ООП: Инкапсуляция

Практикум по программированию

Инкапсуляция

Инкапсуляция — ограничение доступа к составляющим объект компонентам (методам и переменным). Инкапсуляция делает некоторые из компонент доступными только внутри класса.

Инкапсуляция

Одиночное подчеркивание в начале имени атрибута говорит о том, что переменная или метод не предназначен для использования вне методов класса, однако атрибут доступен по этому имени.

```
class A:
    def _private(self):
        print("Это приватный метод!")

>>> a = A()
>>> a._private()
Это приватный метод!
```

Двойное подчеркивание в начале имени атрибута даёт большую защиту: атрибут становится недоступным по этому имени.

```
>>> class B:
...     def __private(self):
...         print("Это приватный метод!")
...
>>> b = B()
>>> b.__private()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'B' object has no attribute '__private'
```

Инкапсуляция: геттеры и сеттеры

```
class Car:

    __maxspeed = 0
    __name = ""

    def __init__(self):
        self.__maxspeed = 200
        self.__name = "Supercar"

    def drive(self):
        print 'driving. maxspeed ' + str(self.__maxspeed)

redcar = Car()
redcar.drive()
redcar.__maxspeed = 10  # will not change variable because its private
redcar.drive()
```

Инкапсуляция: геттеры и сеттеры

```
class Car:

    __maxspeed = 0
    __name = ""

    def __init__(self):
        self.__maxspeed = 200
        self.__name = "Supercar"

    def drive(self):
        print 'driving. maxspeed ' + str(self.__maxspeed)

    def setMaxSpeed(self,speed):
        self.__maxspeed = speed

redcar = Car()
redcar.drive()
redcar.setMaxSpeed(320)
redcar.drive()
```

Инкапсуляция: геттеры и сеттеры

```
class Person:
    def __init__(self, name, age):
        self.name = name  # устанавливаем имя
        self.age = age    # устанавливаем возраст

    def display_info(self):
        print("Имя:", self.name, "\tВозраст:", self.age)


tom = Person("Tom", 23)
tom.name = "Человек-паук"       # изменяем атрибут name
tom.age = -129                  # изменяем атрибут age
tom.display_info()              # Имя: Человек-паук       Возраст: -129
```

Инкапсуляция: геттеры и сеттеры

```
class Person:
    def __init__(self, name, age):
        self.__name = name      # устанавливаем имя
        self.__age = age        # устанавливаем возраст

    def set_age(self, age):
        if age in range(1, 100):
            self.__age = age
        else:
            print("Недопустимый возраст")

    def get_age(self):
        return self.__age

    def get_name(self):
        return self.__name

    def display_info(self):
        print("Имя:", self.__name, "\tВозраст:", self.__age)
```

```
tom = Person("Tom", 23)

tom.__age = 43          # Атрибут age
не изменится
tom.display_info()      # Имя: Tom
Возраст: 23
tom.set_age(-3486)      #
Недопустимый возраст
tom.set_age(25)
tom.display_info()      # Имя: Tom
Возраст: 25
```