# Элементы объектно-ориентированного программирования

НГТУ НЭТИ

Новосибирский государственный технический университет НЭТИ

**Ларьков Алексей Сергеевич**

**Объектно-ориентированное программирование (ООП)** - парадигма программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

# Особенности

- Увеличение уровня абстракции и читаемости кода
- Структурирование кода
- Поддержка, модифицируемость, расширяемость.

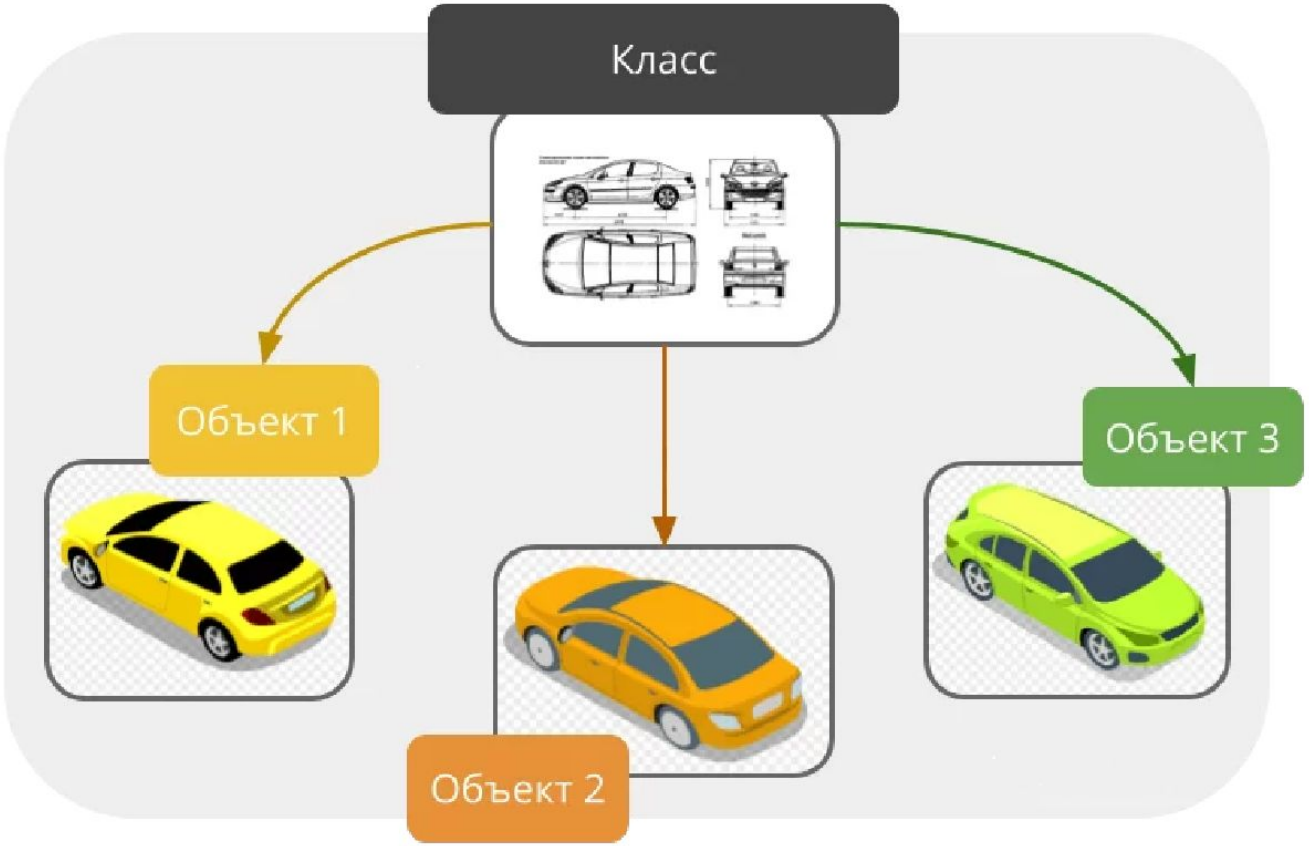
# Преимущества и недостатки ООП

- Повторное использование
- Модулярный подход
- Дебагинг
- Инкапсуляция данных

- Подробное представление о ПО
- Сложность программы
- Не каждый аспект ПО является лучшем решением для реализации в качестве объекта

# Объекты

**Объект** — сущность, которая принадлежит некоторой области исследования и имеет определенный набор свойств и методов.



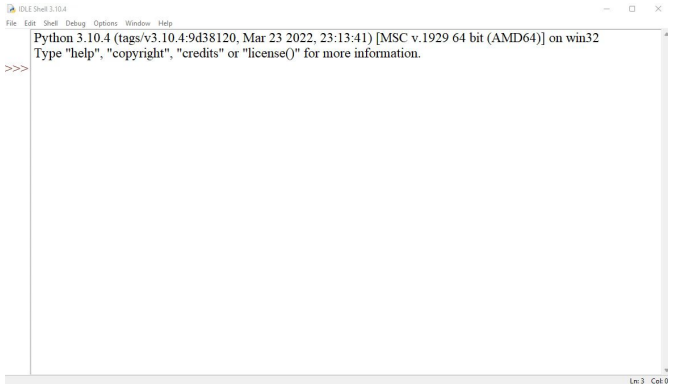
# Класс

**Класс** — это тип данных, состоящий из набора атрибутов (свойств) и методов — функций для работы с этими атрибутами

```
class Class_name:
    Variable_name =  value
    ....

    def Metod_Name (self, ...):
        self.Variable_name = value
        ....

    ....
```

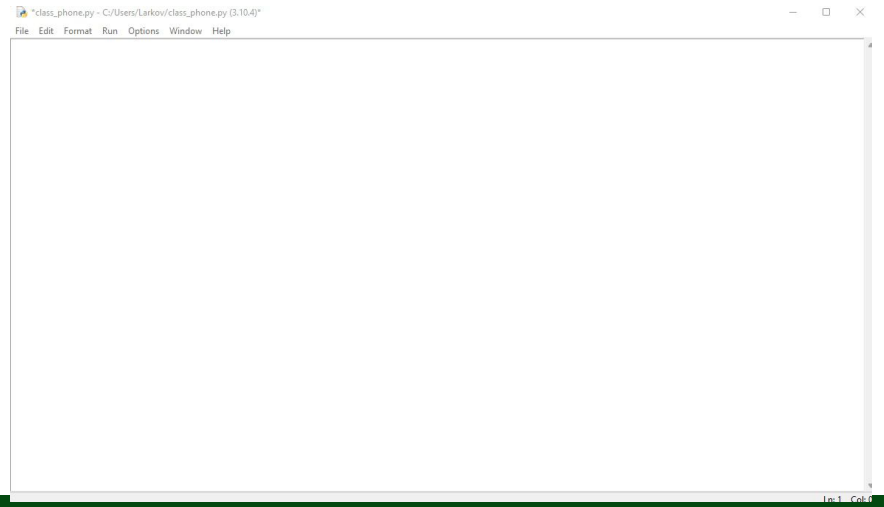


# Атрибуты

Атрибуты класса — свойства объектов, которыми будут обладать вновь созданные экземпляры класса.

**self** – ссылка на текущий экземпляр класса.

Передается в качестве первого параметра метода.



# Методы класса

```
class Student:
        def __init__(self, name, surname):
            self.name = name
            self.surname = surname

stud_1 = Student("Alex", "Ivanov")
print (stud_1.name, stud_1.surname)
```

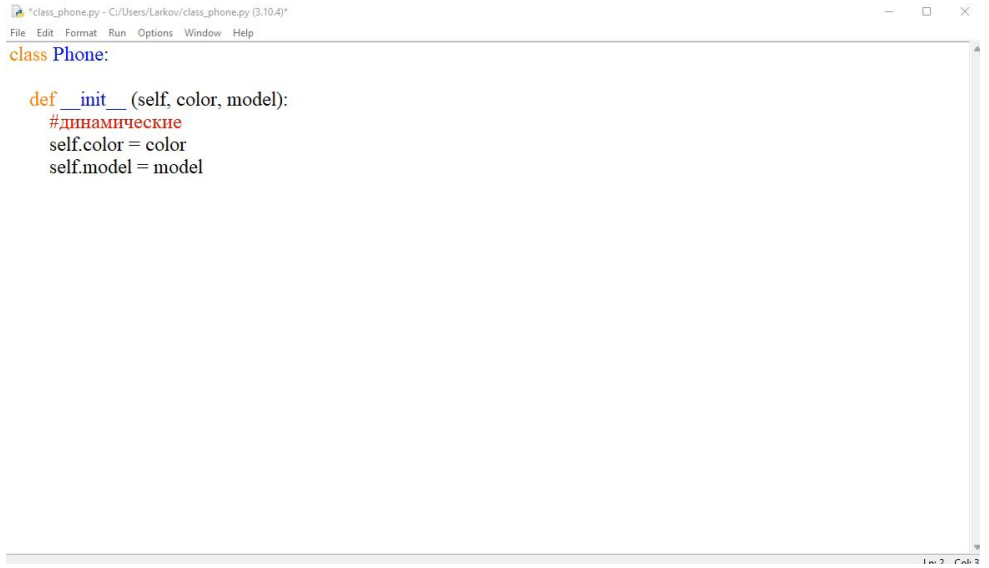
```
Alex Ivanov
```

```
class Student:
        def names (self, name, surname):
            self.name = name
            self.surname = surname
stud_1 = Student ()
stud_1.names ("Alex", "Ivanov")
print (stud_1.name, stud_1.surname)
```

```
Alex Ivanov
```

# Методы касса

- Обычные методы
- Статические методы
- Методы класса
- Специальные методы



| Атрибут | Назначение | Тип |
|---|---|---|
| __new__(cls[, ...]) | Конструктор. Создает экземпляр(объект) класса. Сам класс передается в качестве аргумента. | Функция |
| __init__(self[, ...]) | Инициализатор. Принимает свежесозданный объект класса из конструктора. | Функция |
| __del__(self) | Деструктор. Вызывается при удалении объекта сборщиком мусора | Функция |
| __str__(self) | Возвращает строковое представление объекта. | Функция |
| __hash__(self) | Возвращает хэш-сумму объекта. | Функция |
| __setattr__(self, attr, val) | Создает новый атрибут для объекта класса с именем attr и значением val | Функция |
| __doc__ | Документация класса. | Строка (тип str) |
| __dict__ | Словарь, в котором хранится пространство имен класса | Словарь (тип dict) |

# Принципы ООП

- Абстракция
- Инкапсуляция
- Наследование
- Полиморфизм

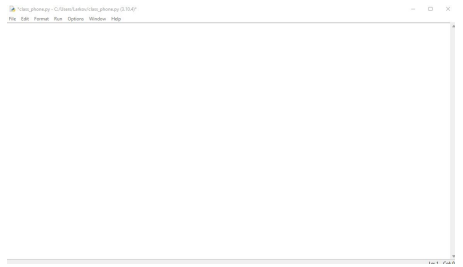
# Инкапсуляция

**Инкапсуляция** - принцип ООП, согласно которому сложность реализации программного компонента должна быть спрятана за его интерфейсом.

**Защищенные атрибуты** (protected) - **_name**

**Приватные атрибуты** (private) - **__name**

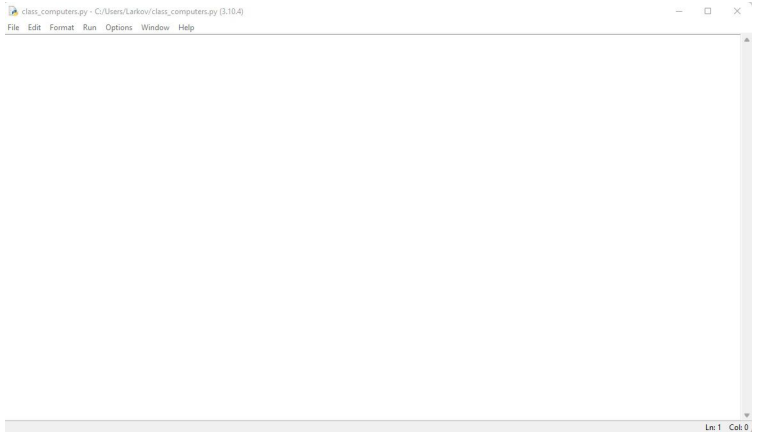
**Публичные атрибуты** (public) - **name**



# Наследование

**Наследование** - способ создания нового класса на основе уже существующего, при котором класс-потомок заимствует свойства и методы родительского класса и также добавляет собственные.

**class** <имя_нового_класса>(<имя_родителя>):



# Полиморфизм

**Полиморфизм -** это поддержка нескольких реализаций на основе общего интерфейса.