Наследование

С технической стороны неследование в языках программирование это средство повторного использования кода. В наследовании участвуют два класса (в некоторых языках больше). Класс, от которого наследуют, называется *предком* или *суперклассом*, класс, который наследует, – *потомком* или *подклассом*. При наследовании потомок получает все поля, свойства и методы предка.

Из-за того, что потомок унаследовал все поля и методы предка, он имеет тот же самый интерфейс, что и предок, таким образом может использоваться везде, где допустимо использование предка. Таким образом с семантической стороны между предком и потомком возникает связь “является” (“is-a”): потомок “является” предком. Например: класс Студент наследует класс Человек. Очевидно, что студент “является” человеком.

При наследовании область видимости полей предка сохраняется. К приватным полям предка потомок не имеет прямого доступа.

В Python конструктор наследуется, если не объявлен явно.

```
class Person: # предок
    def __init__(self, name):
        self.name = name

    def get_name(self):
        return self.name

class Student(Person): # потомок
    def __init__(self, course):
        # реализацию разберем позже

    def get_course(self):
        return self.course

# использование
s = Student(“Иванов”, “Программирование”)
print(s.get_course()) # свой метод
print(s.get_name()) # унаследованный метод
```

Реализация конструктора потомка

Потомок должен вызвать конструктор предка и передать в него разумные значения в качестве параметров. Вызов конструктора предка осуществляется вызовом конструкции super().

```
class Student(Person):
    # …
    def __init__(self, name, course):
        super().__init__(name) # вызов конструктора предка
        self.course = course # остальная инициализация
    # …
```