Here is a list of all the built-in functions in Python 3.11.4 along with a brief explanation of each function according to the official Python documentation[1]:

- `abs()` returns the absolute value of a number.
- `aiter()` returns an asynchronous iterator for an asynchronous iterable.
- `all()` returns True if all elements of the iterable are true (or if the iterable is empty).
- `any()` returns True if any element of the iterable is true.
- `anext()` when awaited, return the next item from the given asynchronous iterator, or default if given and the iterator is exhausted.
- `ascii()` returns a readable version of an object. Replaces none-ascii characters with escape character.
- `bin()` returns the binary version of a number.
- `bool()` returns the boolean value of the specified object.
- `breakpoint()` calls sys.breakpointhook(). By default, this drops you into the pdb debugger.
- `bytearray()` returns an array of bytes.
- `bytes()` returns a bytes object.
- `callable()` returns True if the specified object is callable, otherwise False.
- `chr()` returns a character from the specified Unicode code.
- `classmethod()` converts a method into a class method.
- `compile()` returns the specified source as an object, ready to be executed.
- `complex()` returns a complex number.
- `delattr()` deletes the specified attribute (property or method) from the specified object.
- `dict()` returns a dictionary (Array).
- `dir()` returns a list of the specified object's properties and methods.
- `divmod()` returns the quotient and the remainder when argument1 is divided by argument2.

- `enumerate()` takes a collection (e.g. a tuple) and returns it as an enumerate object.
- `eval()` evaluates and executes an expression.
- `exec()` executes the specified code (or object).
- `filter()` use a filter function to exclude items in an iterable object.
- `float()` returns a floating point number.
- `format()` formats a specified value.
- `frozenset()` returns a frozenset object.
- `getattr()` returns the value of the specified attribute (property or method).
- `globals()` returns the current global symbol table as a dictionary.
- `hasattr()` returns True if the specified object has the specified attribute (property/method).
- `hash()` returns the hash value of a specified object.
- `help()` executes the built-in help system.
- `hex()` converts a number into a hexadecimal value.
- `id()` returns the id of an object.
- `input()` allowing user input.
- `int()`: Returns an integer number
- `isinstance()`: Returns True if a specified object is an instance of a specified object
- `issubclass()`: Returns True if a specified class is a subclass of a specified object
- `iter()`: Returns an iterator object
- `len()`: Returns the length of an object
- `list()`: Returns a list
- `locals()`: Returns an updated dictionary of the current local symbol table
- `map()`: Returns the specified iterator with the specified function applied to each item
- `max()`: Returns the largest item in an iterable

- `memoryview()`: Returns a memory view object
- `min()`: Returns the smallest item in an iterable
- `next()`: Returns the next item in an iterable
- `object()`: Returns a new object
- `oct()`: Converts a number into an octal
- `open()`: Opens a file and returns a file object
- `ord()`: Convert an integer representing the Unicode of the specified character
- `pow()`: Returns the value of x to the power of y
- `print()`: Prints to the standard output device
- `property()`: Gets, sets, deletes a property
- `range()`: Returns a sequence of numbers, starting from 0 and increments by 1 (by default)
- `repr()`: Returns a readable version of an object
- `reversed()`: Returns a reversed iterator of a sequence
- `round()`: Rounds off to nearest integer or float with n decimal places
- `set()`: Creates and return set objects
- `setattr()`: Sets value for given attribute
- `slice()`: Creates and return slice objects
- `sorted()`: Sorts given sequence
- `staticmethod()`: Converts method into static method
- `str()`: Creates and return string objects
- `sum()`: Sums up all elements in given sequence
- `super()`: Calls parent class method
- `tuple()`: Creates and return tuple objects
- `type()`: Return type of given objects

- vars(): Return **dict** attribute for given module/class/instance
- zip(): Return iterator for tuples where first element is from first sequence, second element from second sequence and so on **import**(): Invoked by import statement