

```
# This file is a backport of the PEP 372 OrderedDict class to be added  
# to Python 2.7 and 3.1.  
#
```

Classes

```
\_\_builtin\_\_.dict(\_\_builtin\_\_.object)  
OrderedDict(\_\_builtin\_\_.dict, \_abcoll.MutableMapping)  
\_abcoll.MutableMapping(\_abcoll.Mapping)  
OrderedDict(\_\_builtin\_\_.dict, \_abcoll.MutableMapping)
```

```
class OrderedDict(\_\_builtin\_\_.dict, \_abcoll.MutableMapping)
```

Method resolution order:

```
OrderedDict  
\_\_builtin\_\_.dict  
\_abcoll.MutableMapping  
\_abcoll.Mapping  
\_abcoll.Sized  
\_abcoll.Iterable  
\_abcoll.Container  
\_\_builtin\_\_.object
```

Assignment Project Exam Help

<https://powcoder.com>

Methods defined here:

Add WeChat powcoder

```
__delitem__(self, key)  
__eq__(self, other)  
__init__(self, *args, **kwargs)  
__iter__(self)  
__reduce__(self)  
__reversed__(self)  
__setitem__(self, key, value)  
clear(self)  
copy(self)  
dequeueitem(self)  
items(self)
```

keys(self)

pop(self, key, default=<object object>)

popitem(self)

setdefault(self, key, default=None)

update(self, other=(), **kwds)

values(self)

Class methods defined here:

fromkeys(cls, iterable, value=None) from [abc.ABCMeta](#)

Data descriptors defined here:

__dict__
dictionary for instance variables (if defined)

__weakref__
list of weak references to the object (if defined)

<https://powcoder.com>

Data and other attributes defined here:

__abstractmethods__ = frozenset([])

Methods inherited from [__builtin__.dict](#):

__cmp__(...)
x.[__cmp__](#)(y) <==> cmp(x,y)

__contains__(...)
D.[__contains__](#)(k) -> True if D has a key k, else False

__ge__(...)
x.[__ge__](#)(y) <==> x>=y

__getattr__(...)
x.[__getattr__](#)('name') <==> x.name

__getitem__(...)
x.[__getitem__](#)(y) <==> x[y]

__gt__(...)
x.[__gt__](#)(y) <==> x>y

__le__(...)

Assignment Project Exam Help

Add WeChat powcoder

`x. le(y) <==> x<=y`

`len(...)
x. len() <==> len(x)`

`lt(...)
x. lt(y) <==> x<y`

`ne(...)
x. ne(y) <==> x!=y`

`repr(...)
x. repr() <==> repr(x)`

`sizeof(...)
D. sizeof() -> size of D in memory, in bytes`

`get(...)
D. get(k[,d]) -> D[k] if k in D, else d. d defaults to None`

`has_key(...)
D. has_key(k) -> True if D has a key k, else False`

`iteritems(...)
D. iteritems() -> an iterator over the (key, value) items of D`

`iterkeys(...)
D. iterkeys() -> an iterator over the keys of D`

`itervalues(...)
D. itervalues() -> an iterator over the values of D`

Data and other attributes inherited from [builtin .dict](#):

`__hash__ = None`

`__new__ = <built-in method __new__ of type object>
T. new(S, ...) -> a new object with type S, a subtype of T`

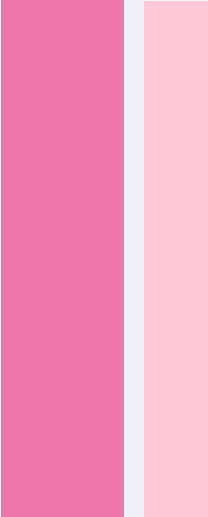
Class methods inherited from [abcoll.Sized](#):

`__subclasshook__(cls, C) from abc.ABCMeta`

Data and other attributes inherited from [abcoll.Sized](#):

`__metaclass__ = <class 'abc.ABCMeta'>
Metaclass for defining Abstract Base Classes (ABCs).`

Use this metaclass to create an ABC. An ABC can be subclassed directly, and then acts as a mix-in class. You can also re



```
ster
unrelated concrete classes (even built-in classes) and unre
ted
ABCs as 'virtual subclasses' -- these and their descendants
ill
be considered subclasses of the registering ABC by the buil
in
issubclass() function, but the registering ABC won't show u
in
their MRO (Method Resolution Order) nor will method
implementations defined by the registering ABC be callable
ot
even via super()).
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder