help

Type help() for interactive help, or help(object) for help about object.

help(set)

Help on class set in module builtins:

class set(object)

* set() -> new empty set object

set(iterable) -> new set object

* Build an unordered collection of unique elements.
* Methods defined here:
* \_\_and\_\_(self, value, /)
* Return self&value.

* \_\_contains\_\_(...)
* x.\_\_contains\_\_(y) <==> y in x.

* \_\_eq\_\_(self, value, /)
* Return self==value.

* \_\_ge\_\_(self, value, /)
* Return self>=value.
* \_\_getattribute\_\_(self, name, /)
* Return getattr(self, name).
* \_\_gt\_\_(self, value, /)
* Return self>value.

* \_\_iand\_\_(self, value, /)
* Return self&=value.

* \_\_init\_\_(self, /, \*args, \*\*kwargs)
* Initialize self. See help(type(self)) for accurate signature.

* \_\_ior\_\_(self, value, /)
* Return self|=value.

* \_\_isub\_\_(self, value, /)
* Return self-=value.
* \_\_iter\_\_(self, /)
* Implement iter(self).
* \_\_ixor\_\_(self, value, /)
* Return self^=value.
* \_\_le\_\_(self, value, /)
* Return self<=value.
* \_\_len\_\_(self, /)
* Return len(self).
* \_\_lt\_\_(self, value, /)
* Return self<value.
* \_\_ne\_\_(self, value, /)
* Return self!=value.
* \_\_or\_\_(self, value, /)
* Return self|value.
* \_\_rand\_\_(self, value, /)
* Return value&self.
* \_\_reduce\_\_(...)
* Return state information for pickling.
* \_\_repr\_\_(self, /)
* Return repr(self).
* \_\_ror\_\_(self, value, /)
* Return value|self.
* \_\_rsub\_\_(self, value, /)
* Return value-self.
* \_\_rxor\_\_(self, value, /)
* Return value^self.
* \_\_sizeof\_\_(...)
* S.\_\_sizeof\_\_() -> size of S in memory, in bytes
* \_\_sub\_\_(self, value, /)
* Return self-value.
* \_\_xor\_\_(self, value, /)

Return self^value.

* add(...)
* Add an element to a set.
* This has no effect if the element is already present.
* clear(...)
* Remove all elements from this set.
* copy(...)
* Return a shallow copy of a set.
* difference(...)
* Return the difference of two or more sets as a new set.
* (i.e. all elements that are in this set but not the others.)

difference\_update(...)

* Remove all elements of another set from this set.
* discard(...)
* Remove an element from a set if it is a member.
* If the element is not a member, do nothing.
* intersection(...)
* Return the intersection of two sets as a new set.
* (i.e. all elements that are in both sets.)
* intersection\_update(...)
* Update a set with the intersection of itself and another.
* isdisjoint(...)
* Return True if two sets have a null intersection.
* issubset(...)
* Report whether another set contains this set.
* issuperset(...)
* Report whether this set contains another set.
* pop(...)
* Remove and return an arbitrary set element.
* Raises KeyError if the set is empty.
* remove(...)
* Remove an element from a set; it must be a member.
* If the element is not a member, raise a KeyError.
* symmetric\_difference(...)
* Return the symmetric difference of two sets as a new set.
* (i.e. all elements that are in exactly one of the sets.)
* symmetric\_difference\_update(...)
* Update a set with the symmetric difference of itself and another.
* union(...)
* Return the union of sets as a new set.
* (i.e. all elements that are in either set.)
* update(...)
* Update a set with the union of itself and others.
* ----------------------------------------------------------------------
* Class methods defined here:
* \_\_class\_getitem\_\_(...) from builtins.type
* See PEP 585
* ----------------------------------------------------------------------
* Static methods defined here:
* \_\_new\_\_(\*args, \*\*kwargs) from builtins.type
* Create and return a new object. See help(type) for accurate signature.
* ----------------------------------------------------------------------
* Data and other attributes defined here:
* \_\_hash\_\_ = None

help(list)

* elp on class list in module builtins:

class list(object)

* list(iterable=(), /)
* Built-in mutable sequence.
* If no argument is given, the constructor creates a new empty list.
* The argument must be an iterable if specified.
* Methods defined here:
* \_\_add\_\_(self, value, /)
* Return self+value.
* \_\_contains\_\_(self, key, /)
* Return key in self.
* \_\_delitem\_\_(self, key, /)
* Delete self[key].
* \_\_eq\_\_(self, value, /)
* Return self==value.
* \_\_ge\_\_(self, value, /)
* Return self>=value.
* \_\_getattribute\_\_(self, name, /)
* Return getattr(self, name).
* \_\_getitem\_\_(...)
* x.\_\_getitem\_\_(y) <==> x[y]
* \_\_gt\_\_(self, value, /)
* Return self>value.
* \_\_iadd\_\_(self, value, /)
* Implement self+=value.
* \_\_imul\_\_(self, value, /)
* Implement self\*=value.
* \_\_init\_\_(self, /, \*args, \*\*kwargs)
* Initialize self. See help(type(self)) for accurate signature.
* \_\_iter\_\_(self, /)
* Implement iter(self).
* \_\_le\_\_(self, value, /)
* Return self<=value.
* \_\_len\_\_(self, /)
* Return len(self).
* \_\_lt\_\_(self, value, /)
* Return self<value.
* \_\_mul\_\_(self, value, /)
* Return self\*value.
* \_\_ne\_\_(self, value, /)
* Return self!=value.
* \_\_repr\_\_(self, /)
* Return repr(self).
* \_\_reversed\_\_(self, /)
* Return a reverse iterator over the list.
* \_\_rmul\_\_(self, value, /)
* Return value\*self.
* \_\_setitem\_\_(self, key, value, /)
* Set self[key] to value.
* \_\_sizeof\_\_(self, /)
* Return the size of the list in memory, in bytes.
* append(self, object, /)
* Append object to the end of the list.
* clear(self, /)
* Remove all items from list.
* copy(self, /)
* Return a shallow copy of the list.

* count(self, value, /)
* Return number of occurrences of value.
* extend(self, iterable, /)
* Extend list by appending elements from the iterable.
* index(self, value, start=0, stop=9223372036854775807, /)
* Return first index of value.
* Raises ValueError if the value is not present.
* insert(self, index, object, /)
* Insert object before index.
* pop(self, index=-1, /)
* | Remove and return item at index (default last).
* Raises IndexError if list is empty or index is out of range.
* remove(self, value, /)
* Remove first occurrence of value.
* Raises ValueError if the value is not present.
* reverse(self, /)
* Reverse \*IN PLACE\*.
* sort(self, /, \*, key=None, reverse=False)
* Sort the list in ascending order and return None.
* The sort is in-place (i.e. the list itself is modified) and stable (i.e. the
* order of two equal elements is maintained).
* If a key function is given, apply it once to each list item and sort them,
* ascending or descending, according to their function values.
* The reverse flag can be set to sort in descending order.
* ----------------------------------------------------------------------
* Class methods defined here:
* \_\_class\_getitem\_\_(...) from builtins.type
* See PEP 585
* ----------------------------------------------------------------------
* Static methods defined here:
* \_\_new\_\_(\*args, \*\*kwargs) from builtins.type
* Create and return a new object. See help(type) for accurate signature.
* ----------------------------------------------------------------------
* Data and other attributes defined here:
* \_\_hash\_\_ = None
* help(tuple)
* Help on class tuple in module builtins:
* class tuple(object)
* tuple(iterable=(), /)
* Built-in immutable sequence.
* If no argument is given, the constructor returns an empty tuple.
* If iterable is specified the tuple is initialized from iterable's items.
* If the argument is a tuple, the return value is the same object.
* Built-in subclasses:
* asyncgen\_hooks
* UnraisableHookArgs
* Methods defined here:
* \_\_add\_\_(self, value, /)
* Return self+value.
* \_\_contains\_\_(self, key, /)
* Return key in self.
* \_\_eq\_\_(self, value, /)

Return self==value.

* \_\_ge\_\_(self, value, /)
* Return self>=value.
* \_\_getattribute\_\_(self, name, /)
* Return getattr(self, name).
* \_\_getitem\_\_(self, key, /)
* Return self[key].
* \_\_getnewargs\_\_(self, /)
* \_\_gt\_\_(self, value, /)
* Return self>value.
* \_\_hash\_\_(self, /)
* Return hash(self).
* \_\_iter\_\_(self, /)
* Implement iter(self).
* \_\_le\_\_(self, value, /)
* Return self<=value.
* \_\_len\_\_(self, /)
* Return len(self).
* \_\_lt\_\_(self, value, /)
* Return self<value.
* \_\_mul\_\_(self, value, /)
* Return self\*value.
* \_\_ne\_\_(self, value, /)
* Return self!=value.
* \_\_repr\_\_(self, /)
* Return repr(self).
* \_\_rmul\_\_(self, value, /)
* Return value\*self.
* count(self, value, /)
* Return number of occurrences of value.
* index(self, value, start=0, stop=9223372036854775807, /)
* Return first index of value.
* Raises ValueError if the value is not present.
* ----------------------------------------------------------------------
* Class methods defined here:
* \_\_class\_getitem\_\_(...) from builtins.type
* See PEP 585
* ----------------------------------------------------------------------
* Static methods defined here:
* \_\_new\_\_(\*args, \*\*kwargs) from builtins.type
* Create and return a new object. See help(type) for accurate signature.

help(dict)

* elp on class dict in module builtins:

class dict(object)

* dict() -> new empty dictionary
* dict(mapping) -> new dictionary initialized from a mapping object's
* (key, value) pairs
* dict(iterable) -> new dictionary initialized as if via:
* d = {}
* for k, v in iterable:
* d[k] = v
* dict(\*\*kwargs) -> new dictionary initialized with the name=value pairs
* in the keyword argument list. For example: dict(one=1, two=2)
* Methods defined here:
* \_\_contains\_\_(self, key, /)
* True if the dictionary has the specified key, else False.
* \_\_delitem\_\_(self, key, /)
* Delete self[key].
* \_\_eq\_\_(self, value, /)
* Return self==value.
* \_\_ge\_\_(self, value, /)
* Return self>=value.

* \_\_getattribute\_\_(self, name, /)
* Return getattr(self, name).
* \_\_getitem\_\_(...)
* x.\_\_getitem\_\_(y) <==> x[y]
* \_\_gt\_\_(self, value, /)
* Return self>value.
* \_\_init\_\_(self, /, \*args, \*\*kwargs)
* Initialize self. See help(type(self)) for accurate signature.
* \_\_ior\_\_(self, value, /)
* Return self|=value.
* \_\_iter\_\_(self, /)
* Implement iter(self).
* \_\_le\_\_(self, value, /)
* Return self<=value.
* \_\_len\_\_(self, /)
* Return len(self).
* \_\_lt\_\_(self, value, /)
* Return self<value.
* \_\_ne\_\_(self, value, /)
* Return self!=value.
* \_\_or\_\_(self, value, /)
* Return self|value.
* \_\_repr\_\_(self, /)
* | Return repr(self).
* | \_\_reversed\_\_(self, /)
* Return a reverse iterator over the dict keys.
* \_\_ror\_\_(self, value, /)
* Return value|self.

|

* \_\_setitem\_\_(self, key, value, /)
* Set self[key] to value.
* \_\_sizeof\_\_(...)
* D.\_\_sizeof\_\_() -> size of D in memory, in bytes
* clear(...)
* D.clear() -> None. Remove all items from D.

|

* copy(...)
* D.copy() -> a shallow copy of D
* get(self, key, default=None, /)
* Return the value for key if key is in the dictionary, else default.
* items(...)
* D.items() -> a set-like object providing a view on D's items
* keys(...)
* | D.keys() -> a set-like object providing a view on D's keys
* |
* | pop(...)
* | D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
* |
* | If the key is not found, return the default if given; otherwise,
* | raise a KeyError.
* |
* | popitem(self, /)
* | Remove and return a (key, value) pair as a 2-tuple.
* |
* | Pairs are returned in LIFO (last-in, first-out) order.
* | Raises KeyError if the dict is empty.
* |
* | setdefault(self, key, default=None, /)
* | Insert key with a value of default if key is not in the dictionary.
* |
* | Return the value for key if key is in the dictionary, else default.
* |
* | update(...)
* | D.update([E, ]\*\*F) -> None. Update D from dict/iterable E and F.
* | If E is present and has a .keys() method, then does: for k in E: D[k] = E[k]
* | If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v
* | In either case, this is followed by: for k in F: D[k] = F[k]
* |
* | values(...)
* | D.values() -> an object providing a view on D's values
* |
* | ----------------------------------------------------------------------
* | Class methods defined here:
* |
* | \_\_class\_getitem\_\_(...) from builtins.type
* | See PEP 585
* |
* | fromkeys(iterable, value=None, /) from builtins.type
* | Create a new dictionary with keys from iterable and values set to value.
* |
* | ----------------------------------------------------------------------
* | Static methods defined here:
* |
* | \_\_new\_\_(\*args, \*\*kwargs) from builtins.type
* | Create and return a new object. See help(type) for accurate signature.
* |
* | ----------------------------------------------------------------------
* | Data and other attributes defined here:
* |
* | \_\_hash\_\_ = None