Режимы доступа. Шпаргалка

Режимы доступа

```
class Employee:

    def __init__(self, first, last, pay):
        self.__first = first # приватный атрибут, используется только для доступа внутри объекта
        self.__last = last # приватный атрибут, используется только для доступа внутри объекта
        self._email = f'{self.first}.{self.last}@email.com' # защищенный атрибут
        self.pay = pay # публичный атрибут
```

Типы методов

```
import datetime

class Employee:

    raise_amt = 1.04

    def __init__(self, first, last, pay):
        """ Если описанная функциональность использует свойства объекта, """
        """ то необходимо создавать обычный метод """
        self.first = first
        self.last = last
        self.email = first + '.' + last + '@email.com'
        self.pay = pay

        Employee.num_of_emps += 1


    @classmethod
    def set_raise_amt(cls, amount):
        """ При необходимости использовать класс (атрибуты класса или сам класс) """
        """ создаются класс-методы """
        cls.raise_amt = amount

    @staticmethod
    def is_workday(day):
        """ В случае, если нет необходимости использовать объект или класс, """
        """ создается статик-метод """
        if day.weekday() == 5 or day.weekday() == 6:
            return False
        return True
```

Работа с геттерами, сеттерами и делитерами

```
class Employee:

    def __init__(self, first, last):
        self.first = first
        self.last = last

    @property
    def email(self):
        return f'{self.first}.{self.last}@email.com'

    @property
    def fullname(self):
        return f'{self.first} {self.last}'

    @fullname.setter
    def fullname(self, new_fn):
        first, last = new_fn.split(' ')
        self.first = first
        self.last = last

    @fullname.deleter
    def fullname(self):
        self.first = None
        self.last = None


emp = Employee('Ivan', 'Ivanov')

print(emp.fullname)

emp.fullname = 'Petr Petrov'
print(emp.fullname)

del emp.fullname
print(emp.fullname)
```