# **Python | Output Formatting**

There are several ways to present the output of a program. Data can be printed in a human-readable form, or written to a file for future use, or even in some other specified form. Users often want more control over the formatting of output than simply printing space-separated values. There are several ways to format output.

* To use [formatted string literals](https://www.geeksforgeeks.org/formatted-string-literals-f-strings-python/), begin a string with f or F before the opening quotation mark or triple quotation mark.
* The [str.format()](https://www.geeksforgeeks.org/python-format-function/) method of strings helps a user create a fancier output
* Users can do all the string handling by using string slicing and concatenation operations to create any layout that the users want. The string type has some methods that perform useful operations for padding strings to a given column width.

**Formatting output using String modulo operator(%) :**   
The % operator can also be used for string formatting. It interprets the left argument much like a printf()-style format as in C language strings to be applied to the right argument. In Python, there is no printf() function but the functionality of the ancient printf is contained in Python. To this purpose, the modulo operator % is overloaded by the string class to perform string formatting. Therefore, it is often called a string modulo (or sometimes even called modulus) operator.

The string modulo operator ( % ) is still available in Python(3.x) and is widely used. But nowadays the old style of formatting is removed from the language.

|  |
| --- |
| # Python program showing how to use  # string modulo operator(%) to print  # fancier output    # print integer and float value  print("Geeks : %2d, Portal : %5.2f"%(1, 05.333))    # print integer value  print("Total students : %3d, Boys : %2d"%(240, 120))    # print octal value  print("%7.3o"%(25))    # print exponential value  print("%10.3E"%(356.08977)) |

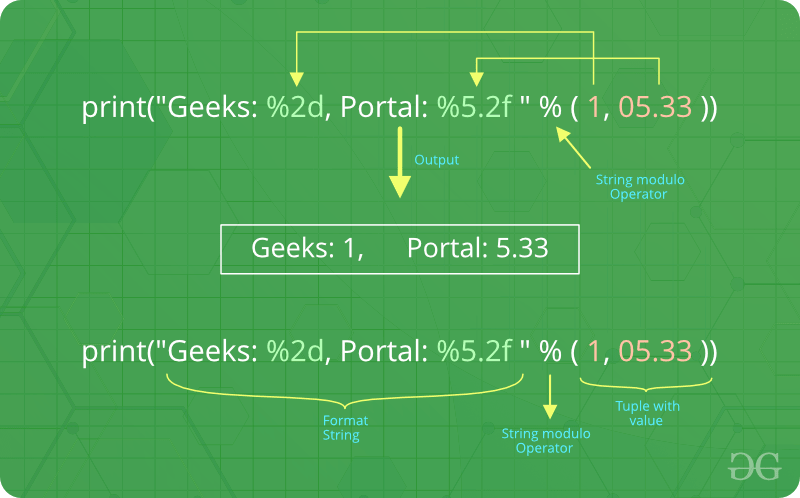
**Output :**

Geeks : 1, Portal : 5.33

Total students : 240, Boys : 120

031

3.561E+02



There are two of those in our example: “%2d” and “%5.2f”. The general syntax for a format placeholder is:

%[flags][width][.precision]type

Let’s take a look at the placeholders in our example.

* The first placeholder “%2d” is used for the first component of our tuple, i.e. the integer 1. The number will be printed with 2 characters. As 1 consists only of one digit, the output is padded with 1 leading blanks.
* The second one “%5.2f” is a format description for a float number. Like other placeholders, it is introduced with the % character. This is followed by the total number of digits the string should contain. This number includes the decimal point and all the digits, i.e. before and after the decimal point.
* Our float number 05.333 has to be formatted with 5 characters. The decimal part of the number or the precision is set to 2, i.e. the number following the “.” in our placeholder. Finally, the last character “f” of our placeholder stands for “float”.

**Formatting output using the format method :**   
The format() method was added in Python(2.6). The format method of strings requires more manual effort. Users use {} to mark where a variable will be substituted and can provide detailed formatting directives, but the user also needs to provide the information to be formatted. This method lets us concatenate elements within an output through positional formatting. For Example –

**Code 1:**

|  |
| --- |
| # Python program showing  # use of format() method    # using format() method  print('I love {} for "{}!"'.format('Geeks', 'Geeks'))    # using format() method and referring  # a position of the object  print('{0} and {1}'.format('Geeks', 'Portal'))    print('{1} and {0}'.format('Geeks', 'Portal'))      # the above formatting can also be done by using f-Strings  # Although, this features work only with python 3.6 or above.    print(f"I love {'Geeks'} for \"{'Geeks'}!\"")    # using format() method and referring  # a position of the object  print(f"{'Geeks'} and {'Portal'}") |

**Output :**

I love Geeks for "Geeks!"

Geeks and Portal

Portal and Geeks

The brackets and characters within them (called **format fields**) are replaced with the objects passed into the format() method. A number in the brackets can be used to refer to the position of the object passed into the format() method.   
    
**Code 2:**

|  |
| --- |
| # Python program showing  # a use of format() method    # combining positional and keyword arguments  print('Number one portal is {0}, {1}, and {other}.'       .format('Geeks', 'For', other ='Geeks'))    # using format() method with number  print("Geeks :{0:2d}, Portal :{1:8.2f}".        format(12, 00.546))    # Changing positional argument  print("Second argument: {1:3d}, first one: {0:7.2f}".        format(47.42, 11))    print("Geeks: {a:5d},  Portal: {p:8.2f}".       format(a =453, p =59.058)) |

**Output:**

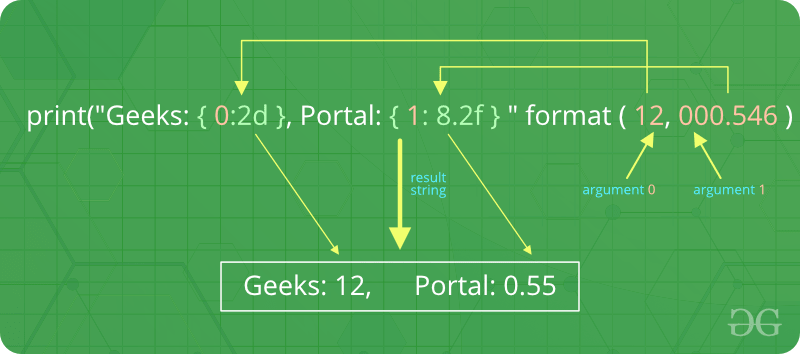
Number one portal is Geeks, For, and Geeks.

Geeks :12, Portal : 0.55

Second argument: 11, first one: 47.42

Geeks: 453, Portal: 59.06

The following diagram with an example usage depicts how the format method works for positional parameters:



**Code 3:**

|  |
| --- |
| # Python program to  # show format () is  # used in dictionary    tab ={'geeks': 4127, 'for': 4098, 'geek': 8637678}    # using format() in dictionary  print('Geeks: {0[geeks]:d}; For: {0[for]:d}; '      'Geeks: {0[geek]:d}'.format(tab))    data =dict(fun ="GeeksForGeeks", adj ="Portal")    # using format() in dictionary  print("I love {fun} computer {adj}".format(\*\*data)) |

**Output:**

Geeks: 4127; For: 4098; Geeks: 8637678

I love GeeksForGeeks computer Portal

**Formatting output using the String method :**   
This output is formatted by using string slicing and concatenation operations. The string type has some methods that help in formatting output in a fancier way. Some methods which help in formatting an output are [str.](https://www.geeksforgeeks.org/python-string-ljust-rjust-center/)rjust[()](https://www.geeksforgeeks.org/python-string-ljust-rjust-center/), [str.rjust()](https://www.geeksforgeeks.org/python-string-ljust-rjust-center/), and [str.centre()](https://www.geeksforgeeks.org/python-string-ljust-rjust-center/)

|  |
| --- |
| # Python program to  # format a output using  # string() method    cstr ="I love geeksforgeeks"    # Printing the center aligned  # string with fillchr  print("Center aligned string with fillchr: ")  print(cstr.center(40, '#'))    # Printing the left aligned  # string with "-" padding  print("The left aligned string is : ")  print(cstr.ljust(40, '-'))    # Printing the right aligned string  # with "-" padding  print("The right aligned string is : ")  print(cstr.rjust(40, '-')) |

**Output:**

Center aligned string with fillchr:

##########I love geeksforgeeks##########

The left aligned string is :

I love geeksforgeeks--------------------

The right aligned string is :

--------------------I love geeksforgeeks