**String Handling and Function**

To share an important note that you can also use triple quotes to create strings. However, programmers use them to mark multi-line strings and doc strings.

# Python string examples - all assignments are identical.

String\_var = 'Python'

String\_var = "Python"

String\_var = """Python"""

# with Triple quotes Strings can extend to multiple lines

String\_var = """ This document will help you to

explore all the concepts

of Python Strings!!! """

# Replace "document" with "tutorial" and store in another variable

substr\_var = String\_var.replace("document", "notes")

print (substr\_var)

sample\_str = 'Python String'

print (sample\_str[0]) # return 1st character

# output: P

print (sample\_str[-1]) # return last character

# output: g

print (sample\_str[-2]) # return last second character

# output: n

#### Slice A String In Python

To retrieve a range of characters in a String, we use ‘slicing operator,’ the colon ‘:’ sign. With the slicing operator, we define the range as [a:b]. It’ll let us print all the characters of the String starting from index ‘a’ up to char at index ‘b-1’. So the char at index ‘b’ is not a part of the output.

sample\_str = 'Python String'

print (sample\_str[3:5]) #return a range of character

# ho

print (sample\_str[7:]) # return all characters from index 7

# String

print (sample\_str[:6]) # return all characters before index 6

# Python

print (sample\_str[7:-4])

# St

**Conversion Functions**

**capitalize() –** Returns the string with the first character capitalized and rest of the characters in lower case.

var = 'PYTHON'

print (var.capitalize())

# Python

**lower() –** Converts all the characters of the String to lowercase

var = 'TechBeamers'

print (var.lower())

# techbeamers

**upper() –** Converts all the characters of the String to uppercase

var = 'TechBeamers'

print (var.upper())

# TECHBEAMERS

**swapcase() –** Swaps the case of every character in the String means that lowercase characters got converted to uppercase and vice-versa.

var = 'TechBeamers'

print (var.swapcase())

# tECHbEAMERS

**title() –** Returns the ‘titlecased’ version of String, which means that all words start with uppercase and the rest of the characters in words are in lowercase.

var = 'welcome to Python programming'

print (var.title())

# Welcome To Python Programming

**count( str[, beg [, end]]) –** Returns the number of times substring ‘str’ occurs in the range [beg, end] if beg and end index are given else the search continues in full String Search is case-sensitive.

var='TechBeamers'

str='e'

print (var.count(str))

# 3

var1='Eagle Eyes'

print (var1.count('e'))

# 2

var2='Eagle Eyes'

print (var2.count('E',0,5))

# 1

#### Comparison Functions

**islower() –** Returns ‘True’ if all the characters in the String are in lowercase. If any of the char is in uppercase, it will return False.

var='Python'

print (var.islower())

# False

var='python'

print (var.islower())

# True

**isupper() –** Returns ‘True’ if all the characters in the String are in uppercase. If any of the char is in lowercase, it will return False.

var='Python'

print (var.isupper())

# False

var='PYTHON'

print (var.isupper())

# True

**isdecimal() –** Returns ‘True’ if all the characters in String are decimal. If any character in the String is of other data-type, it will return False.

Decimal characters are those from the Unicode category Nd.

num=u'2016'

print (num.isdecimal())

# True

**isdigit() –** Returns ‘True’ for any char for which isdecimal() would return ‘True and some characters in the ‘No’ category. If there are any characters other than these, it will return False’.

Precisely, digits are the characters for which Unicode property includes: Numeric\_Type=Digit or Numeric\_Type=Decimal.

For example, superscripts are digits, but fractions not.

print ('2'.isdigit())

# True

print ('²'.isdigit())

# True

**isnumeric() –** Returns ‘True’ if all the characters of the Unicode String lie in any one of the categories Nd, No, and NI.

If there are any characters other than these, it will return False.

Precisely, Numeric characters are those for which Unicode property includes: Numeric\_Type=Digit, Numeric\_Type=Decimal or Numeric\_Type=Numeric.

num=u'2016'

print (num.isnumeric())

# True

num=u'year2016'

print (num.isnumeric())

# False

**isalpha() –** Returns ‘True’ if String contains at least one character (non-empty String), and all the characters are alphabetic, ‘False’ otherwise.

print ('python'.isalpha())

# True

print ('python3'.isalpha())

# False

**isalnum() –** Returns ‘True’ if String contains at least one character (non-empty String), and all the characters are either alphabetic or decimal digits, ‘False’ otherwise.

print ('python'.isalnum())

# True

print ('python3'.isalnum())

# True

#### Padding Functions

**rjust(width[,fillchar]) –** Returns string filled with input char while pushing the original content on the right side.

By default, the padding uses a space. Otherwise, ‘fillchar’ specifies the filler character.

var='Python'

print (var.rjust(10))

# Python

print (var.rjust(10,'-'))

# ----Python

**ljust(width[,fillchar]) –** Returns a padded version of String with the original String left-justified to a total of width columns

By default, the padding uses a space. Otherwise, ‘fillchar’ specifies the filler character.

var='Python'

print (var.ljust(10))

# Python

print (var.ljust(10,'-'))

# Python----

**center(width[,fillchar]) –** Returns string filled with the input char while pushing the original content into the center.

By default, the padding uses a space. Otherwise, ‘fillchar’ specifies the filler character.

var='Python'

print (var.center(20))

# Python

print (var.center(20,'\*'))

# \*\*\*\*\*\*\*Python\*\*\*\*\*\*\*

**zfill(width) –** Returns string filled with the original content padded on the left with zeros so that the total length of String becomes equal to the input size.

If there is a leading sign (+/-) present in the String, then with this function, padding starts after the symbol, not before it.

var='Python'

print (var.zfill(10))

# 0000Python

var='+Python'

print (var.zfill(10))

# +000Python

#### String Substitution Functions

**replace(old,new[,count]) –** Replaces all the occurrences of substring ‘old’ with ‘new’ in the String.

If the count is available, then only ‘count’ number of occurrences of ‘old’ will be replaced with the ‘new’ var.

Where old =substring to replace, new =substring

var='This is a good example'

str='was'

print (var.replace('is',str))

# Thwas was a good exampleprint (var.replace('is',str,1))

# Thwas is a good example

**split([sep[,maxsplit]]) –** Returns a list of substring obtained after splitting the String with ‘sep’ as a delimiter.

Where, sep= delimiter, the default is space, maxsplit= number of splits to be done

var = "This is a good example"

print (var.split())

# ['This', 'is', 'a', 'good', 'example']print (var.split(' ', 3))

# ['This', 'is', 'a', 'good example']

**splitlines(num) –** Splits the String at line breaks and returns the list after removing the line breaks.

Where num = if this is a positive value. It indicates that line breaks will appear in the returned list.

var='Print new line\nNextline\n\nMove again to new line'

print (var.splitlines())

# ['Print new line', 'Nextline', '', 'Move again to new line']print (var.splitlines(1))

# ['Print new line\n', 'Nextline\n', '\n', 'Move again to new line']

**join(seq) –** Returns a String obtained after concatenating the sequence ‘seq’ with a delimiter string.

Where: the seq= sequence of elements to join

seq=('ab','bc','cd')

str='='

print (str.join(seq))

# ab=bc=cd

#### Misc String Functions

**lstrip([chars]) –** Returns a string after removing the characters from the beginning of the String.

Where: Chars=this is the character to be trimmed from the String.

The default is whitespace character.

var=' This is a good example '

print (var.lstrip())

# This is a good example

var='\*\*\*\*\*This is a good example\*\*\*\*\*'

print (var.lstrip('\*'))

# This is a good example\*\*\*\*\*\*\*\*\*\*

**rstrip() –** Returns a string after removing the characters from the End of the String.

Where: Chars=this is the character to be trimmed from the String. The default is whitespace character.

var=' This is a good example '

print (var.rstrip())

# This is a good example

var='\*\*\*\*\*This is a good example\*\*\*\*\*'

print (var.lstrip('\*'))

# \*\*\*\*\*This is a good example

**rindex(str[,i [,j]]) –** Searches for ‘str’ in the complete String (if i and j not defined) or in a sub-string of String (if i and j are defined). This function returns the last index where ‘str’ is available.

If ‘str’ is not there, then it raises a ValueError exception.

Where: i=search starts from this index, j=search ends at this index.

var='This is a good example'

str='is'

print (var.rindex(str,0,10))

# 5

print (var.rindex(str,10))

# ValueError: substring not found

**len(string) –** Returns the length of given String

var='This is a good example'

print (len(var))

# 22

##### **List Of Format Symbols**

Following is the table containing the complete list of symbols that you can use with the ‘%’ operator.

**Symbol Conversion**

%c

character

%s

string conversion via str() before formatting

%i

signed decimal integer

%d

signed decimal integer

%u

unsigned decimal integer

%o

octal integer

%x

hexadecimal integer (lowercase letters)

%X

hexadecimal integer (UPPER-case letters)

%e

exponential notation (with lowercase ‘e’)

%E

exponential notation (with UPPER-case ‘E’)

%f

floating-point real number

%g

the shorter of %f and %e

%G

the shorter of %f and %E