Strings

CONTENTS

- Indexing and slicing operations,
- String methods
  - index
  - count
  - strip
  - split
  - other text processing methods

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, string is a sequence of Unicode character.

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 and docstrings.

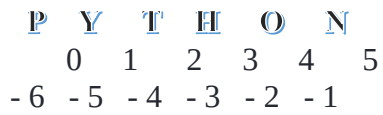
Example:

```
all of the following are equivalent
my_string = 'Hello' print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
           the world of Python"""
print(my_string)
```

```
Hello
Hello
Hello
Hello, welcome to
           the world of Python
```

Indexing and slicing

We can access individual characters using indexing and a range of characters using slicing. Index starts from 0. Trying to access a character out of index range will raise an IndexError. The index must be an integer. We can't use float or other types, this will result into TypeError. Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on. We can access a range of items in a string by using the slicing operator (colon).



Example:

```
var = "Python"
print(var)       # Python
print(var[2])    # t
print(var[-1])   # n
print(var[0:3])  # Pyt
print(var[-3:])  # hon
print(var[:4])   # Pyth
print(var[4:])   # on
print(var[:-1])  # Pytho
print(var[1:-1]) # ytho
```

```
Python
t
n
Pyt
hon
Pyth
on
Pytho
ytho
```

Ex. WAP to swap first and last character of a string entered by user

```
mystr = input("Enter a string : ")
v1 = mystr[1:-1]
v2 = mystr[0]
v3 = mystr[-1]
v4 = v3 + v1 + v2
print(v4)

print(mystr[-1]+mystr[1:-1]+mystr[0])
```

```
Enter a string : mumbai
iumbam
iumbam
```

Strings are Immutable

```
s = 'abc'
# s[2] = 'z' # - TypeError: 'str' object does not support item assignment
# print(s)

s = s[0:2] + 'z'
print(s)
```

```
abz
```

String Methods

Example:

```
# Capitalizes first letter of string
print("hello world".capitalize())  # Hello world

# Converts lowercase letters in string to uppercase.
print("hello world".upper()) # HELLO WORLD

# Inverts case for all letters in string.
print("HeLLo WorLD".swapcase())  # hEllO wORld

# Converts all uppercase letters in string to lowercase.
print("Hello WORLD".lower()) # hello world

# Capitalizes first letter of each word of the string
print("hello world".title()) # Hello World
```

```
Hello world
HELLO WORLD
hEllO wORld
hello world
Hello World
```

Example:

```
# Returns the length of the string
print(len("Hello World")) # 11

# Counts how many times str occurs in string.
print("hello world".count("o")) # 2

#or in a substring of string if starting index beg and ending index end are given.
print("hello world".count("o", 0, 5)) # 1
```

```
11
2
1
```

Example:

```
   #  find(str, beg=0 end=len(string)) --> Determine if str occurs in string or in a
   #  substring of string if starting index beg and ending index end are given returns
   #  index if found and -1 otherwise.
print("hello world".find("l"))      # 2
print("hello  world".find("l",5,10)) # 9

# rfind(str, beg=0,end=len(string)) -- > Same as find(), but search backwards in string.
print("hello world".rfind("l"))     # 9
print("hello  world".rfind("l",0,5)) # 3
```

```
2
9
9
3
```

Example:

```
   #  startswith(str, beg=0, end=len(string)) --> Determines if string or a substring of
   #  string (if starting index beg and ending index end are given) starts with substring str;
   #  returns true if so and false otherwise.
print("hello world".startswith("l"))
print("hello world".startswith("l",2,7)) #6th character not included

   #  endswith(suffix, beg=0, end=len(string)) --> Determines if string or a substring of
   #  string (if starting index beg and ending index end are given) ends with suffix;
   #  returns true if so and false otherwise.
print("hello world".endswith("r"))
print("hello world".endswith("r",3,8)) #8th character not included
```

```
False
True
False
False
```

Example:

```
   #  Boolean Functions -

   #  Returns true if string has at least 1 character and all characters are alphanumeric
      and f print("HelloWorld1".isalnum())

   #  Returns true if string has at least 1 character and all characters are alphabetic
      and fal print("HelloWorld".isalpha())

   #  Returns true if string contains only whitespace characters and false otherwise.
print(" ".isspace())

   #  Returns true if string contains only digits and false
      otherwise. print("123".isdigit())

   #  Returns true if string has at least 1 cased character and all cased characters are
      in low print("hello".islower())

   #  Returns true if string has at least 1 cased character and all cased characters are
      in upp print("HELLO".isupper())

   #  Returns true if string is properly "titlecased" and false otherwise.
print("Hello World".istitle())
```

```
True
True
True
True
True
True
True
```

format()

The `format()` method that is available with the string object is very versatile and powerful in formatting strings. Format strings contains curly braces {} as placeholders or replacement fields which gets replaced. We can use positional arguments or keyword arguments to specify the order.

Example:

```
# default(implicit) order
default_order = "{}, {} and {}".format('John','Bill','Sean')
print('\n--- Default Order ---')
print(default_order)

# order using positional argument
positional_order = "{1}, {0} and {2}".format('John','Bill','Sean')
print('\n--- Positional Order ---')
print(positional_order)

# order using keyword argument
keyword_order = "{s}, {b} and {j}".format(j='John',b='Bill',s='Sean')
print(keyword_order)

# formatting integers
print( "Binary representation of {0} is {0:b}".format(12))
'Binary representation of 12 is 1100'

# formatting floats
print( "Exponent representation: {0:e}".format(1566.345))
'Exponent representation: 1.566345e+03'


# round off
print( "One third is: {0:.3f}".format(1/3))
'One third is: 0.333'

# string alignment
print( "|{:<10}|{:^10}|{:>10}|".format('butter','bread','ham'))
'|butter    |  bread   |       ham|'
# formatting integers
print( "Binary representation of {0} is {0:b}".format(12))
'Binary representation of 12 is 1100'

# formatting floats
print( "Exponent representation: {0:e}".format(1566.345))
'Exponent representation: 1.566345e+03'

# round off
print( "One third is: {0:.3f}".format(1/3))
'One third is: 0.333'

# string alignment
print( "|{:<10}|{:^10}|{:>10}|".format('butter','bread','ham'))
'|butter    |  bread   |       ham|'

x = 12.3456789
print('The value of x is %3.2f' %x)
The value of x is 12.35
print('The value of x is %3.4f' %x)
The value of x is 12.3457
```

raw string

r -> makes the string ignore its special characters like /n in above case

Example:

```
var = r'C:\folder\name'
var
```

```
'C:\\folder\\name'
```

Ex. WAP to reverse a string entered by user

```
rev = "".join(reversed(input("Enter string : ")))
print(rev)
```

```
Enter string : mumbai
iabmum
```