

Explanation:

In the above program, factorial of an inputted number is calculated.

The factorial of a number is the product of all the integers from 1 to that number.

For example, the factorial of 8 (denoted as 8!) is $1*2*3*4*5*6*7*8 = 40320$.

Factorial is not defined for negative numbers and the factorial of zero is one, *i.e.*, $0! = 1$

To calculate the factorial, input of the number is accepted from the user. A factorial variable 'fact' is initialized to 1. A while loop is used to multiply the number to find the factorial. The process continues until the value of control/loop var. *i* becomes equal to the inputted number 'num'. In the last statement, the factorial of the given number is printed.

Example 17: Program to calculate the total amount payable by the customer on purchase of any item with GST levied on it. Develop a user-friendly approach for the program using while loop.

```
*prog_gst_cl12.py - C:/Users/preeti/AppData/Local/Programs/Python/Python36-32/prog_gst_cl12.py (3.6.5)*
File Edit Format Run Options Window Help
#Program to calculate bill amount inclusive of GST
choice = 'y'
total = 0.0
while choice == 'y' or choice == 'Y':    #This loop shall executes as per user's choice till y is entered
    item_price = int(input("Enter the item price : "))
    gst = int(input("Enter the GST on the item : "))
    total = total + (item_price + (item_price*gst)/100)
    choice = input("Press y to continue else press any other key :")
print("Total amount to be paid = ",total)
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 b
it (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python36-32/prog
_gst_cl12.py
Enter the item price : 4000
Enter the GST on the item : 15
Press y to continue else press any other key :y
Enter the item price : 500
Enter the GST on the item : 5
Press y to continue else press any other key :y
Enter the item price : 300
Enter the GST on the item : 10
Press y to continue else press any other key :n
Total amount to be paid = 5455.0
>>>
```

1.10 STRINGS

In Python, a string is a sequence of characters enclosed within quotes. Python treats single quotes and double quotes as equal. An individual character in a string is accessed using a subscript (index). The subscript should always be an integer (positive/negative) and begin with 0.

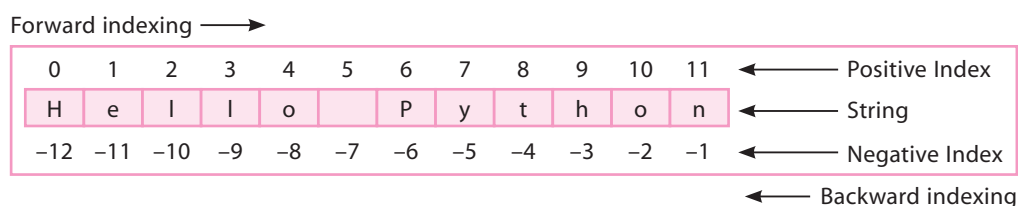


Fig. 1.5: String Representation in Python



Strings are immutable, *i.e.*, we cannot modify/change the contents of a string after its creation. In other words, we can say, Item assignment is not supported in strings.

1.10.1 String Operations

String can be manipulated using operators like concatenate (+), repetition (*), and membership operators like in and not in. Let us take a quick look at the important string operations available in Python:

Operator	Name	Description
+	Concatenation	Adds or joins two strings
*	Repetition	Concatenates multiple copies of the same string
in/not in	Membership	Returns true if a character exists in the given string
[:]	Range(start, stop, [step])	Extracts the characters from the given range
[]	Slice[n : m]	Extracts the characters from the given index

Let us understand these operations using the example given below:

Example 18: Consider two strings:

```
str1 = "Hello"
```

```
str2 = "Python"
```

Observe the result obtained after performing important string operations.

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> str1 = "Hello"
>>> str2 = "Python"
>>> print(str1 + str2)    #String concatenation
HelloPython
>>> print(str1*3)        #String Repetition
HelloHelloHello
>>> print('h' in str1)   #Membership operator - in
False
>>> print('H' in str1)
True
>>> print('H' not in str1)
False
Ln: 16 Col: 14
```

1.10.2 String Slicing

Slicing is used to retrieve a subset of values. A slice of a string is nothing but a substring. This extracted substring is termed as **slice**. A chunk of characters can be extracted from a string using slice operator with three indices in square brackets separated by colon ([:]).

The syntax is:

Syntax:

```
String_name[start:end:step]
```

Here,

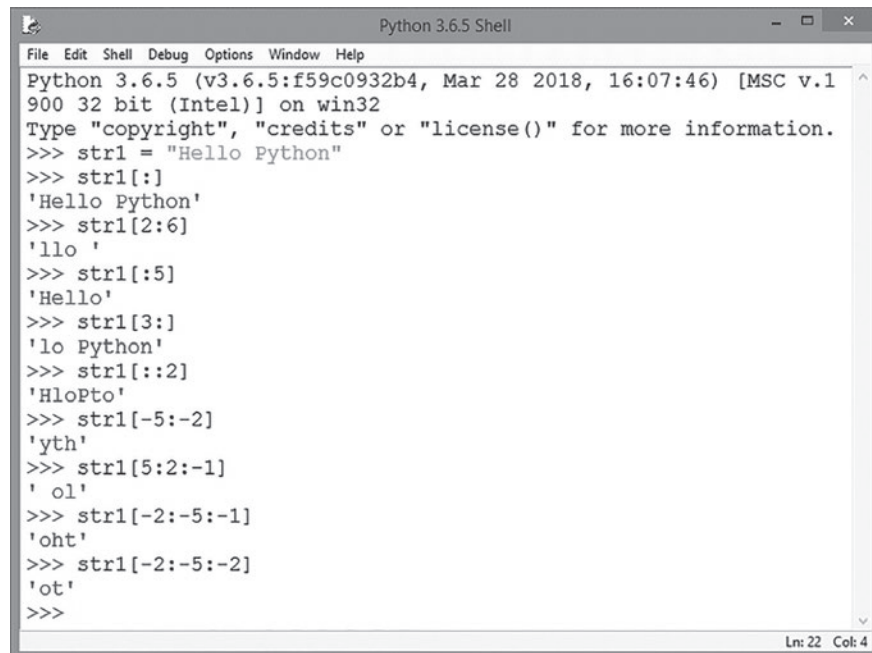
- start—starting integer where the slicing starts
- end—position till which the slicing takes place. The slicing stops at index end-1.
- step—integer value which determines the increment between each index for slicing.

start, stop values are optional. If a single parameter is passed, start and end are set to None.

Example 19: Consider a string `str1` with the following content:

```
str1 = "Hello Python"
```

The various slice operations and the output retrieved are shown below:



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1
900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> str1 = "Hello Python"
>>> str1[:]
'Hello Python'
>>> str1[2:6]
'ello '
>>> str1[:5]
'Hello'
>>> str1[3:]
'lo Python'
>>> str1[::2]
'HloPto'
>>> str1[-5:-2]
'yth'
>>> str1[5:2:-1]
' ol'
>>> str1[-2:-5:-1]
'oht'
>>> str1[-2:-5:-2]
'ot'
>>>
```

1.10.3 Built-in String Methods

Python provides the following built-in methods to manipulate strings:

Method	Description	Example
isalpha()	Returns True if the string contains only letters, otherwise returns False. Syntax: <code>str.isalpha()</code>	<pre>>>> str = "Good" >>> print(str.isalpha()) True #Returns True as no special character or digit is present in the string. >>> str1="This is a string" >>> print(str1.isalpha()) False #Returns False as the string contains spaces. >>> str1="Working with...Python!!" >>> print(str1.isalpha()) False #Returns False as the string contains special characters and spaces.</pre>
isdigit()	This method returns True if string contains only digits, otherwise False. Syntax: <code>str.isdigit()</code>	<pre>>>> str1="123456" >>> print(str1.isdigit()) True #Returns True as the string contains only digits. >>> str1 = "Ram bagged 1st position" >>> print(str1.isdigit()) False #Returns False because apart from digits, the string contains letters and spaces.</pre>



lower()	Converts all the uppercase letters in the string to lowercase. Syntax: <code>str.lower()</code>	<pre>>>> str1= "Learning PYTHON" >>> print(str1.lower()) learning python #Converts uppercase letters only to lowercase. >>> str1= "learning python" >>> print(str1.lower()) learning python #if already in lowercase, then it will simply return the string.</pre>
islower()	Returns True if all letters in the string are in lowercase. Syntax: <code>str.islower()</code>	<pre>>>> str1="python" >>> print(str1.islower()) True >>> str1 = "Python" >>> print(str1.islower()) False</pre>
upper()	Converts lowercase letters in the string to uppercase. Syntax: <code>str.upper()</code>	<pre>>>> var1= "Welcome" >>> print(var1.upper()) WELCOME >>> var1= "WELCOME" >>> print(var1.upper()) WELCOME #if already in uppercase, then it will simply return the string.</pre>
isupper()	Returns True if the string is in uppercase. Syntax: <code>str.isupper()</code>	<pre>>>> str1= "PYTHON" >>> print(str1.isupper()) True >>> str1= "PythOn" >>> print(str1.isupper()) False</pre>
lstrip() or lstrip(chars)	Returns the string after removing the space(s) from the left of the string. Syntax: <code>str.lstrip()</code> or <code>str.lstrip(chars)</code> <code>chars</code> (optional) – a string specifying the set of characters to be removed from the left. All combinations of characters in the <code>chars</code> argument are removed from the left of the string until the left character of the string mismatches.	<pre>>>> str1= " Green Revolution" >>> print(str1.lstrip()) Green Revolution #Here no argument was given, hence it removed all leading whitespaces from the left of the string. >>> str2= "Green Revolution" >>> print(str2.lstrip("Gr")) een Revolution >>> str2= "Green Revolution" >>> print(str2.lstrip("rG")) een Revolution #Here all elements of the given argument are matched with the left of the str2 and, if found, are removed.</pre>

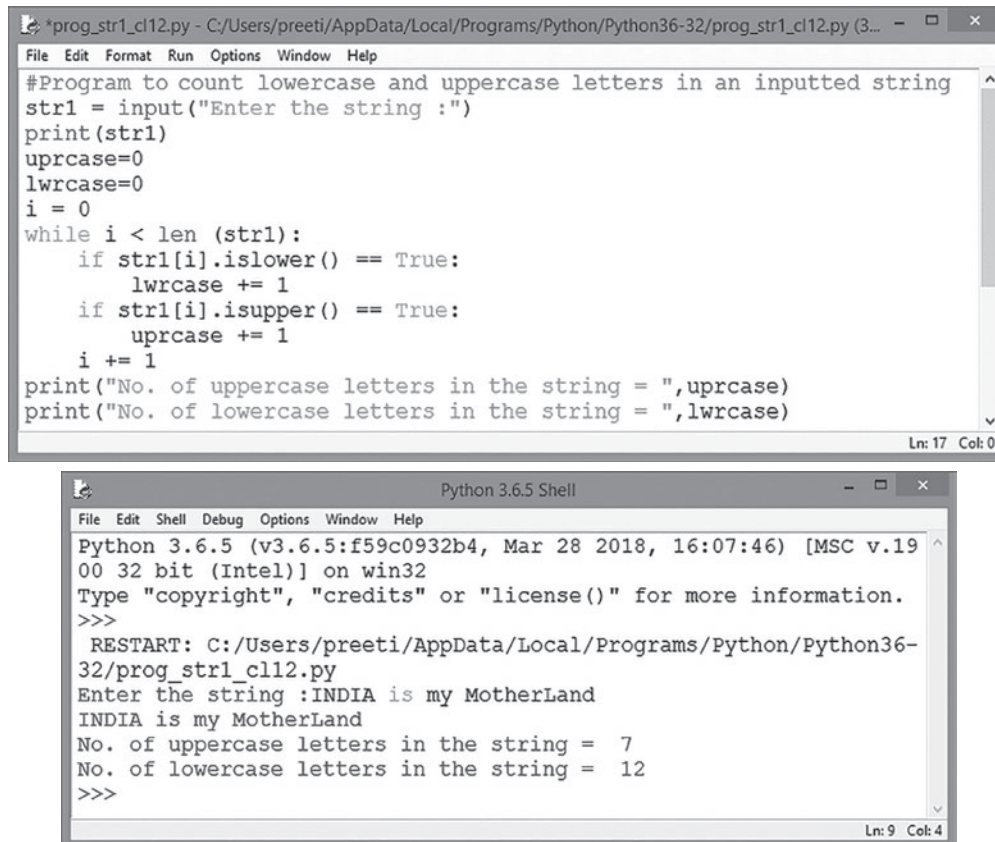


rstrip() or rstrip(chars)	<p>This method removes all the trailing whitespaces from the right of the string.</p> <p>Syntax: rstrip() or str.rstrip(chars) chars (optional) – a string specifying the set of characters to be removed from the right. All combinations of characters in the <i>chars</i> argument are removed from the right of the string until the right character of the string mismatches.</p>	<pre>>>> str1= "Green Revolution" >>> print(str1.rstrip()) Green Revolution #Here no argument was given, hence it removed all leading whitespaces from the right of the string. >>> str1= "Computers" >>> print(str1.rstrip("rs")) Compute</pre> <p>#Here the letters 'rs' are passed as an argument; it is matched from the right of the string and removed from the right of str1.</p>
isspace()	<p>Returns True if string contains only whitespace characters, otherwise returns False.</p> <p>Syntax: str.isspace()</p>	<pre>>>> str1= " " >>> print(str1.isspace()) True >>> str1=" Python " >>> print(str1.isspace()) False</pre>
istitle()	<p>The istitle() method doesn't take any arguments. It returns True if string is properly "titlecased", else returns False if the string is not a "titlecased" string or an empty string.</p> <p>Syntax: str.istitle()</p>	<pre>>>> str1= "All Learn Python" >>> print(str1.istitle()) True >>> s= "All learn Python" >>> print(s.istitle()) False >>> s= "This Is @ Symbol" >>> print(s.istitle()) True >>> s= "PYTHON" >>> print(s.istitle()) False</pre>
join(sequence)	<p>Returns a string in which the string elements have been joined by a string separator.</p> <p>Syntax: str.join(sequence) sequence – Join() takes an argument which is of sequence data type capable of returning its elements one at a time. This method returns a string, which is the concatenation of each element of the string and the string separator between each element of the string.</p>	<pre>>>> str1= "12345" >>> s= "-" >>> s.join(str1) '1-2-3-4-5' >>> str2= "abcd" >>> s= "#" >>> s.join(str2) 'a#b#c#d'</pre>



swapcase()	<p>This method converts and returns all uppercase characters to lowercase and vice versa of the given string. It does not take any argument.</p> <p>Syntax: <code>str.swapcase()</code></p> <p>The <code>swapcase()</code> method returns a string with all the cases changed.</p>	<pre>>>> str1= "Welcome" >>> str1.swapcase() 'wELCOME' >>> str2= "PYTHON" >>> str2.swapcase() 'python' >>> s= "pYThoN" >>> s.swapcase() 'PyTHOn'</pre>
partition (Separator)	<p>Partition method is used to split the given string using the specified separator and return a tuple with three parts: Substring before the separator; separator itself; a substring after the separator.</p> <p>Syntax: <code>str. partition(Separator)</code></p> <p>Separator: This argument is required to separate a string. If the separator is not found, it returns the string itself, followed by two empty strings within the parentheses, as tuple.</p>	<pre>>>> str3= "xyz@gmail.com" >>> str3.partition(' ') ('xyz@gmail.com', ' ', '') #Here separator is not found, returns the string itself, followed by two empty strings. >>> str2= "Hardworkpays" >>> str2.partition('work') ('Hard', 'work', 'pays') #Here str2 is separated into three parts— 1) the substring before the separator, i.e., 'Hard' 2) the separator itself, i.e., 'work', and 3) the substring part after the separator, i.e., 'pays' >>> str5= str3.partition('@') >>> print(str5) ('xyz', '@', 'gmail.com') >>> str4= str2.partition('-') >>> print(str4) ('Hardworkpays', '', '')</pre>
<p>In internal storage or the memory of computer, the characters are stored in integer value. A specific value is used for a given character and it is based on ASCII code. There are different numbers assigned to capital letters and small letters.</p> <p>Python provides two functions for character encoding: <code>ord()</code> and <code>chr()</code>.</p>		
ord()	<p>ord() – function returns the ASCII code of the character.</p>	<pre>>>> ch= 'b' >>> ord(ch) 98 >>> ord('A') 65</pre>
chr()	<p>chr() – function returns character represented by the inputted ASCII number.</p>	<pre>>>> chr(97) 'a' >>> chr(66) 'B'</pre>

Example 20: Program to input a string and count the number of uppercase and lowercase letters.



```
*prog_str1_cl12.py - C:/Users/preeti/AppData/Local/Programs/Python/Python36-32/prog_str1_cl12.py (3...
File Edit Format Run Options Window Help
#Program to count lowercase and uppercase letters in an inputted string
str1 = input("Enter the string :")
print(str1)
uprcase=0
lwrcase=0
i = 0
while i < len (str1):
    if str1[i].islower() == True:
        lwrcase += 1
    if str1[i].isupper() == True:
        uprcase += 1
    i += 1
print("No. of uppercase letters in the string = ",uprcase)
print("No. of lowercase letters in the string = ",lwrcase)
Ln: 17 Col: 0

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.19
00 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python36-
32/prog_str1_cl12.py
Enter the string :INDIA is my MotherLand
INDIA is my MotherLand
No. of uppercase letters in the string = 7
No. of lowercase letters in the string = 12
>>>
Ln: 9 Col: 4
```

Explanation:

The above program counts the total number of lowercase and uppercase letters in an inputted string. The string can contain a mix of characters, numbers or any other special characters.

The inputted string is stored in the variable 'str 1'. Two variables are initialized to 0 for storing the number of uppercase and lowercase letters respectively. The loop shall iterate till the end of the string which is taken into account for calculating its length. For checking if a character is lowercase or uppercase, we have used two inbuilt methods, islower() & isupper() of the string library. If the character is in lowercase, the counter var 'lwrcase' shall be incremented and for uppercase, var 'uprcase' shall be incremented and finally the count shall be displayed using appropriate print() statements.

1.11 LISTS

Like strings, lists are a sequence of values. A list is a data type that can be used to store any type and number of variables and information. The values in the list are called elements or items or list members.

A list in Python is formed by enclosing the values inside square brackets ([]). **Unlike strings, lists are mutable**, i.e., values in a list can be changed or modified and can be accessed using index value enclosed in square brackets.

Syntax:

```
[list_name] = [item1,item2,item3...,itemn]
```

For example, L1 = [10, 20, 30, 40]

Consider list1 containing 10 elements.

```
list1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```