Теоретический материал к занятию Основы ООП 4 занятие.

Представим ситуацию, что у нас есть родительский и дочерние классы, а в каждом из них есть метод с одинаковым названием. Какой метод вызовется при обращении через экземпляр дочернего класса? Правильный ответ - метод из дочернего класса. Это и есть так называемое переопределение методов. Когда мы наследуемся от какого то класса и изменяем поведение метода так, как на требуется

Рассмотрим пример

```
class Parent:
    def say_hello():
        print('Привет я метод родительского класса')

class Children(Parent):
    def say_hello():
        print('Привет я метод дочернего класса')


child = Children
child.say_hello()
```

Результат

Привет я метод дочернего класса

Перегрузка операторов — один из способов реализации полиморфизма, когда мы можем задать свою реализацию какого-либо метода в своём классе.
Полиморфизм – это способность одного и того же объекта вести себя по-разному в зависимости от того, в контексте какого класса он используется.

Для начала список методов с двойным подчеркиванием, в которых мы можем изменить поведение

```
__new__(cls[, ...]) — управляет созданием экземпляра. В качестве
обязательного аргумента принимает класс (не путать с экземпляром). Должен
возвращать экземпляр класса для его последующей передачи методу __init__.
__init__(self[, ...]) - конструктор.
__del__(self) - вызывается при удалении объекта сборщиком мусора.
__repr__(self) - вызывается встроенной функцией repr; возвращает "сырые"
данные, использующиеся для внутреннего представления в python.
__str__(self) - вызывается функциями str, print и format. Возвращает
строковое представление объекта.
__format__(self, format_spec) - используется функцией format (а также методом
format у строк).
__lt__(self, other) - x < y .
__le__(self, other) - x ≤ y.
__eq__(self, other) - x == y.
__ne__(self, other) - x != y
```

__gt__(self, other) - x > y.
__ge__(self, other) - x >= y.
__hash__(self) - получение хэш-суммы объекта, например, для добавления в словарь.
__bool__(self) - вызывается при проверке истинности. Если этот метод не определён, вызывается метод __len__ (объекты, имеющие ненулевую длину, считаются истинными).
__getattr__(self, name) - вызывается, когда атрибут экземпляра класса не найден в обычных местах (например, у экземпляра нет метода с таким названием).
__setattr__(self, name, value) - назначение атрибута.
__delattr__(self, name) - удаление атрибута (del obj.name).
__call__(self[, args...]) - вызов экземпляра класса как функции.
__len__(self) - длина объекта.
__getitem__(self, key) - доступ по индексу (или ключу).
__setitem__(self, key, value) - назначение элемента по индексу.
__delitem__(self, key) - удаление элемента по индексу.
__iter__(self) - возвращает итератор для контейнера.
__reversed__(self) - итератор из элементов, следующих в обратном порядке.
__contains__(self, item) - проверка на принадлежность элемента контейнеру (item in self).

Теперь рассмотрим таблицу для перегрузки математических операторов
__add__(self, other) - сложение. x + y.
__sub__(self, other) - вычитание (x - y).
__mul__(self, other) - умножение (x * y).
__truediv__(self, other) - деление (x / y).
__floordiv__(self, other) - целочисленное деление (x // y).
__mod__(self, other) - остаток от деления (x % y).
__divmod__(self, other) - частное и остаток (divmod(x, y)).
__pow__(self, other[, modulo]) - возведение в степень (x ** y, pow(x, y[, modulo])).
__iadd__(self, other) - +=.
__isub__(self, other) - -=.
__imul__(self, other) - *=.
__itruediv__(self, other) - /=.
__ifloordiv__(self, other) - //=.
__imod__(self, other) - %=.
__ipow__(self, other[, modulo]) - **=.

Это не полная таблица, но для примера нам подойдет. Давайте попробуем заставить + например умножать

```
class Test(int):
    def __init__(self, num) -> None:
        super().__init__()
        self.num  = num
    def __add__(self, num2):
        return self.num * num2


a = Test(5)
```

```
print(a + 10)
```