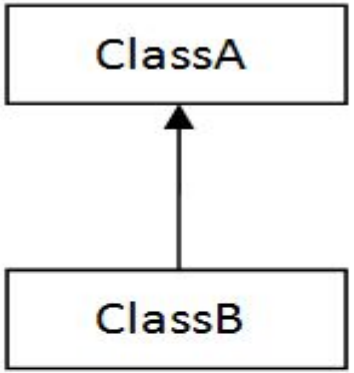
Python
2-месяц 3-урок

Тема: Магические методы в классах и Множественное наследование

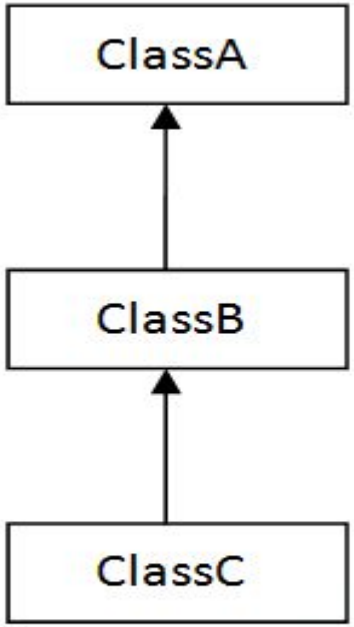
Множественное наследование

Язык программирования **Python** являясь языком, поддерживающим парадигму объектно ориентированного программирования (ООП), также поддерживает и возможность **множественного наследования**. То есть, возможность у класса потомка наследовать функционал не от одного, а от нескольких родителей. Благодаря этому мы можем создавать сложные структуры, сохраняя простой и легко-поддерживаемый код.

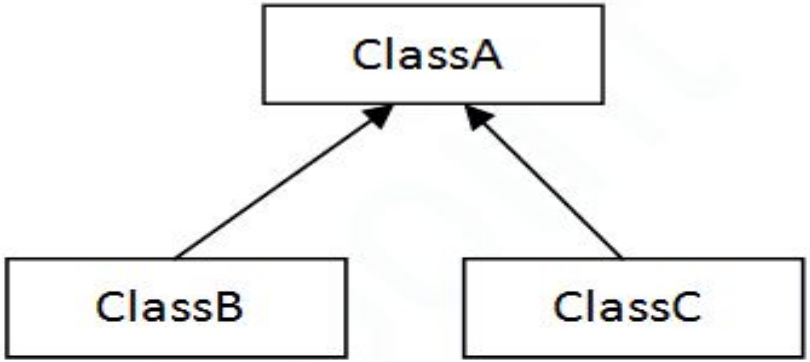
Виды Наследования



1) Single

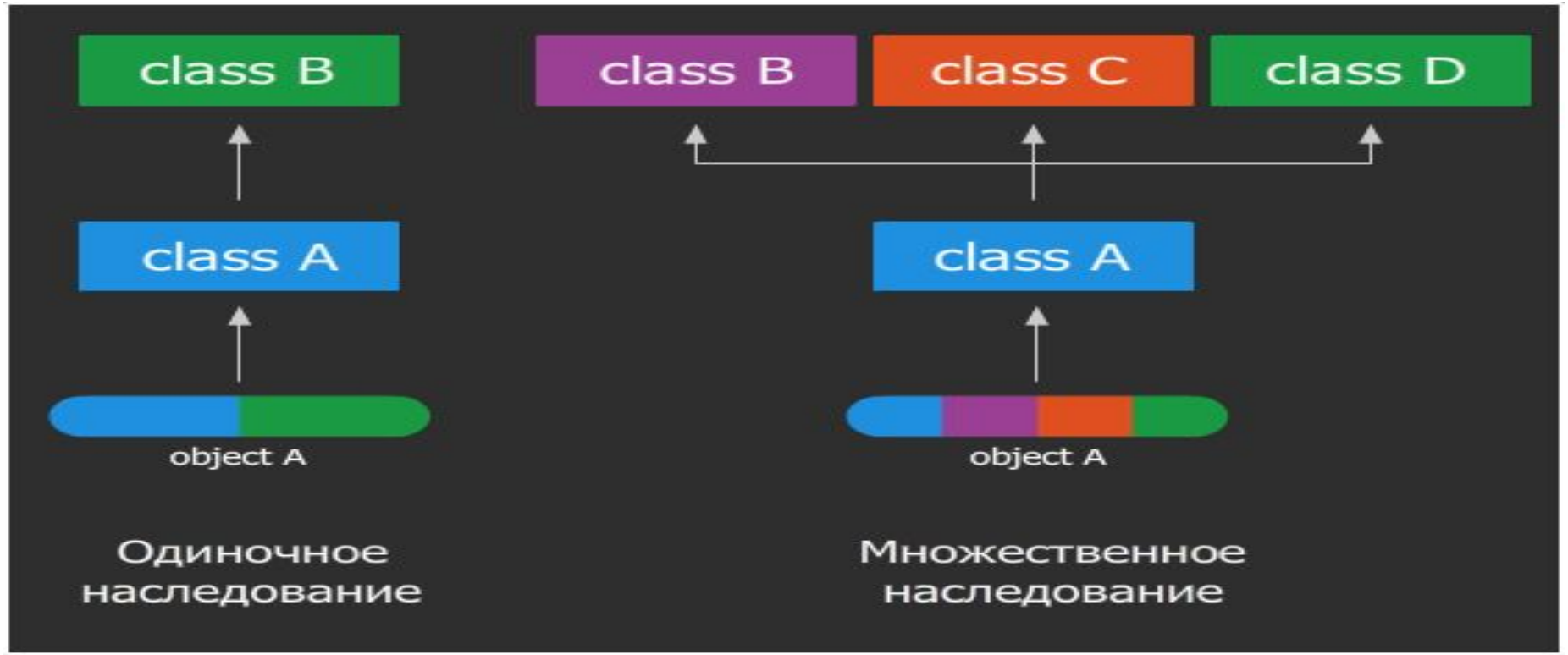


2) Multilevel



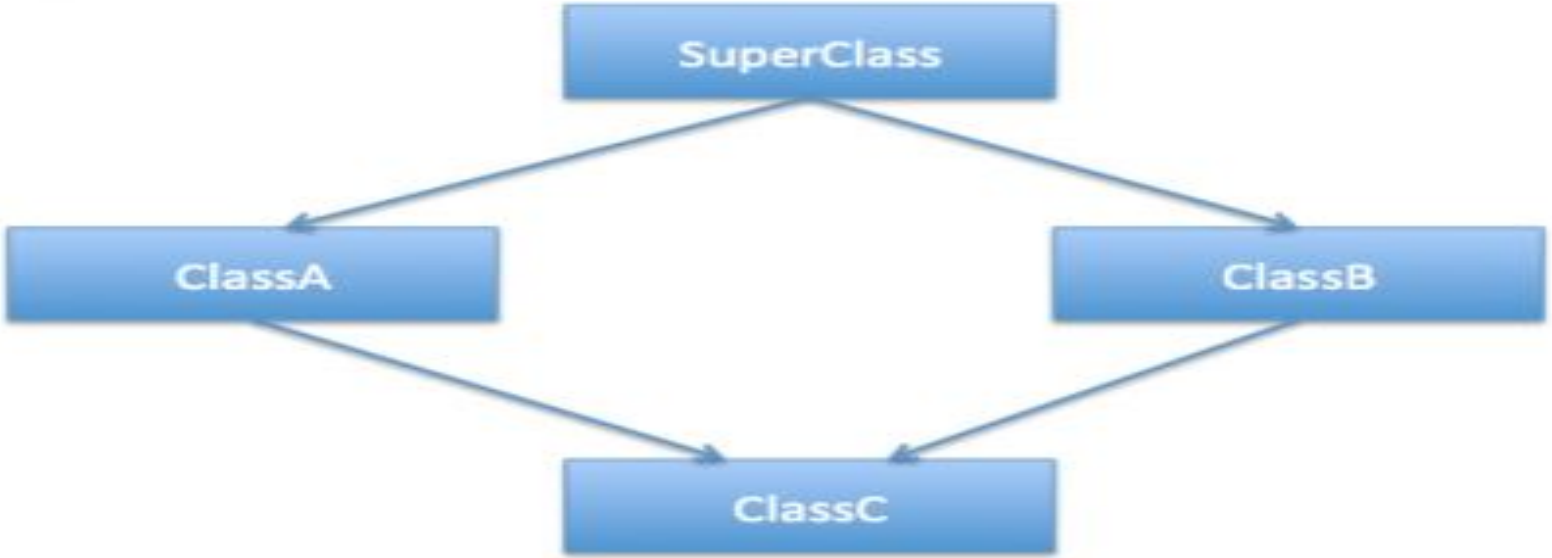
3) Hierarchical

Виды Множественного Наследования



Пример Множественного Наследования

Ромбовидный



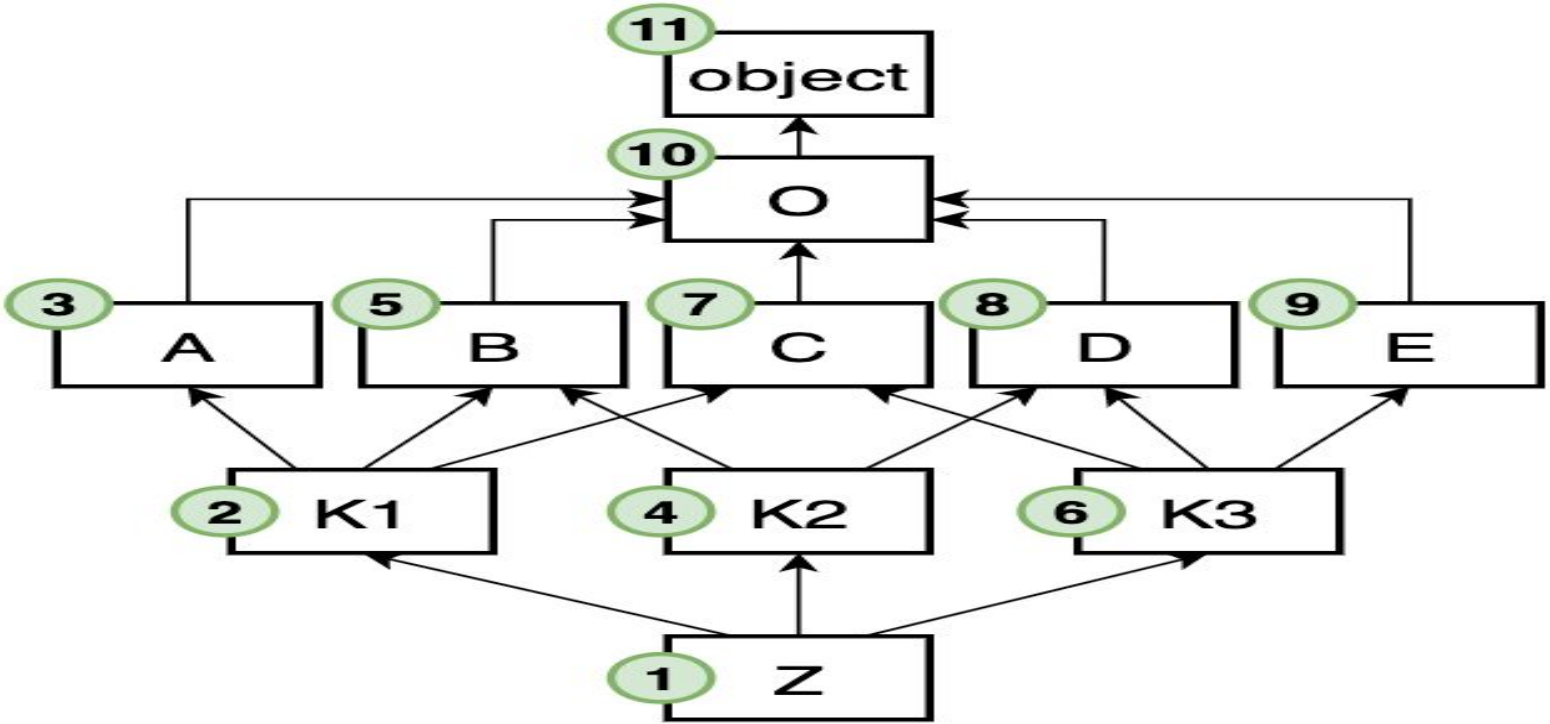
Примеры Множественного Наследования

```
class Smartphone(MusicPlayerMixin):
    pass
```

```
class Radio(MusicPlayerMixin):
    pass
```

```
class Amphibian(Auto, Boat, MusicPlayerMixin):
    pass
```

Почему стоит быть осторожным !!!!!



Магические методы к классах

Что такое магические методы? Они всё в объектно-ориентированном Питоне. Это специальные методы, с помощью которых вы можете добавить в ваши классы «магию». Они всегда обрамлены двумя нижними подчеркиваниями (например, __init__ или __lt__). Ещё, они не так хорошо документированы, как хотелось бы. Все магические методы описаны в документации, но весьма беспорядочно и почти безо всякой организации. Поэтому, чтобы исправить то, что я воспринимаю как недостаток документации Питона, я собираюсь предоставить больше информации о магических методах, написанной на понятном языке и обильно снабженной примерами. Надеюсь, это руководство вам понравится. Используйте его как обучающий материал, памятку или полное описание. Я просто постарался как можно понятнее описать магические методы.

Примеры магических методов (init)

__init__(self, [...)

Инициализатор класса. Ему передаётся всё, с чем был вызван первоначальный конструктор (так, например, если мы вызываем x = SomeClass(10, 'foo'), __init__ получит 10 и 'foo' в качестве аргументов. __init__ почти повсеместно используется при определении классов.

Примеры Магических методов

- __sub__ for subtraction(-)
- __mul__ for multiplication(*)
- __truediv__ for division(/)
- __eq__ for equality (==)
- __lt__ for less than(<)
- __gt__ for greater than(>)
- __le__ for less than or equal to (≤)
- __ge__ for greater than or equal to (≥)

Примеры Магических методов

```
def __add__(self, other):

    if isinstance(other, int) or isinstance(other, float):

        return Complex(self.re + other,self.im)

    elif  isinstance(other, Complex):

        return Complex(self.re + other.re , self.im + other.im)

    else:

        raise TypeError
```

Примеры Магических методов

```
def __sub__(self, other):
    if isinstance(other, int) or isinstance(other, float):
        return Complex(self.re - other,self.im)
    elif  isinstance(other, Complex):
        return Complex(self.re - other.re, self.im - other.im)
    else:
        raise TypeError
```

Примеры Магических методов

```
def __mul__(self, other):

    if isinstance(other, int) or
isinstance(other, float):

        return Complex(self.re *
other, self.im * other)

    elif isinstance(other,
Complex):

#   (a+bi)*(c+di) = ac + adi +bic -bd

    return Complex(self.re * other.re - self.im * other.im,

                self.re * other.im + self.im * other.re)

else:

    raise TypeError
```

Домашнее Задание

Задание № 1 Множественное Наследование (Ромбовидный)

1. Создать **4 класса** и с помощью их преобразить РОМБОВИДНЫЙ тип множественного наследования
2. У супер класса должны быть **не меньше 4 атрибутов**
3. У каждого класса должно быть **не меньше 2 методов**
4. Также должны быть соблюдены **def __str__(self) + super()**
5. И к каждому классу создать объект (в итоге будет не меньше **4 объектов**)

Домашнее Задание

Задание № 2 Множественное Наследование (Один ко многим)

1. Создать **4 класса** , один из них является **супер-классом** , от которого наследуются все остальные **3 класса**
2. У супер класса должно быть как минимум **4 атрибута** (**def __init__(self, atribut, atribut2):**)
3. У каждого класса должно быть как минимум **2 метода** (**def method(self):**)
4. Также должны быть соблюдены **def __str__(self) + super()**
5. К каждому классу создать по одному объекту (**в итоге должно быть 4 объекта**)

Домашнее Задание

Задание № 3 Магические методы

1. Создать 2 класса с магическими методами
2. 1-ый класс это класс Кинотеатр , в котором будут фильмы и нужно использовать как минимум 2 магических метода для выведения фильма
3. Должен быть выбор фильмов и цена у каждого разный
4. 2-ой класс это класс Старбакс в котором пишут имена на кофе
5. Если имя больше или равно 9 символов пишут только 5 символов имени
6. Если имя меньше 9 символов пишут все символы имени
7. Если имя меньше 5 пишут последние три символа имени
8. **Использовать именно магические методы**