF
```

[ ]:
```

```

1.1 string methods

1.1.1 str.upper()

- To convert a string to uppercase

[45]:
```
s = 'python'
s1 = s.upper() # returns a new string with all alphabets changed
# to uppercase
print(s1)
# print(s)
```

```
PYTHON
python
```

[49]:
```
s = 'python' # strings are immutable
s.upper()
print(s)
```

```
python
```

1.1.2 str.lower()

- To convert a string into lowercase

[50]:
```
s = 'PYTHON'
s1 = s.lower()
print(s1)
```

```
python
```

1.1.3 str.capitalize()

- To convert a string's first character into uppercase

[52]:
```
s = 'python is easy'
s1 = s.capitalize()
print(s1)
```

```
Python is easy
```

1.1.4 str.title()

- To convert everywords first character into uppercase

[53]:
```
s = 'python is simple'
s1 = s.title()
print(s1)
```

```
Python Is Simple
```

1.1.5 str.swapcase()

- To covert lowercase alphabets into upper and vice versa

[54]:
```
s = 'pYtHon Is SiMPlE'
s1 = s.swapcase()
print(s1)
```

```
PyThON iS sImpLe
```

1.1.6 str.isupper()

- To check if the string is made of uppercase alphabets only (if it contais alphabets at all)
- Returns a True, if every alphabet in the string is a uppercase alphabet, else False

[55]:
```
s = 'HELLO ALL'
print(s.isupper())
```

```
True
```

[56]:
```
s = 'HELLO AlL'
print(s.isupper())
```

```
False
```

[57]:
```
s = 'helloworld'
print(s.isupper())
```

```
False
```

[58]:
```
s = 'HELLO THIS IS 123 $%^^'
print(s.isupper())
```

```
True
```

[59]:
```
s = 'HELLO THIS Is 123 $%^^'
print(s.isupper())
```

```
False
```

[60]:
```
s = '&^*&^*'
print(s.isupper())
```

```
False
```

1.1.7 str.islower()

- To check if the string is made of lower alphabets only (if it contais alphabets at all)
- Returns a True, if every alphabets in the string is an lowercase alphabet, else False

[ ]:
```

```

1.1.8 str.isdigit()

- Returns true if the string contains only digits, else false

[61]:
```
s = '1234'
print(s.isdigit())
```

```
True
```

[62]:
```
s = '1234s'
print(s.isdigit())
```

```
False
```

[63]:
```
s = '1234#'
print(s.isdigit())
```

```
False
```

[69]:
```
'7'.isdigit()
```

[69]:
```
True
```

1.1.9 str.isalpha()

- Returns True, if the string is containing alphabets only, else false

[71]:
```
s = 'hello WORLD'
print(s.isalpha())
```

```
False
```

[72]:
```
s = 'helloWORLD'
print(s.isalpha())
```

```
True
```

[73]:
```
s = 'helloWORL$D'
print(s.isalpha())
```

```
False
```

1.1.10 str.isalnum()

- To check if the string is containing alphanumeric values
- Return True, if either of the following is True
  - String contains alphabets and digits only
  - String contains only alphabets
  - String contains only digits

[74]:
```
s = '20A91a0347'
print(s.isalnum())
```

```
True
```

[75]:
```
s = '20A91a0347 '
print(s.isalnum())
```

```
False
```

[76]:
```
s = '1234'
print(s.isalnum())
```

```
True
```

[77]:
```
s = 'asdfkljsdSDFJ'
print(s.isalnum())
```

```
True
```

1.1.11 str.istitle()

- To check if a string is in title case

[79]:
```
s = 'this is python'
print(s.istitle())
```

```
False
```

[80]:
```
s = 'This Is Python'
print(s.istitle())
```

```
True
```

1.1.12 str.ljust()

- To left align the string using a padding character

- Will put the string to left side and pads the remaining spaces with specified character, if no character specified pads using space

[82]:
```
s = 'python'
s1 = s.ljust(20) # left justification
s1
```

[82]: 'python '

[83]:
```
s = 'python'
s1 = s.ljust(20, '#')
s1
```

[83]: 'python##############'

1.2 ### str.rjust()

1.2.1 str.center()

- To align a string in the center of given no of spaces

[87]:
```
s = 'python'
s1 = s.center(10)
s1
```

[87]: ' python '

[88]:
```
s = 'python'
s1 = s.center(10, '$')
s1
```

[88]: '$$python$$'

1.2.2 str.count()

- Returns the number of times a given substring present in the given string

[1]:
```
s = 'this is python'
print(s.count('i'))
```

2

[2]:
```
s = 'this is python'
print(s.count('is'))
```

2

[3]:
```
s = 'this is python'
print(s.count('tho'))
```

```
1
```

[4]:
```
s = 'this is python'
print(s.count('python'))
```

```
1
```

[5]:
```
s = 'this is python'
print(s.count('c'))
```

```
0
```

1.2.3 str.startswith()

- Checks if the original string starts with a substring given

[6]:
```
s = 'this is python'
print(s.startswith('t'))
```

```
True
```

[7]:
```
s = 'this is python'
print(s.startswith('h'))
```

```
False
```

[8]:
```
s = 'this is python'
print(s.startswith('this'))
```

```
True
```

[9]:
```
s = 'this is python'
print(s.startswith('thisis'))
```

```
False
```

1.3 ### str.endswith()

[ ]:
```

```

1.3.1 str.rstrip()

- To strip extra characters on the right side of the string

[11]:
```
s = 'this is python              ' # trailing spaces
s1 = s.rstrip()
s1
```

[11]: 'this is python'

[14]:
```
s = 'this is pythonnnnnnnnnnnnnnnn'
s1 = s.rstrip('n')
print(s1)
```

```
this is pytho
```

1.3.2 str.lstrip()

- To strip extra characters on the right side of the string

[16]:
```
x = '               this is python' # leading spaces
x1 = x.lstrip()
x1
```

[16]: 'this is python'

[17]:
```
x = 'zzzzzzzzzzzzzzzzzzthis is python'
x1 = x.lstrip('z')
x1
```

[17]: 'this is python'

1.3.3 str.strip()

- To truncate both leading and trailing characters from the given string

[19]:
```
s = '                    this is python                  '
s1 = s.strip()
s1
```

[19]: 'this is python'

1.3.4 str.find()

- To find a substring in the given string
- If substring is found, returns the index of the first character of substring, else returns -1

[20]:
```
s = 'this is python'
print(s.find('t'))
```

```
0
```

[21]:
```
s = 'this is python'
print(s.find('is'))
```

```
2
```

[22]:
```
s = 'this is python'
print(s.find('hello'))
```

```
-1
```

[23]:
```
s = 'this is python'
print(s.find('z'))
```

```
-1
```

[25]:
```
s = 'this is python'
print(s.find('t', 1))
```

```
10
```

[27]:
```
s = 'this is python'
print(s.find('p', 1, 7))
```

```
-1
```

[28]:
```
s = 'this is python'
print(s.find('p', 1, 8)) # end bound excluded
```

```
-1
```

1.3.5 str.split()

- Splits the given string with the given delimeter and returns a list

[29]:
```
s = 'this is python' # 3
print(s.split()) # always a list
```

```
['this', 'is', 'python']
```

[30]:
```
s = 'this is python' # 3
print(s.split('i')) # always a list
```

```
['th', 's ', 's python']
```

[31]:
```
s = 'this is python' # 3
print(s.split('t')) # always a list
```

```
['', 'his is py', 'hon']
```

1.3.6 str.join(iterable)

- joins the elements of iterable using given string
- iterable should contains string values in order for the join to work

[32]:
```
x = ['a', 'b', 'c', 'd']
print(''.join(x)) # 'abcd'
```

```
abcd
```

[33]:
```
x = ['a', 'b', 'c', 'd']
print(' '.join(x))
```

```
a b c d
```

[34]:
```
x = ['a', 'b', 'c', 'd']
print('python'.join(x))
```

```
apythonbpythoncpythond
```

[35]:
```
x = [10, 20, 30, 40]
z = 'p'.join(x)
print(z)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [35], in <cell line: 2>()
      1 x = [10, 20, 30, 40]
----> 2 z = 'p'.join(x)
      3 print(z)

TypeError: sequence item 0: expected str instance, int found
```

[36]:
```
print('abcd'.join('python'))
```

```
pabcdyabcdtabcdhabcdoabcdn
```