Абстрактные методы содержат только определение метода без реализации. Предполагается, что класс-потомок должен переопределить метод и реализовать его функциональность. Чтобы такое предположение сделать более очевидным, часто внутри абстрактного метода возбуждают исключение.

Существуют для того чтобы их переопределили.

## Абстрактные методы

class Class1(object):

def test(self, x): # Абстрактный метод

# Возбуждаем исключение с помощью raise

raise NotImplementedError("Необходимо переопределить метод")

class Class2(Class1): # Наследуем абстрактный метод

def test(self, x): # Переопределяем метод

print x

class Class3(Class1): # Класс не переопределяет метод

pass

c2 = Class2()

c2.test(50) # Выведет: 50

c3 = Class3()

try: # Перехватываем исключения

c3.test(50) # Ошибка. Метод test() не переопределен

except NotImplementedError, msg:

print msg # Выведет: Необходимо переопределить метод

Начиная с версии Python 2.6, в состав библиотеки входит модуль abc. В этом модуле определен декоратор @abstractmethod, который позволяет указать, что метод, перед которым расположен декоратор, является абстрактным. При попытке создать экземпляр класса-потомка, в котором не переопределен абстрактный метод, возбуждается исключение TypeErrror. Рассмотрим использование декоратора @abstractmethod на примере.

## Использование декоратора @abstractmethod

from abc import ABCMeta, abstractmethod

class Class1(object):

\_\_metaclass\_\_ = ABCMeta

@abstractmethod

def test(self, x): # Абстрактный метод

pass

class Class2(Class1): # Наследуем абстрактный метод

def test(self, x): # Переопределяем метод

print x

class Class3(Class1): # Класс не переопределяет метод

pass

c2 = Class2()

c2.test(50) # Выведет: 50

try:

c3 = Class3() # Ошибка. Метод test() не переопределен

c3.test(50)

except TypeError, msg:

print msg # Can't instantiate abstract class Class3

# with abstract methods test

Источник: http://python-3.ru/page/abstraktnye-metody