?????????? ??????

Все объекты. Тип данных - по сути правило операций (функций) для одного типа. Для определения используемых при написании конструкций (например +, *, ...) используются магические методы. Очень интересная возможность.

Магические методы

1. Инициализация и управление объектом

__new__(cls, ...)	Создание экземпляра (до __init__)
__init__(self, ...)	Инициализация объекта
__del__(self)	При удалении объекта (деструктор, вызывается редко)
__repr__(self)	Для разработчиков (repr()) — однозначное представление
__str__(self)	Для пользователей (str(), print()) — читаемое представление
__bytes__(self)	bytes()
__format__(self, spec)	format() и f-строки
__hash__(self)	hash() (если None — объект нехэшируемый)
__eq__(self, other)	==
__lt__(self, other)	<
__le__(self, other)	<=
__gt__(self, other)	>
__ge__(self, other)	>=
__bool__(self)	bool(), if obj: (если нет — проверяет __len__)
__len__(self)	len()
__contains__(self, item)	in (если нет — перебирает через __iter__)
__getitem__(self, key)	self[key], for i in obj:
__setitem__(self, key, value)	self[key] = value
__delitem__(self, key)	del self[key]
__iter__(self)	Итератор (for x in obj:)
__next__(self)	next()
__reversed__(self)	reversed()

2. Арифметические операторы

Возвращается всегда новая переменная. Существует три варианта в случае арифметических операций:

- Базовый. Вызывается первым, для объекта слева. Объект справа может быть любого типа. Внутри нужно проверять принадлежность к классу для второго аргумента. В случае ошибки / несоответствия типов / ... вызывается NotImplemented.

```
class MyClass:
    def __add__(self, other):
        if isinstance(other, MyClass):
            return MyClass(self.value + other.value)
        elif isinstance(other, (int, float)):
            return MyClass(self.value + other)
        else:
            # Не знаю, что делать
            return NotImplemented
```

- Если был вызван NotImplemented, право сделать что-то передается объекту справа (правые версии). Если правая версия не определена, то будет TypeError. Но если порядок не важен, можно сделать так:

```
class MyClass:
...
    def __radd__(self, other):
        return self.__add__(other)
```

- Существует способ записи например += Для него может быть отдельный (расширенный) метод. Если не определять, то вызывается соответствующий базовый метод. Можно менять сам объект. Обязательно возвращать объект.

Операция	**Базовые методы**	**Правые версии**		**Расширенные версии**	
+	__add__(self, other)	__radd__(self, other)	other + self	__iadd__(self, other)	+=
-	__sub__(self, other)	__rsub__(self, other)	other - self	__isub__(self, other)	-=
*	__mul__(self, other)	__rmul__(self, other)	other * self	__imul__(self, other)	*=
/	__truediv__(self, other)	__rtruediv__(self, other)	other / self	__itruediv__(self, other)	/=

Операция	Базовые методы	Правые версии		Расширенные версии	
//	__floordiv__(self, other)	__rfloordiv__(self, other)	other // self	__ifloordiv__(self, other)	//=
%	__mod__(self, other)	__rmod__(self, other)	other % self	__imod__(self, other)	%=
**, pow()	__pow__(self, other[, modulo])	__rpow__(self, other)	other ** self	__ipow__(self, other)	**=
-obj	__neg__(self)				
+obj	__pos__(self)				
abs()	__abs__(self)				
round()	__round__(self[, n])				

3. Битовые операторы

Операция	Базовые методы	Правые версии	Расширенные версии	
&	__and__(self, other)	__rand__(self, other)	__iand__(self, other)	&=
|	__or__(self, other)	__ror__(self, other)	__ior__(self, other)	|=
^	__xor__(self, other)	__rxor__(self, other)	__ixor__(self, other)	^=
<<	__lshift__(self, other)	__rlshift__(self, other)	__ilshift__(self, other)	<<=
>>	__rshift__(self, other)	__rrshift__(self, other)	__irshift__(self, other)	>>=
~	__invert__(self)			

4. Методы для контекстного менеджера (``with``)

__enter__(self)	Вход в контекст
__exit__(self, exc_type, exc_val, exc_tb)	Выход из контекста (с обработкой исключений)

5. Работа с атрибутами

__getattr__(self, name)	при обращении к несуществующему атрибуту
__setattr__(self, name, value)	при установке любого атрибута

__delattr__(self, name)	del obj.name
__getattribute__(self, name)	при обращении к ЛЮБОМУ атрибуту (осторожно, рекурсия!), редко используется
__dir__(self)	dir()
__hasattr__(self, name)	hasattr() (не нужен — __getattr__ обработает)

6. Вызов объекта как функции

__call__(self, *args, **kwargs) # obj()

7. Работа с классами и метаклассами

__init_subclass__(cls, **kwargs)	при создании подкласса
__set_name__(self, owner, name)	при создании дескриптора в классе
__prepare__(name, bases, **kwargs)	метакласс: подготовка пространства имён
__instancecheck__(self, instance)	isinstance() (для метаклассов)
__subclasscheck__(self, subclass)	issubclass() (для метаклассов)

8. Дескрипторы (управление атрибутами другого класса)

__get__(self, instance, owner) # получить атрибут
__set__(self, instance, value) # установить атрибут
__delete__(self, instance)   # удалить атрибут

9. Сериализация

__reduce__(self)	pickle (возвращает (callable, args[, state]))
__reduce_ex__(self, protocol)	расширенная версия для pickle
__getstate__(self)	что сохранять в pickle
__setstate__(self, state)	восстановление из pickle

10. Математические и другие

__complex__(self)	complex()
__int__(self)	int()
__float__(self)	float()

__index__(self)	для преобразования в int (для срезов, bin(), hex())
__trunc__(self)	math.trunc()
__floor__(self)	math.floor()
__ceil__(self)	math.ceil()
__matmul__(self, other)	@ (матричное умножение в Python 3.5+)
__rmatmul__(self, other)	right @
__imatmul__(self, other)	@=

11. Асинхронные методы (async/await)

__await__(self)	await obj
__aiter__(self)	async for
__anext__(self)	async next()
__aenter__(self)	async with
__aexit__(self, exc_type, exc_val, exc_tb)	async with exit

Самые частые методы:
`__init__`, `__str__`, `__repr__` — 90% обычных классов
`__add__`, `__radd__`, `__iadd__` — для своего класса с `+`
`__getitem__`, `__setitem__`, `__len__` — чтобы объект вёл себя как коллекция
`__enter__`, `__exit__` — для `with`
`__call__` — сделать объект вызываемым (как функция)
`__eq__`, `__lt__`, `__hash__` — для сравнения и словарей

Revision #5
Created 7 May 2026 02:50:55 by Admin
Updated 7 May 2026 07:25:46 by Admin