dotenv (version 0.0.5)

index
/home/abhishek/.local/lib/python2.7/site-packages/dotenv/__init__.py

```
# encoding=UTF-8
```

Package Contents

__main__

Classes

__builtin__.dict(__builtin__.object)
    Dotenv

class **Dotenv**(__builtin__.dict)
    Method resolution order:
        Dotenv
        __builtin__.dict
        __builtin__.object

    Methods defined here:

    __delitem__(self, key)

    __init__(self, file_path)

    __setitem__(self, key, value)

    Data descriptors defined here:

    __dict__
        dictionary for instance variables (if defined)

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

    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

__eq__(...)
x.__eq__(y) <==> x==y

__ge__(...)
x.__ge__(y) <==> x>=y

__getattribute__(...)
x.__getattribute__('name') <==> x.name

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

__gt__(...)
x.__gt__(y) <==> x>y

__iter__(...)
x.__iter__() <==> iter(x)

__le__(...)
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

clear(...)
D.clear() -> None. Remove all items from D.

copy(...)
D.copy() -> a shallow copy of D

fromkeys(...)
dict.fromkeys(S[,v]) -> New dict with keys from S and values equal to v.
v defaults to None.

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

items(...)
D.items() -> list of D's (key, value) pairs, as 2-tuples

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

keys(...)
D.keys() -> list of D's keys

pop(...)
D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise KeyError is raised

popitem(...)
D.popitem() -> (k, v), remove and return some (key, value) pair as a
2-tuple; but raise KeyError if D is empty.

setdefault(...)
D.setdefault(k[,d]) -> D.get(k,d), also set D[k]=d if k not in D

update(...)
D.update([E,]**F) -> None. Update D from dict/iterable E and F.
If E present and has a .keys() method, does: for k in E: D[k] = E[k]
If E present and lacks .keys() method, 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() -> list of D's values

viewitems(...)
D.viewitems() -> a set-like object providing a view on D's items

viewkeys(...)
D.viewkeys() -> a set-like object providing a view on D's keys

viewvalues(...)
D.viewvalues() -> an object providing a view on D's values

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

Functions

get_variable(file_path, key)

get_variables(file_path)

set_variable(file_path, key, value)

Data

__version__ = '0.0.5'
with_statement = _Feature((2, 5, 0, 'alpha', 1), (2, 6, 0, 'alpha', 0), 32768)