ООП Лекция 3

Преподаватель: Тенигин Альберт Андреевич

Сириус
IT-Колледж

Методы класса

- Первый параметр должен быть cls, который можно использовать для доступа к атрибутам класса.
- Метод класса может обращаться только к атрибутам класса, но не к атрибутам экземпляра.
- Метод класса можно вызывать как с использованием ClassName.method_name() , так и с использованием объекта.
- Метод класса может возвращать объект класса.

Сириус
IT-Колледж

Пример работы с методом класса

```
class Hero:

    hero_class_counter = {}

    def __init__(self, name, hero_class):
        self.name = name
        self.hero_class = hero_class
        if hero_class in Hero.hero_class_counter.keys():
            Hero.hero_class_counter[hero_class] += 1
        else:
            Hero.hero_class_counter[hero_class] = 1

    @classmethod
    def get_counter(cls, hero_class):
        return cls.hero_class_counter.get(hero_class)
```

Создание экземпляра (объекта) класса

```python
from datetime import date

class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    @classmethod
    def fromBirthYear(cls, name, year):
        """Создаёт объект Person,
        вычисляя возраст из года рождения."""
        return cls(name, date.today().year - year)

person_1 = Person('Ivan', 21)
person_2 = Person.fromBirthYear('Petya', 1996)
```

Статические методы

- Не может иметь параметр cls или self.
- Статический метод не может получить доступ к атрибутам класса или атрибутам экземпляра.
- Статический метод можно вызывать с помощью ClassName.method_name(), а также с помощью object.method_name().
- Может возвращать объект класса.

Сириус
IT-Колледж

```python
class PhoneNumber:

    @staticmethod
    def isinternational(number: str):
        return number[0] == '+' and len(number) > 11

    @staticmethod
    def isnational(number: str):
        return len(number) == 10

    @staticmethod
    def clean(number: str) -> str:
        return ''.join([i for i in number if i.isdigit() or i == '+'])
```

Сириус
IT-Колледж

Магические методы

- Магические методы в Python — это специальные методы, которые начинаются и заканчиваются двойным подчеркиванием.
- Магические методы не предназначены для непосредственного вызова вами, но вызов происходит внутри класса при определенном действии.
- Например, когда вы добавляете два числа с помощью оператора +, внутри будет вызываться метод __add__().

Сириус
IT-Колледж

Конструкторы и деструкторы	Описание
__new__(cls, other)	Вызывается при создании объекта
__init__(self, other)	Вызывается при создании объекта, но после __new__
__del__(self)	Деструктор

Сириус
IT-Колледж

__new__()

- В таких языках, как Java и C#, оператор new используется для создания нового экземпляра класса.
- В Python магический метод __new__() неявно вызывается перед __init__() методом.
- Метод __new__() возвращает новый объект, который затем инициализируется с помощью __init__().

Сириус
IT-Колледж

```python
class Human:
    def __new__(cls, name):
        print("сработал магический метод __new__")
        return super().__new__(cls)

    def __init__(self, name):
        print("сработал магический метод __init__")
        self.name = name

vanya = Human('Ivan')
```

```python
class Human:
    def __new__(cls, name):
        print("сработал магический метод __new__")
        return object.__new__(cls)

    def __init__(self, name):
        print("сработал магический метод __init__")
        self.name = name

vanya = Human('Ivan')
```

```
сработал магический метод __new__
сработал магический метод __init__
```

__str__()

- Переопределен, чтобы возвращать печатное строковое представление объекта любого определяемого пользователем класса.
- Мы видели str() встроенную функцию, которая возвращает строку из параметра объекта.
- Например, str(12) возвращает строку '12'.

Сириус
IT-Колледж

```python
class Human:
    def __new__(cls, name):
        print("сработал магический метод __new__")
        return super().__new__(cls)

    def __init__(self, name):
        print("сработал магический метод __init__")
        self.name = name

    def __str__(self):
        return 'Human: name={0}'.format(self.name)

vanya = Human('Ivan')
print(str(vanya))
```

__add__()

Вызывается при выполнении сложения двух объектов этого класса.

```python
class Distance:
    def __init__(self, ft=None, inch=None):
        self.ft = ft
        self.inch = inch

    def __add__(self, other):
        temp = Distance()
        temp.ft = self.ft + other.ft
        temp.inch = self.inch + other.inch
        if temp.inch >= 12:
            temp.ft += 1
            temp.inch -= 12
        return temp

    def __str__(self):
        return 'ft: {0} in: {1}'.format(self.ft ,self.inch)
```

```
dst_1 = Distance(10, 11)
dst_2 = Distance(3, 8)
dst_3 = dst_1 + dst_2
print(str(dst_3))
```

```
ft: 14 in: 7
```

__gt__()

- gt означает greater than, как вы знаете из bash.
- Метод вызывается, когда сравниваются два объекта класса знаком >.
- экземпляр_1 > экземпляр_2

```python
class Distance:
    def __init__(self, ft=None, inch=None):
        self.ft = ft
        self.inch = inch

    def __add__(self, other):
        temp = Distance()
        temp.ft = self.ft + other.ft
        temp.inch = self.inch + other.inch
        if temp.inch >= 12:
            temp.ft += 1
            temp.inch -= 12
        return temp

    def __gt__(self, other):
        return self.ft * 12 + self.inch  > other.ft * 12 + other.inch
```

Attribute Magic Methods	Description
__getattr__(self, name)	Is called when the accessing attribute of a class
__setattr__(self, name, value)	Is called when assigning a value to the attribute of a class.
__delattr__(self, name)	Is called when deleting an attribute of a class.

__add__(self, other)	To get called on add operation using + operator
__sub__(self, other)	To get called on subtraction operation using - operator.
__mul__(self, other)	To get called on multiplication operation using * operator.
__floordiv__(self, other)	To get called on floor division operation using // operator.
__truediv__(self, other)	To get called on division operation using / operator.
__mod__(self, other)	To get called on modulo operation using % operator.
__pow__(self, other[, modulo])	To get called on calculating the power using ** operator.
__lt__(self, other)	To get called on comparison using < operator.
__le__(self, other)	To get called on comparison using <= operator.
__eq__(self, other)	To get called on comparison using == operator.
__ne__(self, other)	To get called on comparison using != operator.
__ge__(self, other)	To get called on comparison using >= operator.

String Magic Methods	Description
__str__(self)	To get called by built-int str() method to return a string representation of a type.
__repr__(self)	To get called by built-int repr() method to return a machine readable representation of a type.
__unicode__(self)	To get called by built-int unicode() method to return an unicode string of a type.
__format__(self, formatstr)	To get called by built-int string.format() method to return a new style of string.
__hash__(self)	To get called by built-int hash() method to return an integer.
__nonzero__(self)	To get called by built-int bool() method to return True or False.
__dir__(self)	To get called by built-int dir() method to return a list of attributes of a class.
__sizeof__(self)	To get called by built-int sys.getsizeof() method to return the size of an object.

Type Conversion	Description
__int__(self)	To get called by built-int int() method to convert a type to an int.
__float__(self)	To get called by built-int float() method to convert a type to float.
__complex__(self)	To get called by built-int complex() method to convert a type to complex.
__oct__(self)	To get called by built-int oct() method to convert a type to octal.
__hex__(self)	To get called by built-int hex() method to convert a type to hexadecimal.
__index__(self)	To get called on type conversion to an int when the object is used in a slice expression.
__trunc__(self)	To get called from math.trunc() method.

Unary operators and functions	Description
__pos__(self)	To get called for unary positive e.g. +someobject.
__neg__(self)	To get called for unary negative e.g. -someobject.
__abs__(self)	To get called by built-in abs() function.
__invert__(self)	To get called for inversion using the ~ operator.
__round__(self,n)	To get called by built-in round() function.
__floor__(self)	To get called by built-in math.floor() function.
__ceil__(self)	To get called by built-in math.ceil() function.
__trunc__(self)	To get called by built-in math.trunc() function.

Augmented Assignment	Description
__iadd__(self, other)	To get called on addition with assignment e.g. a +=b.
__isub__(self, other)	To get called on subtraction with assignment e.g. a -=b.
__imul__(self, other)	To get called on multiplication with assignment e.g. a *=b.
__ifloordiv__(self, other)	To get called on integer division with assignment e.g. a //=b.
__idiv__(self, other)	To get called on division with assignment e.g. a /=b.
__itruediv__(self, other)	To get called on true division with assignment
__imod__(self, other)	To get called on modulo with assignment e.g. a%=b.
__ipow__(self, other)	To get called on exponentswith assignment e.g. a **=b.
__ilshift__(self, other)	To get called on left bitwise shift with assignment e.g. a<<=b.
__irshift__(self, other)	To get called on right bitwise shift with assignment e.g. a >>=b.
__iand__(self, other)	To get called on bitwise AND with assignment e.g. a&=b.
__ior__(self, other)	To get called on bitwise OR with assignment e.g. a|=b.
__ixor__(self, other)	To get called on bitwise XOR with assignment e.g. a ^=b.