```
"""
Inherit docstrings

Found here: http://code.activestate.com/recipes/578587-inherit-method-docstrings-without-breaking-decorat/

Simple Use:
    1) Import this module
    2) Inherit metaclass InheritableDocstrings
    3) Apply decorator inherit_docstring

Example:
    from lib.inherit_docstring import InheritableDocstrings, inherit_docstring

    class Animal:
        def move_to(self, dest):
            '''Move to *dest*'''
            pass

    class Bird(Animal, metaclass=InheritableDocstrings):
        @inherit_docstring
        def move_to(self, dest):
            self._fly_to(dest)

    assert Animal.move_to.__doc__ == Bird.move_to.__doc__

"""


from functools import partial

# Replace this with actual implementation from
# http://code.activestate.com/recipes/577748-calculate-the-mro-of-a-class/
# (though this will work for simple cases)
def mro(*bases):
    return bases[0].__mro__

# This definition is only used to assist static code analyzers
def inherit_docstring(fn):
    '''Copy docstring for method from superclass

    For this decorator to work, the class has to use the `InheritableDocstrings`
    metaclass.
    '''
    raise RuntimeError('Decorator can only be used in classes  '
                       'using the `InheritableDocstrings` metaclass ')

def _inherit_docstring(mro, fn):
    '''Decorator to set docstring for *fn* from *mro* '''

    if fn.__doc__ is not None:
        raise RuntimeError('Function already has docstring ')

    # Search for docstring in superclass
    for cls in mro:
        super_fn = getattr(cls, fn.__name__, None)
        if super_fn is None:
            continue
        fn.__doc__ = super_fn.__doc__
        break
    else:
        raise RuntimeError("Can't inherit docstring for %s: method does not "
                           "exist in superclass" % fn.__name__)

    return fn

class InheritableDocstrings(type):
    @classmethod
    def __prepare__(cls, name, bases, **kwds):
        classdict = super().__prepare__(name, bases, *kwds)

        # Inject decorators into class namespace
        classdict['inherit_docstring'] = partial(_inherit_docstring, mro(*bases))

        return classdict

    def __new__(cls, name, bases, classdict):

        # Decorator may not exist in class dict if the class (metaclass
        # instance) was constructed with an explicit call to `type`.
        # (cf http://bugs.python.org/issue18334)
        if 'inherit_docstring' in classdict:

            # Make sure that class definition hasn't messed with decorators
            copy_impl = getattr(classdict['inherit_docstring'], 'func', None)
            if copy_impl is not _inherit_docstring:
                raise RuntimeError('No inherit_docstring attribute may be created '
                                   'in classes using the InheritableDocstrings metaclass ')

            # Delete decorators from class namespace
            del classdict['inherit_docstring']

        return super().__new__(cls, name, bases, classdict)
```