

Урок №3

Объектная модель, введение в ООП

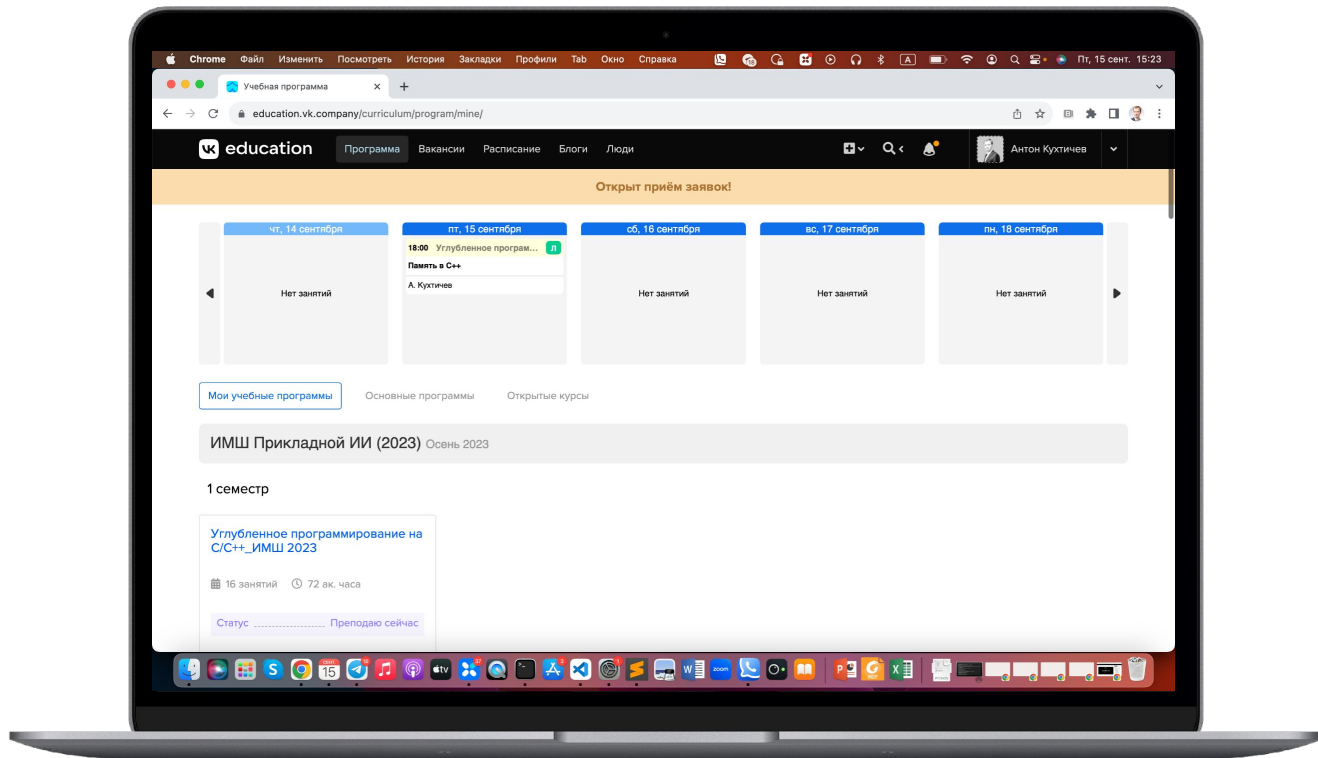
на которой расскажут про классы, свойства, наследование,
магические методы, MRO.

05 октября 2023 года

Антон Кухтичев

Напоминание отметиться на портале

и оставить отзыв
после лекции



Содержание занятия

1. Квиз №2
2. Классы
3. Свойства
4. Магические атрибуты и методы

КВИЗ #2

<https://forms.gle/uezE7aQtNVpjnotb9>

Классы


The background is a solid purple color. It features several white geometric lines: a vertical line on the right side, a large curved line starting from the top left and ending near the bottom right, and two smaller curved lines on the right side that intersect the vertical line.

Классы: атрибуты

```
class Foo:
    name = "cls_name"
    __cls_private = "cls_private"

    def __init__(self, val):
        self.val = val
        self._protected = "protected" # nonpublic
        self.__private = "private"     # _Foo__private

    def print(self):
        print(
            f"{self.val=}, {self._protected=}, {self.__private=}, "
            f"{self.name=}, {self.__cls_private=}"
        )
```



Классы: методы

```
class Foo:
    @staticmethod
    def print_static():
        print("static")

    @classmethod
    def print_cls(cls):
        print(f"class_method for {cls.__name__}")

    def __init__(self, val):
        self.val = val

    def print_offset(self, offset=10):
        print(self.val + offset)

    def __str__(self):
        return f"{self.__class__.__name__}:val={self.val}"
```



Классы

`object.__new__(cls[, ...])`

Статический метод, создает новый экземпляр класса.

После создание экземпляра вызывается (уже у экземпляра) метод `__init__`.

`__init__` ничего не должен возвращать (кроме `None`), иначе - `TypeError`

```
class Singleton:
```

```
    _instance = None
```

```
    def __new__(cls, *args, **kwargs):
```

```
        if cls._instance is None:
```

```
            cls._instance = super().__new__(cls, *args, **kwargs)
```

```
        return cls._instance
```


Классы

`object.__del__(self)`

Деструктор

```
class DBConnector:
    def __init__(self, db_name):
        self.conn = DbDriver(db_name)
    def __del__(self):
        self.conn.close()
        print("DEL")
```

```
db = Connector("users")
```

```
del db # ???
```



Свойства

The background is a solid purple color. It features several white geometric lines: a vertical line on the right side, a large curved line starting from the top left and ending near the bottom right, and two smaller curved lines on the right side that appear to be part of a larger circular or elliptical shape.

Классы: свойства (1)

классический подход

```
class Author:

    def __init__(self, name):
        self.__name = ""
        self.set_name(name)

    def get_name(self):
        return self.__name

    def set_name(self, val):
        self.__name = val
```

pythonic

```
class Author:

    def __init__(self, name):
        self.name = name

    @property
    def name(self):
        return self.__name

    @name.setter
    def name(self, val):
        self.__name = val
```

Классы: свойства (2)

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

    @property
    def name(self):
        """name doc"""
        return self.__name

    @name.setter
    def name(self, val):
        self.__name = val

    @name.deleter
    def name(self):
        del self.__name
```

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

    def get_name(self):
        return self.__name

    def set_name(self, val):
        self.__name = val

    def del_name(self):
        del self.__name

    name = property(
        get_name,
        set_name,
        del_name,
        "name doc",
    )
```

Классы: свойства read/write only

```
class Author:
    def __init__(self, name, password):
        self.__name = name
        self.password_hash = None
        self.password = password

    @property
    def name(self):
        """name is read-only"""
        return self.__name

    @property
    def password(self):
        raise AttributeError("Password is write-only")

    @password.setter
    def password(self, plaintext):
        self.password_hash = make_hash_from_password(plaintext)
```



Классы: доступ к атрибутам

Чтобы найти атрибут экземпляра `obj`, Python обыскивает:

1. Сам объект (`obj.__dict__` и его системные атрибуты)
2. Класс экземпляра (`obj.__class__.__dict__`).
3. Классы, от которых унаследован класс объекта
(`obj.__class__.__mro__`)

Классы: наследование

```
class Timing:
    def __init__(self, start, end):
        self.start = start
        self.end = end

    def duration(self):
        print("Timing.duration")
        return self.end - self.start
```

```
class MinuteTiming(Timing):
    def duration(self):
        print("MinuteTiming.duration")
        seconds = super().duration()
        return seconds / 60
```

```
>>> m = MinuteTiming(1000, 7000)
```

```
>>> m.duration()
```

```
MinuteTiming.duration
```

```
Timing.duration
```

```
100.0
```

Классы: MRO

Порядок разрешения методов (method resolution order) позволяет python выяснить, из какого класса-предка нужно вызывать метод, если он не обнаружен непосредственно в классе-потомке.

```
cls.__mro__
```

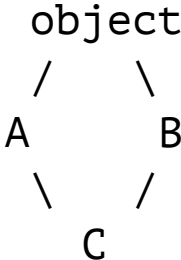
```
cls.mro()
```

```
>>> MinuteTiming.mro()
```

```
[__main__.MinuteTiming, __main__.Timing, object]
```


Классы: локальный порядок старшинства

```
>>> class A:
...     pass
...
>>> class B:
...     pass
...
>>> class C(A, B):
...     pass
...
>>> C.mro()
[<class '__main__.C'>, <class '__main__.A'>, <class '__main__.B'>, <class 'object'>]
>>>
>>> class C(B, A):
...     pass
...
>>> C.mro()
[<class '__main__.C'>, <class '__main__.B'>, <class '__main__.A'>, <class 'object'>]
```



```
graph TD
    object --> A
    object --> B
    A --> C
    B --> C
```

Классы: множественное наследование



Магические атрибуты и методы

Классы: магические атрибуты (1)

Классы

`__name__` — имя класса

`__module__` — модуль, в котором объявлен класс

`__qualname__` — fully qualified имя

`__doc__` — докстринг

`__annotations__` — аннотации статических полей класса

`__dict__` — namespace класса

Методы

`__self__` — объект класса

`__func__` — сама функция, которую мы в классе объявили



Классы: магические атрибуты (2)

Поля, относящиеся к наследованию

`__bases__` — базовые классы

`__base__` — базовый класс, который указан первым по порядку

`__mro__` — список классов, упорядоченный по вызову функции `super`

```
class B(A): pass
```

```
>>> B.__bases__
```

```
(__main__.A,)
```

```
>>> B.__base__
```

```
__main__.A
```

```
>>> B.__mro__
```

```
(__main__.B, __main__.A, object)
```

Классы: магические методы (1)

Доступ к атрибутам

- `__getattr__(self, name)`
- `__getattribute__(self, name)`
- `__setattr__(self, name, val)`
- `__delattr__(self, name)`
- `__dir__(self)`



Классы: магические методы (2)

`object.__call__(self[, args...])`

```
class Adder:  
    def __init__(self, val):  
        self.val = val  
  
    def __call__(self, value):  
        return self.val + value
```

```
a = Adder(10)
```

```
a(5)  # 15
```



Классы: магические методы (3)

To string

`__repr__` — представление объекта. Если возможно, должно быть валидное python выражение для создание такого же объекта

`__str__` — вызывается функциями `str`, `format`, `print`

`__format__` — вызывается при форматировании строки

Классы: магические методы (4)

Сравнение

`object.__lt__(self, other)`

`object.__le__(self, other)`

`object.__eq__(self, other)`

`object.__ne__(self, other)`

`object.__gt__(self, other)`

`object.__ge__(self, other)`

`x < y == x.__lt__(y) # <=, ==, !=, >, >=`

Классы: магические методы (5)

Эмуляция чисел

```
object.__add__(self, other)
object.__sub__(self, other)
object.__mul__(self, other)
object.__matmul__(self, other) (@)
object.__truediv__(self, other)
object.__floordiv__(self, other)
object.__mod__(self, other)
object.__divmod__(self, other)
object.__pow__(self, other[, modulo])
object.__lshift__(self, other)
object.__rshift__(self, other)
object.__and__(self, other)
object.__xor__(self, other)
object.__or__(self, other)
```

Классы: магические методы (б)

Эмуляция чисел

Методы вызываются, когда выполняются операции (+, -, *, @, /, //, %, divmod(), pow(), **, <<, >>, &, ^, |) над объектами

$x + y == x.__add__(y)$

Есть все такие же с префиксом r и i:

`__radd__` — вызывается, если левый операнд не поддерживает `__add__`

`__iadd__` — вызывается, когда $x += y$

Классы: магические методы (7)

Эмуляция контейнеров

`object.__len__(self)`

`object.__length_hint__(self)`

`object.__getitem__(self, key)`

`object.__setitem__(self, key, value)`

`object.__delitem__(self, key)`

`object.__missing__(self, key)`

`object.__iter__(self)`

`object.__next__(self)`

`object.__reversed__(self)`

`object.__contains__(self, item)`

Классы: магические методы (8)

`object.__hash__(self)`

Вызывается функцией `hash()` и коллекциями, которые построены на основе hash-таблиц.

Нужно, чтобы у равных объектов был одинаковый `hash`.

Если определен метод `__eq__` и не определен `__hash__`, то объект не может быть ключом в `hashable` коллекции.

```
>>> key1 = (1, 2, 3)
```

```
>>> key2 = (1, 2, 3, [4, 5])
```

```
>>> s = set()
```

```
>>> s.add(key1)    # ???
```

```
>>> s.add(key2)    # ???
```

Классы: магические методы (9)

`object.__slots__`

Позволяет явно указать поля, которые будут в классе.

В случае указания `__slots__` пропадают поля `__dict__` и `__weakref__`.

Используя `__slots__` можно экономить на памяти и времени доступа к атрибутам объекта.

```
class Point:
```

```
    __slots__ = ("x", "y")
```

```
    def __init__(self, x, y):
```

```
        self.x = x
```

```
        self.y = y
```

Классы: `__init_subclass__`

```
class Timing:
    def __init__(self, start, end):
        self.start = start
        self.end = end

    @classmethod
    def __init_subclass__(cls, **kwargs):
        print("INIT subclass", cls, kwargs)
```

```
class MinuteTiming(Timing):
    def duration(self):
        print("MinuteTiming.duration")
        seconds = super().duration()
        return seconds / 60
```

Домашнее задание

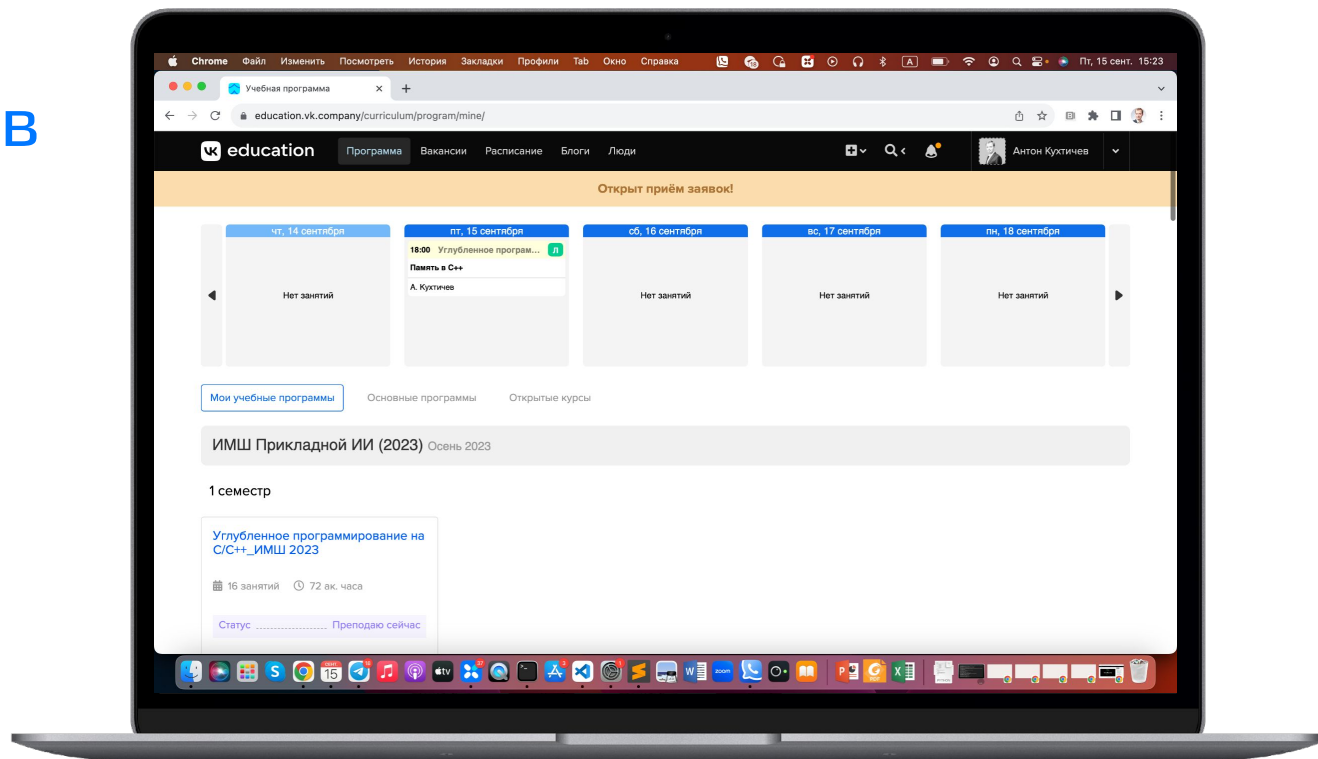
Домашнее задание #03

- Реализовать кастомный список, унаследованный от `list`
- Тесты
- `flake8` + `pylint` перед сдачей



Напоминание оставить отзыв

Это правда важно





Спасибо
за внимание!