Управление доступом

Управление доступом к полям и методам в Python это особые синтаксические или конвенционные подходы к названиям полей и методов классов для ограничения доступа к ним извне.

Публичный доступ

К полям и методам с публичным доступом может получить доступ любая сущность, имеющая достаточную область видимости. Для обозначения полей и методов публичными ничего делать не нужно:

```
class Dummy:
  def __init__(self):
    self.some_field = 42 # публичное поле

  def some_method(self): # публичный метод
    pass

d = Dummy() // для доступа к полям и методам нужен экземпляр
d.some_field = 100 # можно
d.some_method() # можно
```

Приватный доступ

К приватным полям и методам можно получить доступ только из методов того же класса. Для обозначения приватных полей и методов их название начинают с как минимум двух знаков подчеркивания и заканчивают не более одним знаком подчеркивания (обычно вообще без них). Это синтаксическая конструкция, то есть интерпретатор Python будет осуществлять проверку доступа.

```
class Dummy:
  def __init__(self):
    self.__some_field = 42 # приватное поле

  def some_method(self):
    self.__some_field = 100 # можно

d = Dummy()
d.some_method() # можно
d.__some_field = 42 # ошибка доступа
```

Стоит помнить, что приватный уровень доступа распространяется на класс, а не на конкретный объект, поэтому изнутри метода можно обращаться к приватным полям и методам и другого экземпляра того же класса:

```
class Dummy:
  # …
  def other_method(self, other_dummy):
    other_dummy.__some_field = 42 # можно
  # ...
```

Защищенный доступ

К полям и методам с защищенным доступом может получить доступ сам класс из своих методов, или любой его класс наследник. Подробнее о наследовании мы поговорим позже в курсе. Для обозначения защищенных полей и методов, их имя нужно начать с одиночного подчеркивания. Стоит помнить, что это конвенционный уровень доступа, т.е. интерпретатор Python не будет на самом деле ограничивать доступ.

```
class DummyParent:
  def __init__(self):
    self.__private_field = 0 # приватное поле
    self._protected_field = 1 # публичное поле

  def some_method(self):
    self.__private_field = 2 # можно, т.к. тот же класс
    self._protected_field = 3 # можно, т.к. тот же класс

class DummyChild(DummyParent): # наследник
  def child_method(self):
    self._protected_field = 4 # можно, т.к. наследник
    self.__private_field = 5 # создаст новое поле

p = DummyParent()
c = DummyChild()

p.some_method() # можно
p._protected_field = 6 # возможно, но не рекомендуется
c.some_method() # можно
c.child_method() # можно
c._protected_field = 7 # возможно, но не рекомендуется
```

Модификаторы уровня модуля

В Python все сущности модуля экспортируются, то есть могут быть импортированы в других модулях. Если начать имя сущности (глобальной переменной) с одинарного знака подчеркивания, то оно не будет автоматически импортироваться конструкцией `from mymodule import *`, но будет доступно при целенаправленном импорте:

```
# dummy.py
class _Dummy:
  pass

# другой файл
from dummy import * # ничего не импортируется
from dummy import _Dummy # сработает
import dummy # можно использовать dummy._Dummy
```

Настройка доступа отдельных полей и методов на уровне модуля не поддерживается.