4

Strings

"Puppeting on strings..."

Contents

KanNotes

What are Strings?

- Python string is a collection of Unicode characters.
- Python strings can be enclosed in single, double or triple quotes.

  'BlindSpot'
  "BlindSpot"
  ' ' 'BlindSpot' ' '
  """Blindspot"""

- If there are characters like ' " or \ within a string, they can be retained in two ways:

  (a) Escape them by preceding them with a \
  (b) Prepend the string with a 'r' indicating that it is a raw string

```
msg = 'He said, \'Let Us Python.\''
msg = r'He said, 'Let Us Python.''
```

- Multiline strings can be created in 3 ways:

  - All but the last line ends with \
  - Enclosed within """some msg """ or ' ' 'some msg' ' '
  - ('one msg'
     'another msg')

Accessing String Elements

- String elements can be accessed using an index value, starting with 0. Negative index value is allowed. The last character is considered to be at index -1. Positive and negative indices are show in Figure 4.1.

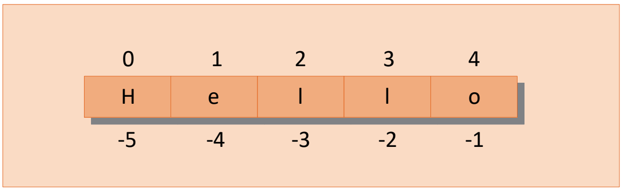

Figure 4.1

- Examples of positive and negative indexing:

```
msg = 'Hello'
a = msg[0]          # yields H
b = msg[4]          # yields o
c = msg[-0]         # yields H, -0 is same as 0
d = msg[-1]         # yields o
e = msg[-2]         # yields l
f = msg[-5]         # yields H
```

- A sub-string can be sliced out of a string.

  s[start : end] - extract from start to end - 1.
  s[start :] - extract from start to end.
  s[: end] - extract from start to end - 1.
  s[-start :] - extract from -start (included) to end.
  s[: -end] - extract from beginning to -end - 1.

- Using too large an index reports an error, but using too large index while slicing is handled elegantly.

```
msg = 'Rafting'
print(msg[3:100])       # prints elements from 't' up to end of string
print(msg[100])         # error since 100 th element doesn't exist
```

String Properties

- Python strings are immutable—they cannot be changed.

```
s = 'Hello'
s[0] = 'M'        # rejected, attempt to mutate string
s = 'Bye'         # s is a variable, it can change
```

- Strings can be concatenated using +.

```
msg3 = ms1 + msg2
```

- Strings can be replicated during printing.

```
print('-', 50)    # prints 50 dashes
```

- Whether one string is part of another can be found out using **in**.

```
print('e' in 'Hello')       # prints True
print('z' in 'Hello')       # print False
print('lo' in 'Hello')       # prints True
```

Built-in Functions

- Some built-in functions can be used with a string:

```
msg = 'Surreal'
print(len(msg))          # prints 7 - length of string
print(min(msg))          # prints S - character with min value
print(max(msg))          # prints u - character with max value
```

String Methods

- When we create a string a nameless object of type **str** is created.

```
msg = 'Surreal'
print(type(msg))         # prints <class 'str'>
print(id(msg))           # prints 33720000
```

  Address of the nameless **str** object is stored in **msg**. which is returned by the built-in **id( )** function.

- An object of type **str** contains methods using which it can be accessed and modified. These methods can be called using a syntax similar to calling a function in a module as shown below:

```
import random
num = random.randint(1, 25)        # syntax  module.function( )
s =  'Hello'
s.upper( )                         # syntax  string.method( )
```

- Different categories of string methods are given below.

  # content test functions
  isalpha( ) - checks if all characters in string are alphabets.
  isdigit( ) - checks if all characters in string are digits.
  isalnum( ) - checks if all characters in string are alphabets or digits.
  islower( ) - checks if all characters in string are lowercase alphabets.
  isupper( ) - checks if all characters in string are uppercase alphabets.
  startswith( ) - checks if string starts with a value.
  endswith( ) - checks if string ends with a value.

  # search and replace
  find( ) - searches for a value, returns its position.
  replace( ) - replace one value with another.

# trims whitespace
lstrip( ) - removes whitespace from the left of string including \t.
rstrip( ) - removes whitespace from the right of string including \t.
strip( ) - removes whitespace from left and right

# split and partition
split( ) - split the string at a specified separator string.
partition( ) - partitions string into 3 parts at first occurrence of specified string.

# join - different than concatenation. It joins string to each element of string1 except last.
join(string1)

- Following program shows how to use the string methods:

```
msg = 'Hello'
print(msg.replace('l', 'L'))        # replaces l with L in Hello
print("-".join("Hello"))            # prints H-e-l-l-o
```

String Conversions

- Two types of string conversions are required frequently:
  - Converting the case of characters in string
  - Converting numbers to string and vice versa
- Case conversions can be done using **str** methods:

  upper( ) - converts string to uppercase.
  lower( ) - converts string to uppercase.
  capitalize( ) - converts first character of string to uppercase.
  title( ) - converts first character of each word to uppercase.
  swapcase( ) - swap cases in the string.

```
msg = 'Hello'
print(msg.upper( ))          # prints HELLO
print('Hello'.upper( ))      # prints HELLO
```

- Built-in functions are used for string to number conversions and vice versa:

  str( ) - converts an int, float or complex to string
  int( ) - converts a numeric string to int
  float( ) - converts a numeric string to float

complex( ) - converts a numeric string to complex

- The built-in function **chr( )** returns a string representing its Unicode value (known as code point). **ord( )** does the reverse.
- Following program shows how to use the conversion functions:

```
age = 25
print('She is ' + str(age) + ' years old')
i = int("34")
f = float("3.14")
c = complex("3+2j")          # "3 + 2j" would be a malformed string
print(ord('A'))              # prints 65
print(chr(65))               # prints A
```

String Comparison

- Two strings can be compared using operators ==, !=, <, >, <=, >=. This is shown in the following program:

```
s1 = "Bombay"
s2 = "bombay"
s3 = "Nagpur"
s4 = "Bombaywala"
s5 = "Bombay"
print(s1 == s2)              # displays False
print(s1 == s5)              # displays True
print(s1 != s3)              # displays True
print(s1 > s5)               # displays False
print(s1 < s2)               # displays True
print(s1 <= s4)              # displays True
```

- String comparison is done in lexicographical order (alphabetical) character by character. Capitals are considered to be less than their lowercase counterparts. Result of comparison operation is either True or False.
- Note that there is only one **str** object containing "Bombay", so **s1** and **s5** both contain the same address.

P</> Programs

Problem 4.1

Demonstrate how to create simple and multi-line strings and whether a string can change after creation. Also show the usage of built-in functions **len( )**, **min( )** and **max( )** on a string.

Program

```
# simple strings
msg1 = 'Hoopla'
print(msg1)
# strings with special characters
msg2 = 'He said, \'Let Us Python\'.'
file1 = 'C:\\temp\\newfile'
file2 = r'C:\temp\newfile'   # raw string - prepend r
print(msg2)
print(file1)
print(file2)

# multiline strings
# whitespace at beginning of second line becomes part of string
msg3 = 'What is this life if full of care...\
      We have no time to stand and stare'
# enter at the end of first line becomes part of string
msg4 = """What is this life if full of care...
We have no time to stand and stare"""
# strings are concatenated properly.( ) necessary
msg5 = ('What is this life if full of care...'
      'We have no time to stand and stare')
print(msg3)
print(msg4)
print(msg5)

# string replication during printing
msg6 = 'MacLearn!!'
print(msg1 * 3)

# immutability of strings
# Utopia cannot change, msg7 can
```

```
msg7 = 'Utopia'
msg8 = 'Today!!!'
msg7 = msg7 + msg8    # concatenation using +
print(msg7)

# use of built-in functions on strings
print(len('Hoopla'))
print(min('Hoopla'))
print(max('Hoopla'))
```

Output

```
Hoopla
He said, 'Let Us Python'.
C:\temp\newfile
C:\temp\newfile
What is this life if full of care...        We have no time to stand and stare
What is this life if full of care...
We have no time to stand and stare
What is this life if full of care...We have no time to stand and stare
HooplaHooplaHoopla
UtopiaToday!!!
6
H
p
```

Tips

- Special characters can be retained in a string by either escaping them or by marking the string as a raw string.
- Strings cannot change, but the variables that store them can.
- **len( )** returns the number of characters present in string. **min( )** and **max( )** return the character with minimum and maximum Unicode value from the string.

Problem 4.2

For a given string 'Bamboozled', write a program to obtain the following output:

```
B a
e d
e d
mboozled
mboozled
mboozled
Bamboo
Bamboo
Bamboo
Bamboo
delzoobmaB
Bamboozled
Bmoze
Bbzd
Boe
BamboozledHype!
BambooMonger!
```

Use multiple ways to get any of the above outputs.

Program

```
s = 'Bamboozled'
# extract B a
print(s[0], s[1])
print(s[-10], s[-9])
# extract e d
print(s[8], s[9])
print(s[-2], s[-1])

# extract mboozled
print(s[2:10])
print(s[2:])
print(s[-8:])

# extract Bamboo
print(s[0:6])
print(s[:6])
print(s[-10:-4])
print(s[:-4])

# reverse Bamboozled
```

```
print([::-1])

print(s[0:10:1])
print(s[0:10:2])
print(s[0:10:3])
print(s[0:10:4])

s = s + 'Hype!'
print(s)
s = s[:6] + 'Monger' + s[-1]
print(s)
```

Tips

- Special characters can be retained in a string by either escaping them or by marking the string as a raw string.
- s[4:8] is same as s[4:8:1], where 1 is the default.
- s[4:8:2] returns a character, then move forward 2 positions, etc.

Problem 4.3

For the following strings find out which are having only alphabets, which are numeric, which are alphanumeric, which are lowercase, which are uppercase. Also find out whether 'And Quiet Flows The Don' begins with 'And' or ends with 'And' :

'NitiAayog'
'And Quiet Flows The Don'
'1234567890'
'Make $1000 a day'

Program

```
s1 = 'NitiAayog'
s2 = 'And Quiet Flows The Don'
s3 = '1234567890'
s4 = 'Make $1000 a day'
print('s1 = ', s1)
print('s2 = ', s2)
print('s3 = ', s3)
print('s4 = ', s4)
```

```
# Content test functions
print('check isalpha on s1, s2')
print(s1.isalpha( ))
print(s2.isalpha( ))

print('check isdigit on s3, s4')
print(s3.isdigit( ))
print(s4.isdigit( ))

print('check isalnum on s1, s2, s3, s4')
print(s1.isalnum( ))
print(s2.isalnum( ))
print(s3.isalnum( ))
print(s4.isalnum( ))

print('check islower on s1, s2')
print(s1.islower( ))
print(s2.islower( ))

print('check isupper on s1, s2')
print(s1.isupper( ))
print(s2.isupper( ))

print('check startswith and endswith on s2')
print(s2.startswith('And'))
print(s2.endswith('And'))
```

Output

```
s1 =  NitiAayog
s2 =  And Quiet Flows The Don
s3 =  1234567890
s4 =  Make $1000 a day
check isalpha on s1, s2
True
False
check isdigit on s3, s4
True
False
check isalnum on s1, s2, s3, s4
```

```
True
False
True
False
check islower on s1, s2
False
False
check isupper on s1, s2
False
False
check startswith and endswith on s2
True
False
```

Problem 4.4

Given the following string:

```
'Bring It On'
'    Flanked by spaces on either side    '
'C:\\Users\\Kanetkar\\Documents'
```

write a program to produce the following output using appropriate string functions.

```
BRING IT ON
bring it on
Bring it on
Bring It On
bRING iT oN
6
9
Bring Him On
Flanked by spaces on either side
    Flanked by spaces on either side
['C:', 'Users', 'Kanetkar', 'Documents']
('C:', '\\', 'Users\\Kanetkar\\Documents')
```

Program

```
s1 = 'Bring It On'
# Conversions
```

```
print(s1.upper( ))
print(s1.lower( ))
print(s1.capitalize( ))
print(s1.title( ))
print(s1.swapcase( ))

# search and replace
print(s1.find('I'))
print(s1.find('On'))
print(s1.replace('It', 'Him'))

# trimming
s2 = '    Flanked by spaces on either side    '
print(s2.lstrip( ))
print(s2.rstrip( ))

# splitting
s3 = 'C:\\Users\\Kanetkar\\Documents'
print(s3.split('\\'))
print(s3.partition('\\'))
```

Problem 4.5

Find all occurrences of 'T' in the string 'The Terrible Tiger Tore The Towel'. Replace all occurrences of 'T' with 't'.

Program

```
s = 'The Terrible Tiger Tore The Towel'
pos = s.find('T', 0)
print(pos, s[pos])
pos = s.find('T', pos + 1)
print(pos, s[pos])
pos = s.find('T', pos + 1)
print(pos, s[pos])
pos = s.find('T', pos + 1)
print(pos, s[pos])
pos = s.find('T', pos + 1)
print(pos, s[pos])
pos = s.find('T', pos + 1)
print(pos, s[pos])
```

```
pos = s.find('T', pos + 1)
print(pos)
c = s.count('T')
s = s.replace('T', 't', c)
print(s)
```

Output

```
0 T
4 T
13 T
19 T
24 T
28 T
-1
the terrible tiger tore the towel
```

Tips

- First call to **search( )** returns the position where first 'T' is found. To search subsequent 'T' search is started from **pos + 1**.
- When 'T' is not found **search( )** returns -1.
- **count( )** returns the number of occurrences of 'T' in the string.
- Third parameter in the call to **replace( )** indicates number of replacements to be done.

Exercises

[A] Answer the following questions:

(a) Write a program that generates the following output from the string 'Shenanigan'.

```
S h
a n
enanigan
Shenan
Shenan
Shenan
```

```
Shenan
Shenanigan
Seaia
Snin
Saa
ShenaniganType
ShenanWabbite
```

(b) Write a program to convert the following string

'Visit ykanetkar.com for online courses in programming'

into

'Visit Ykanetkar.com For Online Courses In Programming'

(c) Write a program to convert the following string

'Light travels faster than sound. This is why some people appear bright until you hear them speak.'

into

'LIGHT travels faster than SOUND. This is why some people appear bright until you hear them speak.'

(d) What will be the output of the following program?

```
s = 'HumptyDumpty'
print('s = ', s)
print(s.isalpha( ))
print(s.isdigit( ))
print(s.isalnum( ))
print(s.islower( ))
print(s.isupper( ))
print(s.startswith('Hump'))
print(s.endswith('Dump'))
```

(e) What is the purpose of a raw string?

(f) If we wish to work with an individual word in the following string, how will you separate them out:

'The difference between stupidity and genius is that genius has its limits'

(g) Mention two ways to store a string: He said, "Let Us Python".

(h) What will be the output of following code snippet?

```
print(id('Imaginary'))
print(type('Imaginary'))
```

(i) What will be the output of the following code snippet?

```
s3 = 'C:\\Users\\Kanetkar\\Documents'
print(s3.split('\\'))
print(s3.partition('\\'))
```

(j) Strings in Python are iterable, sliceable and immutable. (True/False)

(k) How will you extract ' TraPoete' from the string 'ThreadProperties'?

(l) How will you eliminate spaces on either side of the string '   Flanked by spaces on either side   '?

(m) What will be the output of the following code snippet?

```
s1 = s2 = s3 = "Hello"
print(id(s1), id(s2), id(s3))
```

(n) What will get stored in **ch in** the following code snippet:

```
msg = 'Aeroplane'
ch = msg[-0]
```

[B] Match the following pairs assuming msg = 'Keep yourself warm'

a. msg.partition(' ')	1. 18
b. msg.split(' ')	2. kEEP YOURSELF WARM
c. msg.startswith('Keep')	3. Keep yourself warm
d. msg.endswith('Keep')	4. 3
e. msg.swapcase( )	5. True
f. msg.capitalize( )	6. False
g. msg.count('e')	7. ['Keep', 'yourself', 'warm']
h. len(msg)	8. ('Keep', ' ', 'yourself warm')
i. msg[0]	9. Keep yourself w
j. msg[-1]	10. keep yourself wa
k. msg[1:1:1]	11. K
l. msg[-1:3]	12. empty string
m. msg[:-3]	13. m
n. msg[-3:]	14. arm
o. msg[0:-2]	15. empty string