Python - Built-in Functions

Built-in Functions in Python?

Built-in functions are those functions that are pre-defined in the Python interpreter and you don't need to import any module to use them. These functions help to perform a wide variety of operations on strings, iterators, and numbers. For instance, the built-in functions like sum(), min(), and max() are used to simplify mathematical operations.

How to Use Built-in Function in Python?

To use **built-in functions** in your code, simply call the specific function by passing the required parameter (if any) inside the parentheses. Since these functions are pre-defined, you don't need to import any module or package.

Example of Using Built-in Functions

Consider the following example demonstrating the use of built-in functions in your code:

```
# Using print() and len() function

text = "Tutorials Point"

print(len(text)) # Prints 15
```

In the above example, we are using two built-in functions print() and len().

Learn **Python** in-depth with real-world projects through our **Python certification course**. Enroll and become a certified expert to boost your career.

List of Python Built-in Functions

As of Python 3.12.2 version, the list of built-in functions is given below –

Sr.No.	Function & Description
1	**Python aiter() function** Returns an asynchronous iterator for an asynchronous iterable.
2	**Python all() function** Returns true when all elements in iterable is true.
3	**Python anext() function**

	Returns the next item from the given asynchronous iterator.
4	**Python any() function** Checks if any Element of an Iterable is True.
5	**Python ascii() function** Returns String Containing Printable Representation.
6	**Python bin() function** Converts integer to binary string.
7	**Python bool() function** Converts a Value to Boolean.
8	**Python breakpoint() function** This function drops you into the debugger at the call site and calls sys.breakpointhook().
9	**Python bytearray() function** Returns array of given byte size.
10	**Python bytes() function** Returns immutable bytes object.
11	**Python callable() function** Checks if the Object is Callable.
12	**Python chr() function** Returns a Character (a string) from an Integer.
13	**Python classmethod() function** Returns class method for given function.
14	**Python compile() function** Returns a code object.
15	**Python complex() function** Creates a Complex Number.
16	**Python delattr() function** Deletes Attribute From the Object.
17	**Python dict() function** Creates a Dictionary.
18	**Python dir() function** Tries to Return Attributes of Object.
19	**Python divmod() function**

	Returns a Tuple of Quotient and Remainder.
20	**Python enumerate() function** Returns an Enumerate Object.
21	**Python eval() function** Runs Code Within Program.
22	**Python exec() function** Executes Dynamically Created Program.
23	**Python filter() function** Constructs iterator from elements which are true.
24	**Python float() function** Returns floating point number from number, string.
25	**Python format() function** Returns formatted representation of a value.
26	**Python frozenset() function** Returns immutable frozenset object.
27	**Python getattr() function** Returns value of named attribute of an object.
28	**Python globals() function** Returns dictionary of current global symbol table.
29	**Python hasattr() function** Returns whether object has named attribute.
30	**Python hash() function** Returns hash value of an object.
31	**Python help() function** Invokes the built-in Help System.
32	**Python hex() function** Converts to Integer to Hexadecimal.
33	**Python id() function** Returns Identify of an Object.
34	**Python input() function** Reads and returns a line of string.
35	**Python int() function** Returns integer from a number or string.

36	**Python isinstance() function** Checks if a Object is an Instance of Class.
37	**Python issubclass() function** Checks if a Class is Subclass of another Class.
38	**Python iter() function** Returns an iterator.
39	**Python len() function** Returns Length of an Object.
40	**Python list() function** Creates a list in Python.
41	**Python locals() function** Returns dictionary of a current local symbol table.
42	**Python map() function** Applies Function and Returns a List.
43	**Python memoryview() function** Returns memory view of an argument.
44	**Python next() function** Retrieves next item from the iterator.
45	**Python object() function** Creates a featureless object.
46	**Python oct() function** Returns the octal representation of an integer.
47	**Python open() function** Returns a file object.
48	**Python ord() function** Returns an integer of the Unicode character.
49	**Python print() function** Prints the Given Object.
50	**Python property() function** Returns the property attribute.
51	**Python range() function** Returns a sequence of integers.

52	**Python repr() function** Returns a printable representation of the object.
53	**Python reversed() function** Returns the reversed iterator of a sequence.
54	**Python set() function** Constructs and returns a set.
55	**Python setattr() function** Sets the value of an attribute of an object.
56	**Python slice() function** Returns a slice object.
57	**Python sorted() function** Returns a sorted list from the given iterable.
58	**Python staticmethod() function** Transforms a method into a static method.
59	**Python str() function** Returns the string version of the object.
60	**Python super() function** Returns a proxy object of the base class.
61	**Python tuple() function** Returns a tuple.
62	**Python type() function** Returns the type of the object.
63	**Python vars() function** Returns the __dict__ attribute.
64	**Python zip() function** Returns an iterator of tuples.
65	**Python __import__() function** Function called by the import statement.
66	**Python unichr() function** Converts a Unicode code point to its corresponding Unicode character.
67	**Python long() function** Represents integers of arbitrary size.

Built-in Mathematical Functions

There are some additional built-in functions that are used for performing only mathematical operations in Python, they are listed below –

Sr.No.	Function & Description
1	**Python abs() function** The abs() function returns the absolute value of x, i.e. the positive distance between x and zero.
2	**Python max() function** The max() function returns the largest of its arguments or largest number from the iterable (list or tuple).
3	**Python min() function** The function min() returns the smallest of its arguments i.e. the value closest to negative infinity, or smallest number from the iterable (list or tuple)
4	**Python pow() function** The pow() function returns x raised to y. It is equivalent to x**y. The function has third optional argument mod. If given, it returns (x**y) % mod value
5	**Python round() Function** round() is a built-in function in Python. It returns x rounded to n digits from the decimal point.
6	**Python sum() function** The sum() function returns the sum of all numeric items in any iterable (list or tuple). An optional start argument is 0 by default. If given, the numbers in the list are added to start value.

Advantages of Using Built-in Functions

The following are the advantages of using built-in functions:

- The use of the built-in functions simplifies and reduces the code length and enhances the readability of the code.
- Instead of writing the same logic repeatedly, you can use these functions across different sections of the program. This not only saves time but also helps in maintaining consistency of code.
- These functions provide a wide range of functionalities including mathematical operations, datatype conversion, and performing operations on iterators.
- These functions have descriptive names that make the code easier to understand and maintain. Developers need not write additional complex code for performing

certain operations.

Frequently Asked Questions about Built-in Functions

How do I handle errors with built-in functions?

While working with built-in functions, you may encounter errors and to handle those errors you can use the try-except blocks. This may help you identify the type of error and exceptions raised.

Can we extend the functionality of built-in functions?

Yes, we can extend the functionality of built-in functions by using it with other methods and by applying your logic as per the need. However, it will not affect the pre-defined feature of the used function.

Can I create my built-in functions?

No, you cannot create your built-in function. But, Python allows a user to create user-defined functions.

How do I use built-in functions?

Using a built-in function is very simple, call it by its name followed by parentheses, and pass the required arguments inside the parentheses.