Module 03. Strings

September 26, 2018

1 Strings

Strings are amongst the most popular types in Python.

Python has a built-in string class named "str" with many useful features.

String literals can be enclosed by either single or double, although single quotes are more commonly used.

A string literal can span multiple lines, use triple quotes to start and end them. You can use single quotes too, but there must be a backslash at the end of each line to escape the newline.

1.0.1 - String is a Character Array

```
In [8]: var = "Python"
        print(var)
        print(var[2]) # t
        print(var[-1]) # n
        print(var[0:3]) #Pyt
        print(var[-3:])
        print(var[:4])
        print(var[4:])
        print(var[:-1])
        print(var[1:-1])
```

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

1.0.2 - Updating Values in String

```
In [11]: var1 = input("Enter a string : ")
         # var2 = var1[1:-1]
```

```
        var3 = "Hello " + var1
        print(var3)
```

```
Enter a stringmumbai
Hello mumbai
```

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

```
In [13]: 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
```

1.1 Strings are Immutables

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

        s = s[0:2] + 'z'

        print(s)
```

```
abz
```

1.2 String Methods

Capitalizes first letter of string

```
In [12]:  print("hello world".capitalize())
```

```
Hello world
```

Converts lowercase letters in string to uppercase.

```
In [13]: print("hello world".upper())
```

```
HELLO WORLD
```

Inverts case for all letters in string.

```
In [14]:  print("HeLLo WorLD".swapcase())
```

```
hEllO wORld
```

Converts all uppercase letters in string to lowercase.

```
In [15]:  print("Hello WORLD".lower())
```

```
hello world
```

Capitalizes first letter of each word of the string

```
In [16]: print("hello world".title())
```

```
Hello World
```

Returns the length of the string

```
In [17]: print(len("Hello World"))
```

```
11
```

Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.

```
In [14]: print("hello world".count("o",0,5))
```

```
1
```

- Bool returning string methods

```
In [5]: # Returns true if string has at least 1 character and all characters are alphanumeric
        print("HelloWorld1".isalnum())

        # Returns true if string has at least 1 character and all characters are alphabetic an
        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 i
print("hello".islower())

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

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

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

- startswith() and endswith() 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.

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.

```
In [18]: mystr = "hello world"

         print(mystr.startswith("l",2,7)) #6th character not included

         print(mystr.endswith("r",3,8))
```

```
True
False
```

- find() "".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.

rfind(str, beg=0,end=len(string)) – > Same as find(), but search backwards in string.

```
In [15]: mystr = "hello world"

         print(mystr.find("l",5,10))

         print(mystr.rfind("l",0,5))
```

```
2
9
```

```
In [11]: # join(seq) # Merges (concatenates) the string representations of elements in sequenc
         print(", ".join(["tiger","lion","wolf"]))
         print("$".join("abc"))


         mystr= "ab:cd:e:f:gh"

         # Splits string according to delimiter str from behind(space if not provided) and ret
         print(mystr.rsplit(":",2))

         # Splits string according to delimiter str (space if not provided) and returns list o
         print(mystr.split(":",2))

         # replace(old, new , max]) -->  Replaces all occurrences of old in string with new or
         print(mystr.replace(":","*",2))

         # strip(), rstrip(), lstrip() --> removes trailing whitespaces
```

```
tiger, lion, wolf
a$b$c
```

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

```
In [9]: var = r'C:\folder\name'
        var
```

```
Out[9]: 'C:\\folder\\name'
```

Ex. WAP to reverse a string entered by user

```
In [4]: rev = "".join(reversed(input("Enter string : ")))
        print(rev)
```

```
Enter string : mumbai
iabmum
```

Ex. WAP to check entered string is Palindrome or not.

```
In [31]: var = input('Enter a string : ')
         print(var, "is" if var==var[::-1] else "is not","a plaindrome.")  ##here print functi
```

```
Enter a string : abc
abc is not  a plaindrome.
```

```
In [28]: var = 'madam'
         result = var+" is a Palindrome " if var==var[::-1] else var+" is not a plaindrome"
         print(result)  ##here var is serperately added to seprate string so output can be cha
```

```
madamis a Palindrome
```

- format()

```
In [33]: a = 10
         b = 20
         c = a+b
         print(a,"+",b,"=",c)  ## sep param of print() i providing the extra space
```

```
10 + 20 = 30
```

```
In [5]: a, b, c = 10, 20, 30

        # for concatenation we have to provide extra space also convert the int values to str
        result = str(a) + " + " + str(b) + " = " + str(c)

        # format() does it all for us
        #format is method used for string eg.   "".format
        result = "{} + {} = {}".format(a,b,c)

        print(result)
```

```
10 + 20 = 30
```