Unit NO. – 4

**STRINGS**

Prepared by

PROF. PRASHANT M. SURYAVANSHI
[Email – sprashant1234@gmail.com]

PROF. ARTI P. SURYAVANSHI
[Email - artips15@gmail.com]
UNIT IV STRING

Concept of String:

A string is a sequence of characters. The string is a sequence of Unicode character in Python.
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.

\[
\text{e.g.:} \\
\text{my\_string} = 'Hello' \\
\text{my\_string} = "Hello" \\
\text{my\_string} = ""Hello"" \\
\text{my\_string} = """"Hello, welcome to the"
\]

Accessing String:

We can access individual characters using indexing and a range of characters using slicing. 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.

\[
\text{str} = 'programing' \\
\text{print('str=', str)} \\
\text{O/P:- programing} \\
\text{print(str[0] =', str[0])} \\
\text{O/P:- p} \\
\text{print(str[-1] =', str[-1])} \\
\text{O/P:- g} \\
\text{print(str[1:5] =', str[1:5])} \\
\text{O/P:- rogr} \\
\text{print(str[5:-2] =', str[5:-2])} \\
\text{O/P:- ami}
\]

String operation:

1) Finding Length of String:

len(string)
This function Returns the length of the string

\[
\text{e.g.:} \\
\text{msg} = 'Goodbye' \\
\text{print(msg)} \\
\text{len(msg)} \\
\text{O/P:- Goodbye} \\
\text{7}
\]

2) Concatenation of Strings:

Joining of two or more strings into a single one is called concatenation. The + operator does this in Python. Simply writing two string literals together also concatenates them.

\[
\text{str1} = 'Hello' \\
\text{str2} = 'World!' \\
\text{print(str1 + str2)} \\
\text{O/P:- HelloWorld}
\]

3) Appending String:

We can use += operator to append some string to already existing string.

\[
\text{e.g.:-} \\
\text{str} = "python program is fun" \\
\text{str}+= "we enjoy it very much" \\
\text{print(str)} \\
\text{O/P:- python program is fun. We enjoy it very much}
\]
4) Multiply String:-
The * operator can be used to repeat the string for a given number of times.

   e.g.:-
   print('str1 * 3 =', str1 * 3)
   O/P:-‘Hello Hello Hello’

Strings are Immutable:-
String are immutable i.e., we cannot change the existing string.
   e.g.:-
   msg = “Good Morning”
   msg[0] = ‘g’
   Type Error: ‘str’ object does not support item assignment
   To make desired change we need to take new string and manipulate it as per our requirement
   e.g.:-
   msg = “Good Morning”
   new_msg = ‘g’ + msg[1:]
   print(new_msg)
   O/P:-
   Good Morning

String Formatting Operator:-
One of Python’s coolest features is the string format operator %. Following is a simple example –
   e.g.:-
   print “My name is %s and weight is %d kg!” % (‘Zara’, 21)
   O/P:-
   My name is Zara and weight is 21 kg!

Here is the list of complete set of symbols which can be used along with % –

# This prints out ”John is 23 years old.”
name = “John”
age = 23
print(“%s is %d years old.” % (name, age))
O/P:-John is 23 years old

<table>
<thead>
<tr>
<th>Format Symbol</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>%c</td>
<td>character</td>
</tr>
<tr>
<td>%s</td>
<td>string conversion via str() prior to formatting</td>
</tr>
<tr>
<td>%i</td>
<td>signed decimal integer</td>
</tr>
<tr>
<td>%d</td>
<td>signed decimal integer</td>
</tr>
<tr>
<td>%u</td>
<td>unsigned decimal integer</td>
</tr>
<tr>
<td>%o</td>
<td>octal integer</td>
</tr>
<tr>
<td>%x</td>
<td>hexadecimal integer (lowercase letters)</td>
</tr>
<tr>
<td>%X</td>
<td>hexadecimal integer (UPPERcase letters)</td>
</tr>
<tr>
<td>%e</td>
<td>exponential notation (with lowercase ‘e’)</td>
</tr>
<tr>
<td>%E</td>
<td>exponential notation (with UPPERcase ‘E’)</td>
</tr>
<tr>
<td>%f</td>
<td>floating point real number</td>
</tr>
<tr>
<td>%g</td>
<td>the shorter of %f and %e</td>
</tr>
<tr>
<td>%G</td>
<td>the shorter of %f and %E</td>
</tr>
</tbody>
</table>
### Built in String Methods and Function:-

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Methods with Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>capitalize() CAPITALIZES first letter of string</td>
</tr>
<tr>
<td>2</td>
<td>center(width, fillchar) Returns a space-padded string with the original string centered to a total of width columns.</td>
</tr>
<tr>
<td>3</td>
<td>count(str, beg= 0,end=len(string)) Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.</td>
</tr>
<tr>
<td>4</td>
<td>decode(encoding='UTF-8',errors='strict') Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.</td>
</tr>
<tr>
<td>5</td>
<td>encode(encoding='UTF-8',errors='strict') Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.</td>
</tr>
<tr>
<td>6</td>
<td>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.</td>
</tr>
<tr>
<td>7</td>
<td>expandtabs(tabsize=8) Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.</td>
</tr>
<tr>
<td>8</td>
<td>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.</td>
</tr>
<tr>
<td>9</td>
<td>index(str, beg=0, end=len(string)) Same as find(), but raises an exception if str not found.</td>
</tr>
<tr>
<td>10</td>
<td>isalnum() Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.</td>
</tr>
<tr>
<td>11</td>
<td>isalpha() Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.</td>
</tr>
<tr>
<td>12</td>
<td>isdigit() Returns true if string contains only digits and false otherwise.</td>
</tr>
<tr>
<td>13</td>
<td>islower() Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.</td>
</tr>
<tr>
<td>14</td>
<td>isnumeric() Returns true if a unicode string contains only numeric characters and false otherwise.</td>
</tr>
<tr>
<td>15</td>
<td>isspace() Returns true if string contains only whitespace characters and false otherwise.</td>
</tr>
<tr>
<td>16</td>
<td>istitle() Returns true if string is properly &quot;titlecased&quot; and false otherwise.</td>
</tr>
<tr>
<td>17</td>
<td>isupper() Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.</td>
</tr>
<tr>
<td>18</td>
<td>join(seq) Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.</td>
</tr>
<tr>
<td>19</td>
<td>len(string) Returns the length of the string</td>
</tr>
<tr>
<td>20</td>
<td>ljust(width, fillchar) Returns a space-padded string with the original string left-justified to a total of width columns.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>lower()</strong></td>
<td>Converts all uppercase letters in string to lowercase.</td>
</tr>
<tr>
<td><strong>lstrip()</strong></td>
<td>Removes all leading whitespace in string.</td>
</tr>
<tr>
<td><strong>maketrans()</strong></td>
<td>Returns a translation table to be used in translate function.</td>
</tr>
<tr>
<td><strong>max()</strong></td>
<td>Returns the max alphabetical character from the string str.</td>
</tr>
<tr>
<td><strong>min()</strong></td>
<td>Returns the min alphabetical character from the string str.</td>
</tr>
<tr>
<td><strong>replace()</strong></td>
<td>Replaces all occurrences of old in string with new or at most max occurrences if max given.</td>
</tr>
<tr>
<td><strong>rfind()</strong></td>
<td>Same as find(), but search backwards in string.</td>
</tr>
<tr>
<td><strong>rindex()</strong></td>
<td>Same as index(), but search backwards in string.</td>
</tr>
<tr>
<td><strong>rjust()</strong></td>
<td>Returns a space-padded string with the original string right-justified to a total of width columns.</td>
</tr>
<tr>
<td><strong>rstrip()</strong></td>
<td>Removes all trailing whitespace of string.</td>
</tr>
<tr>
<td><strong>split()</strong></td>
<td>Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.</td>
</tr>
<tr>
<td><strong>splitlines</strong></td>
<td>Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.</td>
</tr>
<tr>
<td><strong>startswith()</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>strip()</strong></td>
<td>Performs both lstrip() and rstrip() on string.</td>
</tr>
<tr>
<td><strong>swapcase()</strong></td>
<td>Inverts case for all letters in string.</td>
</tr>
<tr>
<td><strong>title()</strong></td>
<td>Returns “titlecased” version of string, that is, all words begin with uppercase and the rest are lowercase.</td>
</tr>
<tr>
<td><strong>translate()</strong></td>
<td>Translates string according to translation table str(256 chars), removing those in the del string.</td>
</tr>
<tr>
<td><strong>upper()</strong></td>
<td>Converts lowercase letters in string to uppercase.</td>
</tr>
<tr>
<td><strong>zfill()</strong></td>
<td>Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).</td>
</tr>
<tr>
<td><strong>isdecimal()</strong></td>
<td>Returns true if a unicode string contains only decimal characters and false otherwise.</td>
</tr>
</tbody>
</table>

**Slice Operation:**
String slice is an extracted chunk of character from the original string. In python we can obtain the slice with the help of string indices.

**Example:**
```python
msg = " Good Morning"
print(msg[0:4])
```
```
O/P: ‘Good’
Print([5:12])
```
O/P: ‘Morning’
Print([[:4]])
O/P: ‘Good’
Print([5:])
O/P: ‘Morning’
Print([[:]])
O/P: ‘Good Morning’

Ord() and chr() functions:-
In python ord() function used to get integer representation for string character, ord() function takes string argument(a single unicode character) and return its integer value. Similarly, to get string representation of any integer argument function chr() is used in python.

```python
e.g.:
ch='A'
print(ord(ch))
print(chr(97))
print(ord('A'))
O/P:- 65
a
65
```

The in and not in Operator:-
It is called as membership operator. It is used to determine if the string is a member of another string or not

e.g.:
str="Hello"
if('e' in str):
    print(" e is present in the string")
else:
    print("e is not present in the string")
O/P:-
e is present in the string

str="Hello"
if('w' not in str):
    print("w is not present in the string")
else:
    print("w is present in the string")
O/P:-
w is not present in the string

Comparing String:-
We can use relational operator in string comparison. It uses ASCII value of each character from start for comparison.
print("Atharva" != "Atharv")
O/P:- True
print("Atharva"< "Pikku")
O/P:- True
print("Atharva" = "Atharv")
O/P:- False
print("Atharva"> "Pikku")
O/P:- False

Iterating Strings:-
Strings are in sequence type and are iterable, we can iterate, strings using different ways as follows:

e.g.:
str="Python"
for i in Str:
print(i)
O/P:-
Str = “Python”
for i in range(len(str)):
print(str[i])
O/P:-6

The String Module:-
The string module contains no. of constants and functions to process the string
To use the module in python we need to import it at the beginning
We will discuss, some useful function used in module

1) The capwords function to display first letter capital:-
Syntax:-string.capwards(string)
e.g.:-
string = ' Welcome TO JournalDev '
print(string.capwords(string))
Output: Welcome To Journaldev

2) The upper function and lower case
For converting the given string in to upper case letter we have to use str.upper() and
str.lower() is for lower case letter
e.g.:-
import string
text = "Monty Python's Flying Circus"
print("original string is",text)
print("upper", string.upper(text))
print("lower", string.lower(text))
O/P:- MONTY PYTHON'S FLYING CIRCUS
monty python's flying circus

3) Translation of character to other form:-
The maketrans() return the translation table for passing translate().
Syntax:-
String.maketrans(from_ch,to_ch)
e.g.:-
from_ch= “aeo”
to_ch= “012”
new_str=str.maketrans(from_ch,to_ch)
str= “I love programming in python”
print(str)
print(str.translate(new_str))
O/P:- “I love programming in python”
l12v1 pr2gr0mming in pyth2n

PROGRAM:-
1) s="hello"
print(s)
print(s[0])
print(s[-1])
print(s[1:3])
O/P:-
hello
h
o
e
2) s="hello friends"
print(s)
x='e'
if x in s:
    print("e found")
else:
    print("not found")
O/P:-
hello friends
e found
3) s1="india"
s2="is my"
s3="country"
print(s1+s2+s3)
O/P:-
indiais mycountry
4) s1="HELLO"
print(s1)
print(len(s1))
s2=s1.lower()
print(s2)
O/P:-
HELLO
5
Hello
5) s1="HELLO"
s=s1.center(10,'*')
print(s)
O/P:-
**HELLO***
6) str='programing'
print('str=', str)
print('str[0] = ', str[0])
print('str[-1] = ', str[-1])
print('str[1:5] = ', str[1:5])
print('str[5:-2] = ', str[5:-2])
O/P:-
str= programing
str[0] = p
str[-1] = g
str[1:5] = rogr
str[5:-2] = ami
7) print("Atharva" != "Atharv")
print("Atharva"< "Pikku")
O/P:-
True
True