The `operator` module also defines tools for generalized attribute and item lookups. These are useful for making fast field extractors as arguments for `map()`, `sorted()`, `itertools.groupby()`, or other functions that expect a function argument.

`operator.attrgetter`(*attr*[, *args*...])

Return a callable object that fetches *attr* from its operand. If more than one attribute is requested, returns a tuple of attributes. After, `f = attrgetter('name')`, the call `f(b)` returns `b.name`. After, `f = attrgetter('name', 'date')`, the call `f(b)` returns `(b.name, b.date)`.

The attribute names can also contain dots; after `f = attrgetter('date.month')`, the call `f(b)` returns `b.date.month`.

New in version 2.4.

Changed in version 2.5: Added support for multiple attributes.

Changed in version 2.6: Added support for dotted attributes.

`operator.itemgetter`(*item*[, *args*...])

Return a callable object that fetches *item* from its operand using the operand's `__getitem__()` method. If multiple items are specified, returns a tuple of lookup values. Equivalent to:

```
def itemgetter(*items):
    if len(items) == 1:
        item = items[0]
        def g(obj):
            return obj[item]
    else:
        def g(obj):
            return tuple(obj[item] for item in items)
    return g
```

The items can be any type accepted by the operand's `__getitem__()` method. Dictionaries accept any hashable value. Lists, tuples, and strings accept an index or a slice:

```
>>> itemgetter(1)('ABCDEFG')
'B'
>>> itemgetter(1,3,5)('ABCDEFG')
('B', 'D', 'F')
>>> itemgetter(slice(2,None))('ABCDEFG')
'CDEFG'
```

New in version 2.4.

Changed in version 2.5: Added support for multiple item extraction.

Example of using `itemgetter()` to retrieve specific fields from a tuple record:

```
>>> inventory = [('apple', 3), ('banana', 2), ('pear', 5), ('orange', 1)]
>>> getcount = itemgetter(1)
>>> map(getcount, inventory)
[3, 2, 5, 1]
>>> sorted(inventory, key=getcount)
[('orange', 1), ('banana', 2), ('apple', 3), ('pear', 5)]
```