**Best Of Python Strings, Functions And Examples**

**able of Content.**

1. **Creating Python Strings.**
2. [**Accessing Characters**](http://www.techbeamers.com/python-strings-functions-and-examples/#hd2)**in Python Strings.**
3. [**Modifying/Deleting**](http://www.techbeamers.com/python-strings-functions-and-examples/#hd3)**Python Strings.**
4. **Python**[**String Operators**](http://www.techbeamers.com/python-strings-functions-and-examples/#hd4)**.**
5. **String**[**Formatting operators**](http://www.techbeamers.com/python-strings-functions-and-examples/#hd5)**.**
   * **Escape Characters.**
   * **Python Format Characters.**
6. [**Unicode String**](http://www.techbeamers.com/python-strings-functions-and-examples/#hd6)**.**
7. [**Built-in String Functions**](http://www.techbeamers.com/python-strings-functions-and-examples/#hd7)**for the following,**
   * **Python String Conversion.**
   * **Python Compare String.**
   * **Python String Padding.**
   * **Python String Replace.**

## 1- Creating Python Strings.

**Creating Strings is easy, and it is done simply by enclosing the characters in single or double quotes. The strings in Python consider both single and double quotes as the same. In the following example, we are providing the different ways for initializing strings and showcasing the use of Python string replace function.**

**Just 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 docstrings.**

**# 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", "tutorial")**

**print (substr\_var)**

## 2- Accessing Characters In Python Strings.

**You just need to know the index of a character to retrieve it from the String. However, the range of characters can be accessed with the slicing feature.**

**Like the most programming languages, Python allows to index from the zeroth position in Strings. But it also supports negative indexes. Index of ‘-1’ represents the last character of the String. Similarly using ‘-2’ we can access the penultimate element of the string and so on.**

| **P** | **Y** | **T** | **H** | **O** | **N** | **–** | **S** | **T** | **R** | **I** | **N** | **G** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **-13** | **-12** | **-11** | **-10** | **-9** | **-8** | **-7** | **-6** | **-5** | **-4** | **-3** | **-2** | **-1** |

**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**

**To retrieve a range of characters in a String we use ‘slicing operator’, the colon ‘:’. 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 character at index ‘b-1’. So the character 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**

**Next, we have a no. of Python tutorials/quizzes/interview questions on this blog. If you like to try them, then refer any of the posts listed below.**

**Suggested Reading:**

**☛**[**Python Programming Interview Questions Part-1**](http://www.techbeamers.com/python-programming-interview-questions-with-answers/)**.**

[**TOC**](http://www.techbeamers.com/python-strings-functions-and-examples/#toc)

### Understanding Invalid Python String Usage And Error Codes.

**1- If we try to retrieve characters at out of range index then ‘IndexError’ exception will be raised.**

**sample\_str = "Python Supports Machine Learning."**

**print (sample\_str[1024]) #index must be in range**

**# IndexError: string index out of range**

**2- String index must be of integer data type. You should not use a float or any other data type for this purpose. Otherwise, the Python subsystem will flag a TypeError exception as it detects a data type violation for the string index.**

**sample\_str = "Welcome post"**

**print (sample\_str[1.25]) #index must be an integer**

**# TypeError: string indices must be integers**

## 3- Modifying/Deleting Python Strings.

**Python Strings are by design immutable. It suggests that once a String binds to a variable; it can’t be modified. If you want to update the String simply re-assign a new String value to the same variable.**

**sample\_str = 'Python String'**

**sample\_str[2] = 'a'**

**# TypeError: 'str' object does not support item assignment**

**sample\_str = 'Programming String'**

**print (sample\_str)**

**# Output=> Programming String**

**Similarly, we cannot modify the Strings by deleting some characters from it. Instead, we can remove the Strings altogether by using ‘del’ command.**

**sample\_str = "Python is the best scripting language."**

**del sample\_str[1]**

**# TypeError: 'str' object doesn't support item deletion**

**del sample\_str**

**print (sample\_str)**

**# NameError: name 'sample\_str' is not defined**

**Suggested Reading:**

**☛ [Python Programming Interview Questions Part-2](http://www.techbeamers.com/python-interview-questions-and-answers-second-edition/" \t "_blank).**

**[TOC](http://www.techbeamers.com/python-strings-functions-and-examples/" \l "toc)**

## 4- Python String Operators.

| **Operator** | **Operation** | **Description** | **Example Code** |
| --- | --- | --- | --- |
| **+** | **Concatenation** | **Combining two Strings into one.** | **var1 = ‘Python’ var2 = ‘String’ print (var1+var2) # PythonString** |
| **\*** | **Repetition** | **Creates new String by repeating the String given number of times.** | **var1 = ‘Python’ print (var1\*3) # PythonPythonPython** |
| **[ ]** | **Slicing** | **Prints the character at given index.** | **var1 = ‘Python’ print (var1[2]) # t** |
| **[ : ]** | **Range Slicing** | **Prints the characters present at the given range .** | **var1 = ‘Python’ print (var1[2:5]) # tho** |
| **in** | **Membership** | **Returns ‘True’ value if character is present in the given String.** | **var1 = ‘Python’ print (‘n’ in var1) # True** |
| **not in** | **Membership** | **Returns ‘True’ value if character is not present in given String.** | **var1 = ‘Python’ print (‘N’ not in var1) # True** |
| **for** | **Iterating** | **Using for we can iterate through all the characters of the String.** | **var1 = 'Python'**  **for var in var1:**  **print (var)**  **# P**  **# y**  **# t**  **# h**  **# o**  **# n** |
| **r/R** | **Raw String** | **Used to ignore the actual meaning of Escape characters inside a string. For this we add ‘r’ or ‘R’ in front of the String.** | **print (r’\n’) # \n print (R’\n’) # \n** |

**[TOC](http://www.techbeamers.com/python-strings-functions-and-examples/" \l "toc)**

| **Escape Character** | **Used To Print** |
| --- | --- |
| **\\** | **Backslash (\)** |
| **\”** | **Double-quote (“)** |
| **\a** | **ASCII bell (BEL)** |
| **\b** | **ASCII backspace (BS)** |
| **\cx or \Cx** | **Control-x** |
| **\f** | **ASCII Form feed (FF)** |
| **\n** | **ASCII linefeed (LF)** |
| **\N{name}** | **Character named name in the Unicode database (Unicode only)** |
| **\r** | **Carriage Return (CR)** |
| **\t** | **Horizontal Tab (TAB)** |
| **\uxxxx** | **Character with 16-bit hex value xxxx (Unicode only)** |
| **\Uxxxxxxxx** | **Character with 32-bit hex value xxxxxxxx (Unicode only)** |
| **\v** | **ASCII vertical tab (VT)** |
| **\ooo** | **Character with octal value ooo** |
| **\xnn** | **Character with hex value nn where n can be anything from the range 0-9, a-f or A-F.** |

| **Format Symbol** | **Conversion** |
| --- | --- |
| **%c** | **character** |
| **%s** | **string conversion via str() prior to 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** |

| **Function Name** | **Description** | **Example Code** |
| --- | --- | --- |
| **capitalize()** | **Returns the String with 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 are changed 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 rest of the characters in the 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 range [beg, end] if beg and end index are given. If it is not given then substring is searched in whole 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** |

**[TOC](http://www.techbeamers.com/python-strings-functions-and-examples/" \l "toc)**

| **Function Name** | **Description** | **Example Code** |
| --- | --- | --- |
| **islower()** | **Returns ‘True’ if all the characters in the String are in lowercase. If any one character 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 one character 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 anyone character in the String is of other data-type, it will return ‘False’. Decimal characters are those from Unicode category ‘Nd’. Complete list of ‘Nd’ is present at following link: http://www.fileformat.info/info/unicode/category/Nd/list.htm** | **num=u’2016′ print (num.isdecimal()) # True** |
| **isdigit()** | **Returns ‘True’ for any character for which isdecimal() would return ‘True and some characters in ‘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. Complete list of ‘No’ is present at following link: http://www.fileformat.info/info/unicode/category/No/list.htm** | **print (‘2’.isdigit()) # True print (‘²’.isdigit()) # True** |

#### 7.2.2- Python Strings Comparison Functions Section-2.

| **Function Name** | **Description** | **Examples** |
| --- | --- | --- |
| **isnumeric()** | **Returns ‘True’ if all the characters of the Unicode String lie in any one of the category ‘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. Complete list of ‘NI’ is present at following link: http://www.fileformat.info/info/unicode/category/Nl/list.htm** | **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** |

### 7.3- Python String Padding Functions.

| **Function Name** | **Description** | **Examples** |
| --- | --- | --- |
| **rjust(width[,fillchar])** | **Returns a padded version of String with the original String right-justified to a total of width columns. By default, Padding is done by using 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, Padding is done by using space. Otherwise ‘fillchar’ specifies the filler character.** | **var=’Python’ print (var.ljust(10)) # Python print (var.ljust(10,’-‘)) # Python—-** |
| **center(width[,fillchar])** | **Returns a padded version of String with the original String moved to center to a total of width columns. By default, Padding is done by using space. Otherwise ‘fillchar’ specifies the filler character.** | **var=’Python’ print (var.center(20)) # Python print (var.center(20,’\*’)) # \*\*\*\*\*\*\*Python\*\*\*\*\*\*\*** |
| **zfill(width)** | **Returns a padded version of String with the original String padded on the left with zeros so that total length of String becomes equal to width. If there is a leading sign (+/-) present in the String, then with this function padding is done after the sign, not before it.** | **var=’Python’ print (var.zfill(10)) # 0000Python var=’+Python’ print (var.zfill(10)) # +000Python** |

**[TOC](http://www.techbeamers.com/python-strings-functions-and-examples/" \l "toc)**

### 7.4- Python String Functions.

#### 7.4.1- Functions To Find A String In Python.

| **Function Name** | **Description** | **Example Code** |
| --- | --- | --- |
| **find(str [,i [,j]])** | **Searches for ‘str’ in 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 index if ‘str’ is found else returns ‘-1’. where, i=search starts from this index j=search ends at this index.** | **var=”Tech Beamers” str=”Beam” print (var.find(str)) # 5 var=”Tech Beamers” str=”Beam” print (var.find(str,4)) # 5 var=”Tech Beamers” str=”Beam” print (var.find(str,7)) # -1** |
| **index(str[,i [,j]])** | **This is same as ‘find’ method. The only difference is that it raises ‘ValueError’ exception if ‘str’ is not found.** | **var=’Tech Beamers’ str=’Beam’ print (var.index(str)) # 5 var=’Tech Beamers’ str=’Beam’ print (var.index(str,4)) # 5 var=’Tech Beamers’ str=’Beam’ print (var.index(str,7)) # ValueError: substring not found** |
| **rfind(str[,i [,j]])** | **This is same as find() just that this function returns the last index where ‘str’ is found. If ‘str’ is not found it returns ‘-1’.** | **var=’This is a good example’ str=’is’ print (var.rfind(str,0,10)) # 5 print (var.rfind(str,10)) # -1** |
| **count(str[,i [,j]])** | **Returns the number of occurrences of substring ‘str’ in the String. Searches for ‘str’ in complete String (if i and j not defined) or in a sub-string of String (if i and j are defined). where, i=search starts from this index j=search ends at this index.** | **var=’This is a good example’ str=’is’ print (var.count(str)) # 2 print (var.count(str,4,10)) # 1** |

**[TOC](http://www.techbeamers.com/python-strings-functions-and-examples/" \l "toc)**

#### 7.4.2- Functions To Replace A String In Python.

| **Function Name** | **Description** | **Example Code** |
| --- | --- | --- |
| **replace(old,new[,count])** | **Replaces all the occurrences of substring ‘old’ with ‘new’ in the String. If ‘count’ is defined then only ‘count’ number of occurrences of ‘old’ will be replaced with ‘new’. where, old =substring to be replaced new =substring that will replace the old count =number of occurrences of old that will be replaced with new.** | **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 delimiter. where, sep= delimiter, 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 positive value. It indicates that line breaks to be included 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, seq= sequence of elements to be joined** | **seq=(‘ab’,’bc’,’cd’) str=’=’ print (str.join(seq)) # ab=bc=cd** |

#### 7.4.3- Misc String Handling Functions In Python.

| **Function Name** | **Description** | **Example Code** |
| --- | --- | --- |
| **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. 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. 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 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 found. If ‘str’ is not found it raises ‘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)) # 5print (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** |

**[TOC](http://www.techbeamers.com/python-strings-functions-and-examples/" \l "toc)**