**String**

In Python, a string is a sequence of characters enclosed in either single quotes (') or double quotes (") or triple quotation (“””). Strings are used to store and manipulate text. There is not a char data type as in C and C++ in python.

|  |
| --- |
| string1=”Bhaskar”  string2="AA"  string3="""True"""  string4='''23'''  print(string1)  print(type(string2))  print(type(string3))  print(string4)  Output :  Bhaskar  <class 'str'>  <class 'str'>  23 |

**Multiline string:**

Triple quotation (‘’’) is used for multiline string.

|  |
| --- |
| multi\_line\_string='''My  name  is  Bhaskar'''  print(multi\_line\_string)    Output :  My  name  is  Bhaskar |

**Escape character:**

Special characters can be used inside strings using a backslash (\).

* \n: Newline
* \t: Tab
* \\: Backslash
* \': Single quote
* \": Double quote

|  |
| --- |
| escaped\_string = 'Hello\nWorld'  Output:  Hello  World |

Accessing the characters of a string: The two ways to access characters in string are

* Indexing
* Slicing

**Indexing**

Indexing is used to access individual elements in a sequence by their position (or index). In Python, the index starts at 0, meaning the first element is at position 0, the second at 1, and so on.

|  |
| --- |
| string="My name is Bhaskar"  print(string[0])  print(string[-17])  Output  M  M |

|  |
| --- |
| string="I am Bhaskar"  i=0  for x in string:      print(f"The character {x} is present at positive {i} and negative {i-len(string)} index ")      i = i+1    Output:  The character I is present at positive 0 and negative -12 index  The character is present at positive 1 and negative -11 index  The character a is present at positive 2 and negative -10 index  The character m is present at positive 3 and negative -9 index  The character is present at positive 4 and negative -8 index  The character B is present at positive 5 and negative -7 index  The character h is present at positive 6 and negative -6 index  The character a is present at positive 7 and negative -5 index  The character s is present at positive 8 and negative -4 index  The character k is present at positive 9 and negative -3 index  The character a is present at positive 10 and negative -2 index  The character r is present at positive 11 and negative -1 index |

**Slicing:**

Slicing is used to access a subset (or slice) of elements from a sequence. Slicing allows us to extract a range of elements by specifying a start, stop, and step (optional).

Syntax: sequence[start:stop:step]

* start: The index where the slice starts (inclusive).
* stop: The index where the slice ends (exclusive).
* step: The number of steps between each index (optional).

We can also use negative indices in slicing to count from the end of the sequence.

|  |
| --- |
| s="Hello I am Bhaskar"  print(s[0:5:1])  print(s[0:5:2])  print(s[::-1])  Output:  Hello  Hlo  raksahB ma I olleH |

**Mathematical operations for string**

For concatenation operation, all operands must be string. For repetition operation, one operand must be integer and another must be string.

|  |
| --- |
| s="Python"  print("The concatenation operation of string with this line with s is : "+s)  print("Repetition of string : "\*3)    Output  The concatenation operation of string with this line with s is: Python  Repetition of string: Repetition of string : Repetition of string |

|  |
| --- |
| **Length of string**:  s="Python"  print(len(s))  print(s.\_\_len\_\_())    Output  6  6 |

Membership operator in string: The membership operators in string are: ‘in’ and ‘not in’.

|  |
| --- |
| s="Bhaskar"  print("B" in s)  print("B" not in s)    Output  True  False |

|  |
| --- |
| **Comparison operator and Equality operator:**  print("Python"=="Python")  print("Python"!="Python")  print("Python"<"python")  print("Python"<="python")  print("Python">"python")  print("Python">="python")  print("python"<="pple")  print(chr(98)) #To find character of ascii 98  print(ord('A')) #To find ascii of character A    Output  True  False  True  True  False  False  False  b  65 |

Strip function is used to remove the spacing of the string. There are three functions to remove space located at front and back of a string of strip.

|  |
| --- |
| name=" Bhakar "  print("Name removing space at right:",name.rstrip())  print("Name removing space at left:",name.lstrip())  print("Name with stripping at both front and back:",  name.strip())  Output:  Name removing space at right: Bhakar  Name removing space at left: Bhakar  Name with stripping at both front and back: Bhakar |

A **substring** is simply a part of a string, extracted using **indexing** or **slicing**. In Python, we can extract a substring by specifying the portion of the string we want to extract.

To find an index of substring in a string, there are four functions illustrated by example below.

* find() : The find() method returns the lowest index of the first occurrence of the specified substring. If the substring is not found, it returns -1.
* rfind() : The rfind() method returns the highest index of the last occurrence of the specified substring. If the substring is not found, it returns -1.
* index():The index() method is similar to find(), but instead of returning -1 when the substring is not found, it raises a ValueError
* rindex():The rindex() method is similar to rfind(), but instead of returning -1 when the substring is not found, it raises a ValueError.

|  |
| --- |
| a="We are having fun in the office"  print("find of e: ",a.find("o"))  print('rfind of e:',a.rfind("o"))  print('rfind of e:',a.rfind("o",1,10))#to find in index between 1 and 10 and if not found gives -1  print("index of e: ",a.index("o"))  print('rindex of e:',a.rindex("o"))  print('rindex of e:',a.rindex("o",1,10))  print("find where character not present: ",a.find("e",10,15))  print("index where character not present:",a.index("e",10,15))  output:  find of e: 25  rfind of e: 25  rfind of e: -1  index of e: 25  rindex of e: 25  Traceback (most recent call last):  File "e:\intern\string\_operation.py", line 24, in <module>  print('rindex of e:',a.rindex("o",1,10))  ^^^^^^^^^^^^^^^^^^  ValueError: substring not found |

|  |
| --- |
| a="We are having a lot of fun here"  print("Number of e: ",a.count("e"))  print("Number of e between 5 and 10: ",a.count("e",5,10))  output:  Number of e: 4  Number of e between 5 and 10: 1 |

|  |
| --- |
| **Replace substring:** Replacing does not change string but only changes while printing.  a="Learning python is always interesting."  print("replacing interesting with beautiful:",a.replace("interesting","beautiful"))  print(a)    Output  replacing interesting with beautiful: Learning python is always beautiful.  Learning python is always interesting. |

|  |
| --- |
| **Spliting of string:**  a="Python is easy for senior developers.".split()  print("after splitting: ",a)  for i in a:  print(i,end="\n")    date="2059-10-24"  split\_date=date.split("-")  print("Date after splitting: ",split\_date)    Output  after splitting: ['Python', 'is', 'easy', 'for', 'senior', 'developers.']  Python  is  easy  for  senior  developers.  Date after splitting: ['2059', '10', '24'] |

|  |
| --- |
| **Joining of list and tuple for creating string**  lists=['Python', 'is', 'easy', 'for', 'senior',  'developers.']  a=" ".join(lists)  print("list after joining: ",a)  tuples=('Python', 'is', 'easy', 'for', 'senior',  'developers.')  b="-".join(tuples)  print("tuples after joining with -: ",b)    Output  list after joining: Python is easy for senior developers.  tuples after joining with -: Python-is-easy-for-senior-developers. |
| **Changing case of string**  a="Python is EASY for senior developers."  print(".upper function: ",a.upper())  print(".lower function: ",a.lower())  print(".swapcase function: ",a.swapcase())  print(".title function: ",a.title())  print(".capitalized function: ",a.capitalize())    Output  .upper function: PYTHON IS EASY FOR SENIOR DEVELOPERS.  .lower function: python is easy for senior developers.  .swapcase function: pYTHON IS easy FOR SENIOR DEVELOPERS.  .title function: Python Is Easy For Senior Developers.  .capitalized function: Python is easy for senior developers. |

|  |
| --- |
| **Checking type of character**  print("Learning python 3.x".isalnum())  print("Learning python 3x".isalnum())  print("Learningpython3x".isalnum())  print("Learning".isalpha())  print("1234".isdigit())  print("LEARNING".isupper())  print("learnIng".isupper())  print("Learning".islower())  print("learning".islower())  print(" ".isspace())  print("Learning python".istitle())  print("Learning Python".istitle())    Output  False  False  True  True  True  True  False  False  True  True  False  True |

|  |
| --- |
| **Formating of string**  name="Bhaskar"  age=21  print(f"My name is {name} and age is {age}.")  print("My name is {0} and age is {1}.".format(name,age))  print("My name is {x} and age is {y}.".format(x=name,y=age))    print("The integer is {}".format(123))  print("The integer is {:d}".format(123))  print("The integer is {:5d}".format(123))  print("The integer is {:05d}".format(123))  Output:  My name is Bhaskar and age is 21.  My name is Bhaskar and age is 21.  My name is Bhaskar and age is 21.  The integer is 123  The integer is 123  The integer is 123  The integer is 00123 |

|  |
| --- |
| **Methods and function of string:**  print(dir(str))  Output  ['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_',  '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_getnewargs\_\_', '\_\_getstate\_\_',  '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_',  '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_',  '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'capitalize',  'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format\_map',  'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric',  'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition',  'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',  'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill'] |