Python Programming

by Narendra Allam

Chapter 2

Strings

Topics Covering

- Strings
    - Define a string - Multiple quotes and Multiple lines
    - String functions
    - String slicing - start, end & step
    - Negative indexing
    - Scalar multiplication
    - Commenting in python
- Interview Questions
- Exercise Programs
- Notes

Strings

- String is a Collection of characters.
- Any pair of quotes can be used to represent a string.
- Strings are immutable, we cannot add, delete, modify individual characters in a string.

In[]
```
s = 'Apple'
s = "Apple"

s = '''Apple is sweet,
Orange is sour'''

s = """Apple is sweet,
Orange is sour"""


s =  "John's Byke"
```

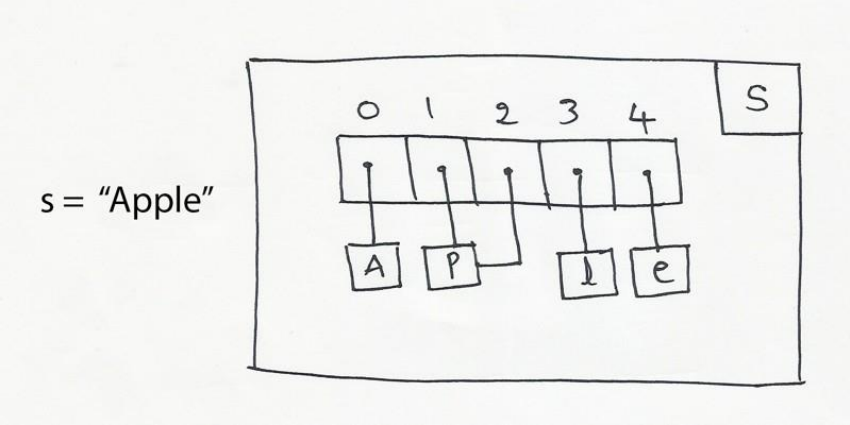
In[]
```
s = 'Apple'
```

Individual characters in a string can be accessed using square brackets and indexing. Indexing starts from zero.
s[0] is 'A'
s[1] is 'p'
and so on.

```
In[] s = 'Apple'
     print s[0], s[1], s[2]
```

```
A p p
```

internal represenation of a string




```
In[] print id(s[0]), id(s[1]), id(s[2])
```

```
4482563288 4481946088 4481946088
```

In the above example 'p' is stored only once and its reference(address) is placed two times, at index 1 and 2, in the list of characters.

Finding length of the string - number of character in a string
len() function:

```
In[] s = "Hello World!"
     print len(s)
```

```
12
```

Strings are immutable

- we cannot change individual characters
- We cannot add or delete characters

```
In[]
s[4] = 'X'
```

```
---------------------------------------------------------------------
---------
TypeError                                 Traceback (most recent c
all last)
<ipython-input-6-a389171c1303> in <module>()
      1 # **** Strings are immutable, we cannot change the  charact
ers
----> 2 s[4] = 'X'

TypeError: 'str' object does not support item assignment
```

```
In[] print s[100]
```

```
---------------------------------------------------------------------
---------
IndexError                                Traceback (most recent c
all last)
<ipython-input-7-8a5646048cdd> in <module>()
----> 1 print s[100]

IndexError: string index out of range
```

String slicing

Slicing the technique of extracting sub string or a set of characters form a string.

syntax :

```
string[start:end:step]
```

- start - index start at which slicing is started
- end - index at which slicing is ended, end index is exclusive
- step - step value is with which start value gets incremented/decremented.

Note: Default step value is 1.

Lets see some examples,

```
In[] s = "Hello World!"

     s[6:11]
```

```
Output: 'World'
```

```
In[] s[1:5]
```

```
Output: 'ello'
```

```
In[] s[:4]
```

```
Output: 'Hell'
```

```
In[] s[4:]
```

```
Output: 'o World!'
```

```
In[] s[1:9]
```

```
Output: 'ello Wor'
```

```
In[] s[1:9:1]
```

```
Output: 'ello Wor'
```

```
In[] s[1:9:2]
```

```
Output: 'el o'
```

In the above example,
start is 1,
end is 9 and
step is 2.

first it prints s[1],
then s[1 + step] => s[1 + 2] => s[3]
prints s[3]
thne s[3 + step] which is s[5] and so on,
until it crosses 8.

```
In[] s[:10:3]
```

```
Output: 'HlWl'
```

```
In[] s[:]
```

```
Output: 'Hello World!'
```

```
In[] s[::]
Output: 'Hello World!'

In[] s[::2]
Output: 'HloWrd'
```

In the above example, it takes entire string, but step is 2, default start value is 0. so indices produced are, 0, 2, 4, 6, 8, and 10.

```
In[] s[9:2]
Output: ''

In[] s[9:2:-1]
Output: 'lroW ol'
```

-ve indexing [fig]

Python supports -ve indexing. Index of last character is -1, last but one is -2 and so on.

```
In[] s = "Hello World!"
     s[-1]
Output: '!'

In[] s[-2]
Output: 'd'
```

Slicing using -ve indexing:

```
In[] s[-9:-3]
Output: 'lo Wor'
```

default step value is 1,
-9 + 1 ==> -8
-8 + 1 ==> -7
start value -9 is goin towards -3,
-9 ==> -3, so s[-9:-3] is a valid slice.

```
In[] s[-3: -10]
Output: ''
```

Above is not a valid slice, because

step is 1, default.
-3 + 1 ==> -2
-2 + 1 ==> -1
so on
-3 <== -10
-3 is not going towards -10, it never reaches -10, so invalid slice.
It returns ''(null string)

Some more examples,

```
In[] s[-3: -10:-1]
Output: 'lroWol'

In[] s[-4:-1:1]
Output: 'rld'

In[] s[-2:-10:-1]
Output: 'dlroWol'
```

Reversing a string

```
In[] s[::-1]
Output: '!dlroWolleH'
```

Unfortuantely this is the only standard way we can reverse a string in python. There are other complicated ways but not used in production.

```
In[] s[3::-1]
Output: 'lleH'

In[] s[:3:-1]
Output: '!dlroW'
```

String functions

There are some usuful functions on strings, below is the listing.

```
In[] s = "hello World! 123$"
```

capitalize(): Captilize the first character and make remaining characters smalle

```
In[] s.capitalize()
Output: 'Hello world! 123$'
```

Note: String functions do not effect original string, instead they take a copy of original string, process it and returns.

```
In[] s
Output: 'hello World! 123$'
```

count(): Counts number of chars/substrings it has

```
In[] s.count('l')
Output: 3

In[] s.count('hell')
Output: 1
```

upper() and lower(): changing case to upper and lower, no effect on numbers and other characters.

```
In[] s.upper()
Output: 'HELLO WORLD! 123$'

In[] s.lower()
Output: 'hello world! 123$'
```

Validation functions

```
In[] s.endswith("3$")
```

```
Output: True
```

```
In[] s.endswith("5$")
```

```
Output: False
```

```
In[] s.startswith("Apple")
```

```
Output: False
```

```
In[] s.startswith("hello")
```

```
Output: True
```

```
In[] s = 'Apple123'
     s.isalpha()
```

```
Output: False
```

```
In[] s = 'Apple'
     s.isalpha()
```

```
Output: True
```

```
In[] s = "2314"
     s.isdigit()
```

```
Output: True
```

replace(): replaces all the occurances of substring in target string

```
In[] s = 'Apple'
     s.replace('p', '$')
     print s
```

```
     Apple
```

As we discussed, original string doesn't get changed, we just have to capture the modified string if we want to, as below

```
In[] s = 'Apple'
     s = s.replace('p', '$')
     print s
```

```
     A$$le
```

```
In[] s = 'Apple'
     s1 = s.replace('App', 'Tupp')
     print s1, s
```

```
Tupple Apple
```

strip(): Strips spaces on both the sides of the string. We can pass any custome chars/substrings if we want to strip. Below are the examples.

```
In[] s = ' Apple '
     print len(s)
     s = s.strip()
     print len(s)
```

```
7
5
```

```
In[] s = ' Apple'
     print len(s)
     s = s.lstrip()
     print len(s)
```

```
6
5
```

```
In[] s = 'Apple '
     print len(s)
     s = s.rstrip()
     print len(s)
```

```
6
5
```

stripping custom chars/substrings

```
In[] s = 'ApApTuple'
     s.strip('Ap')
```

```
Output: 'Tuple'
```

split(): Splits entire string into multiple words seperated by spaces. We can pass custom sperators if want to.

In[]
```
s = "Apple is a fruit"
l = s.split()
print l, type(l)
```

```
['Apple', 'is', 'a', 'fruit'] <type 'list'>
```

In[]
```
date = '12/02/1984'
l = date.split('/')
print l
```

```
['12', '02', '1984']
```

In[]
```
l[0]
```

Output: '12'

In[]
```
date = '12/02/1984'
l = date.split('/', 1)
print l
```

```
['12', '02/1984']
```

In[]
```
date = '12/02/1984'
l = date.rsplit('/', 1)
print l
print l[-1]
```

```
['12/02', '1984']
1984
```

In[]
```
s = '''Once upon a time in India, there was a king called Tippu.
India was a great country.'''

print s.find('India')
print s.find('Pakisthan')
```

```
20
-1
```

rfind(): searching from the end

In[]
```
s.rfind('India')
```

Output: 58

Index:

In[]
```
s.index('India')
```

Output: 20

In[]
```
s.index('Pakisthan')
```

```
---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
<ipython-input-60-a9f8becf29f2> in <module>()
----> 1 s.index('Pakisthan')

ValueError: substring not  found
```

Printing string from the word 'king':

In[]
```
s = '''Once upon a time in India, there was a king called Tippu.
India was a great country.'''

print s[s.find('great'):]
```

```
great country.
```

List of chars to string:

In[]
```
l = ['A', 'p', 'p', 'l', 'e']
print ''.join(l)
```

```
Apple
```

In[]
```
l = ['A', 'p', 'p', 'l', 'e']
print '|'.join(l)
```

```
A|p|p|l|e
```

In[]
```
emp_data = ['1234', 'John', '23400.0', 'Chicago']

print ','.join(emp_data)
```

```
1234,John,23400.0,Chicago
```

String to list of characters:

In[]
```
s = 'Apple'
print list(s)
```

```
['A', 'p', 'p', 'l', 'e']
```

Program: Reverse the word 'India' in-place in the below string.

In[]
```
s = '''Once upon a time in India, there was a king called Tippu. In
dia was a great country.'''
word = 'India'

s.replace(word, word[::-1])
```

Output:
```
'Once upon a time in aidnI, there was a king called Tippu. aidnI w
as a great country.'
```

Program: Count all the vowels in the given string.

In[]
```
s = '''once upon a time in india, there was a king called tippu. in
dia was a great country.'''

s.count('a')+ s.count('e') + s.count('i') + s.count('o') + s.count(
'u')
```

Output:
```
29
```

Scalar multiplication

In[]
```
'Apple' * 3
```

Output:
```
'AppleAppleApple'
```

Concatenating Strings

In[]
```
'Apple' + 'Orange'
```

Output:
```
'AppleOrange'
```

Commenting in python

Comments are used in the code for descibing the logic. This helps the new devlopers, understanding code better.
In python,

- Hash (#) is uded for single line comments
- Tripple single quotes (''' ''') are used for multiline comments
- Tripple double quotes (""" """) are used for doc strings (describing function parameters or class properties etc.,)

Check all the three types of comments in the below code snippet.

In[]
```
def area(a, b, c):









    s = (a + b+ c)/2.0
    res =  s*(s-a)*(s-b)*(s-c)
    return res ** 0.5
```

Note: You don't need to understand eveything written above. Dont worry! above example is just to give you a glance on commenting.

Interview Questions

1) Output?

In[]
```
s = "Hello World!"
print s[1:9:2]
```

```
el o
```

In[]
```
s = 'Hello World!'
print s[3::-1]
```

```
lleH
```

In[]
```
i = int('234.5')
```

```
---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
<ipython-input-73-bee87622bbdb> in <module>()
----> 1 i = int('234.5')

ValueError: invalid literal for int() with base 10: '234.5'
```

In[]
```
print 'Apple123'.upper()
```

```
APPLE123
```

2) How do you reverse a string?

Exercise Programs

1. Add a comma between the characters. If the given woord is 'Apple', it should become 'A,p,p,l,e'
2. Remove the given word in all the places in a string?

Notes:

1. default character encoding in python 2 is ASCII, where as in python 3 it is Unicode
2. lower() and upper() functions do not have any effect on non alphabet characters