ПРОГРАММИРОВАНИЕ В PYTHON

ООП в Python. Методы SUPER(), GETTER(), SETTER().

МЕТОД SUPER()

```
[18] class A:
         def some_method(self):
             print('Spam, eggs!!1')


     class B(A):
         def some_method(self):
             print('Hello, World!')


     x = B()
     x.some_method()
```

было

Стало!

```
[19] class A:
         def some_method(self):
             print('Spam, eggs!!1')


     class B(A):
         def some_method(self):
             super().some_method()
             print('Hello, World!')


     x = B()
     x.some_method()
```

- **super()** - это функция, которая обращается к классу, от которого наследуется текущий
- С помощью **super().some_method()** мы вызвали родительский метод, а после дополнили свой. Именно для этого чаще всего используется эта функция
- **__init__()** - это метод инициализации класса, следовательно **super().__init__()** вызывает метод инициализации из родительского класса. Например, чтобы дополнить его.

python™

SUPER() ДЛЯ КОНСТРУКТОРА

```
class Human:

  def __init__(self):
    self.__name = ''
    self.__surname = ''
    self.__age = 0
    self.__city = ''
```

```
class Student(Human):

  def __init__(self):

    super().__init__()
    self.__grade = 0
    self.__scores = {}

    #Human.__init__(self, name, surname, age, city)
```

SETTER()

```
class Geek:
    def __init__(self, age = 0):
        self._age = age

    # setter method
    def set_age(self, x):
        self._age = x
```

- SETTER используется внутри класса, чтобы установить значение в атрибут данных.
- Например раньше мы устанавливали значения внутри конструктора класса, что является не совсем верным способом. Теперь конструктор будет использован по значению (инициализировать атрибуты), а устанавливать значения мы будем в сеттерах.
- Кроме того, для можно использовать общий сеттер (один метод, который устанавливает значения для всех атрибутов) или частные сеттеры (устанавливают значения конкретно для одного атрибута).

GETTER()

```
class Geek:
    def __init__(self, age = 0):
        self._age = age

    # getter method
    def get_age(self):
        return self._age

    # setter method
    def set_age(self, x):
        self._age = x
```

- Getter() (от слова get - получить) используется внутри класса, чтобы получить значение атрибута данных.
- Раньше чтобы получить доступ к атрибуту мы просто вызывали его в объекте через точку (Geek.age), поскольку атрибут был публичным. Однако getter позволяет получить доступ к атрибуту с приватным модификатором доступа.

python™

ЗАДАНИЕ

- Реализовать базовый класс Device (устройство), который содержит атрибуты: вид устройства, память и год выпуска. Сделать эти атрибуты приватными. Реализовать метод инфо, который выводит на экран информацию об устройстве.

- Создать геттеры и сеттеры для этого класса

- Реализовать дочерний класс Phone от класса Device, который содержит атрибуты: марка, модель. Унаследовать конструктор от базового класса. Реализовать геттеры и сеттеры для класса. Унаследовать метод info и дополнить его выводом марки и модели телефона.