АКАДЕМИЯ АЙТИ

День 20

academyit.ru

ООП

Функции, Именованные и неименованные аргументы функций, локальные и глобальные переменные, лямбда

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur

ООП

Объектно-ориентированное программирование

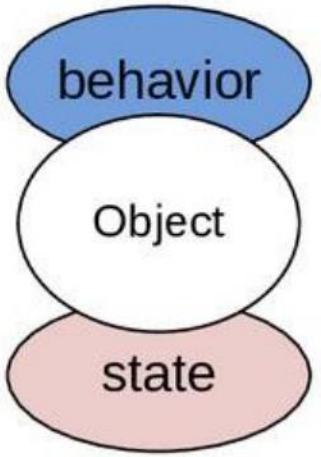
Объектно-ориентированная программа – множество объектов, каждый из которых обладает своими свойствами и поведением, но его внутреннее устройство скрыто от других объектов.

Объекты и классы

Объект – сущность, обладающая *состоянием* и *поведением*

Класс – это множество объектов (экземпляров), имеющих общую структуру и общее поведение

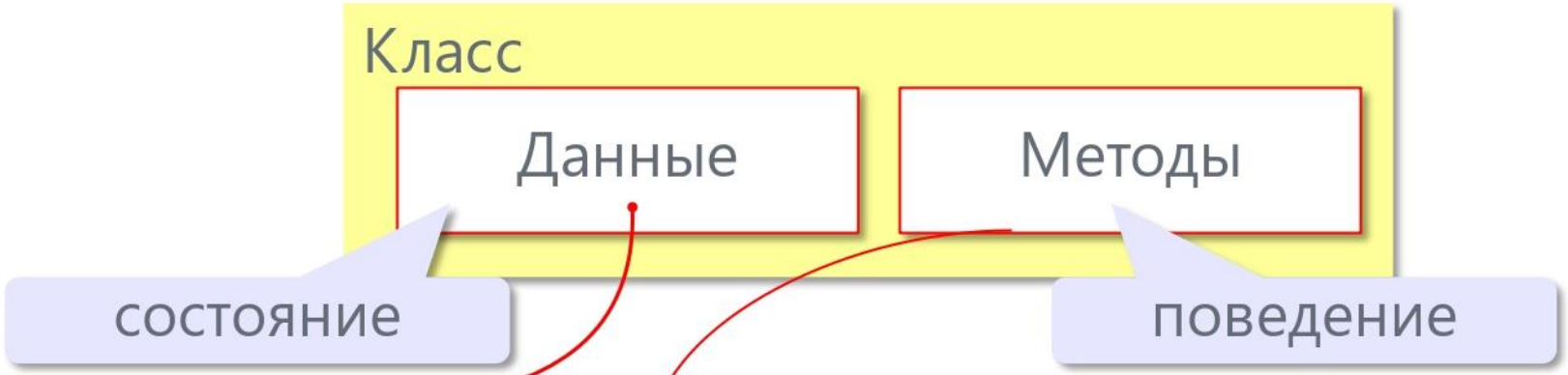
Объекты в реальном мире и программах



Объект реального мира:		Программный объект:
Имя (Название)	⟺	Имя (Идентификатор)
Состояние (state)	⟺	Поля (fields)
Линия поведения (behavior)	⟺	Методы (methods)

Классы в программах

- программа – множество взаимодействующих объектов
- любой объект – экземпляр какого-то класса
- класс – тип данных, состоящий из набора атрибутов (свойств) и методов — функций для работы с этими атрибутами



Поле – это переменная, принадлежащая объекту.

Методы – это функции, определяющие поведение объекта

Создание классов и объектов в Python

- Создание класса

class ИмяКласса:

    определение атрибутов

    определение методов (функций)

Имя класса принято писать с большой буквы!

- Создание объекта

ИмяКласса(аргументы)

Поскольку в программном коде важно не потерять ссылку на только что созданный объект, то обычно его связывают с переменной.

Поэтому создание объекта чаще всего выглядит так:

имя_переменной = ИмяКласса(аргументы)

Пример класса Point

```
class Point:
    """Базовый класс Точка"""
    count_Point = 0 #количество объектов класса
        def __init__(self,x,y):  #конструктор класса
        self.x = x  #инициализируем атрибут x класса
        self.y = y  #инициализируем атрибут y класса
        Point.count_Point +=1  #увеличиваем статическое
поле


pt1 = Point(10,20) # Создадим одну точку
print("Создано всего точек: %d " % Point.count_Point)
pt2 = Point(40,50) # Создадим еще одну точку
print("Создано всего точек: %d " % Point.count_Point)
```

Класс Point

- Переменная `count_Point` — переменная класса, значение которой разделяется между экземплярами этого класса
- Метод `__init__()` — конструктор класса, в котором атрибутам (полям) класса задаются значения
- Другие методы объявляются как обычные функции, за исключением того, что первый аргумент для каждого метода должен быть `self`

Конструктор класса

- Специальный метод класса, который всегда называется __init__ и используется для задания значений атрибутов объектов
- В классе может быть только один конструктор
- Если в классе нет конструктора, то то Python по умолчанию считает его унаследованным от базового класса

Атрибуты класса, атрибуты объекта

Атрибуты класса определяются вне метода, их значение общее для всех экземпляров класса

Атрибуты объекта объявляются для `self` внутри любого метода

Чтобы посмотреть на все атрибуты и методы класса, используйте встроенную функцию dir()

Функции для доступа к атрибутам

getattr(obj, name [, default]) — для доступа к атрибуту объекта.
hasattr(obj, name) — проверить, есть ли в obj атрибут name.
setattr(obj, name, value) — задать атрибут. Если атрибут не существует, он будет создан.
delattr(obj, name) — удалить атрибут

Пример

```
hasattr(pt1, 'x') # атрибут 'x' существует?
getattr(pt1, 'y') # возвращает атрибут 'y'
setattr(pt1, 'x', 22) #устанавливает
атрибуту 'x' значение 22
delattr(pt1, 'x') # удаляет атрибут 'x’
```

Инкапсуляция

- В ООП значения атрибутов обычно скрываются от других объектов и доступны только с использованием методов
- В этом состоит принцип **инкапсуляции**
- В Python для скрытия атрибута или метода перед его именем ставят два подчеркивания __

Пример: скрытый метод класса

```
def __checkValue(x):
    if isinstance(x, int) or isinstance(x, float):
        return True
    return False

def __init__(self,x = 0,y = 0):
        if Point.__checkValue(x) \
                and Point.__checkValue(y):
            self.__x = x
            self.__y = y
            Point.count_Point += 1
        else:
            print("Координаты должны быть числами"
            self.__x = 0
            self.__y = 0
```

Статические методы

Статические методы (с декоратором @staticmethod) работают с атрибутами класса

```
@staticmethod
def get_count_Point():
    print(f"Всего точек: {Point.count_Point}")
```

Методы экземпляра класса

- Методы экземпляра класса принимают объект класса как первый аргумент, который принято называть self и который указывает на сам экземпляр. Количество параметров метода не ограничено.
- Используя параметр self, мы можем менять состояние объекта и обращаться к другим его методам и параметрам. К тому же, используя атрибут self.__class__ , мы получаем доступ к атрибутам класса и возможности менять состояние самого класса.

Примеры методов

```
def set_coords(self, x, y):
    if Point.__checkValue(x) and Point.__checkValue(y):
        self.__x = x
        self.__y = y
    else:
        print("Координаты должны быть числами")

def get_coords(self):
        return self.__x, self.__y

def print_point(self):
        print(self.get_coords())
```

Сборка мусора

- Python автоматически удаляет ненужные объекты, чтобы освободить пространство памяти
- Когда объект присваивают новой переменной или добавляют в контейнер (список, кортеж, словарь), количество ссылок объекта увеличивается.
- Количество ссылок на объект уменьшается, когда он удаляется с помощью del, или его ссылка выходит за пределы видимости.
- Когда количество ссылок достигает нуля, Python автоматически освобождает память.

Переопределение операций в классах Python

- Переопределение операций в Python – это возможность с помощью специальных методов в классах переопределять различные операции языка.
- Имена таких методов включают двойное подчеркивание спереди и сзади.
- Например, когда мы используем оператор +, автоматически вызывается метод __add__
- В Питоне можно переопределить арифметические, логические операции, сравнение, вызов функции, обращение по индексу, вывод объекта и т.д.

Соответствие операций и методов

Оператор	Метод
+	__add__(self, other)
-	__sub__(self, other)
*	__mul__(self, other)
/	__truediv__(self, other)
//	__floordiv__(self, other)
%	__mod__(self, other)
**	__pow__(self, other)

Соответствие операций и методов

Оператор	Метод
<	__lt__(self, other)
>	__gt__(self, other)
<=	__le__(self, other)
>=	__ge__(self, other)
==	__eq__(self, other)
!=	__ne__(self, other)

Соответствие операций и методов

Оператор	Метод
-=	__isub__(self, other)
+=	__iadd__(self, other)
*=	__imul__(self, other)
/=	__idiv__(self, other)
//=	__ifloordiv__(self, other)
%=	__imod__(self, other)
**=	__ipow__(self, other)

Примеры перегрузки операций

```
def __iadd__(self, other):
    if Point.__checkValue(other):
        self.__x += other
        self.__y += other
    return self

def __str__(self):
     return f'{self._x}, {self.__y}'

def __getitem__(self, key):
     if key ==0: return __x
     elif key == 1: return __y
     else return None
```

Пример – класс Time (время)

```
import functools
# на основе < и == делает все сравнения
@functools.total_ordering
class Time:
    """ Время в часах и минутах """
    def __init__(self, h=0, m=0):
        total = h * 60 + m
        self.h = total // 60 % 24
        self.m = total % 60

    def __str__(self):
        return f"{self.h:02}:{self.m:02}"

    def __add__(self, other):  # +
        return Time(self.h + other.h, self.m + other.m)

    def __mul__(self, other):  # t * x
        if isinstance(other, int):
            return Time(self.h * other, self.m * other)

    def __rmul__(self, other):  # x * t
        return self * other

    def __sub__(self, other):  # a - b
        return self + (other * -1)

    def __neg__(self):  # -b
        return self * -1

    def __int__(self):
        return self.h * 60 + self.m

    def __lt__(self, other):  # <
        return int(self) < int(other)

    def __eq__(self, other):  # ==
        return int(self) == int(other)
```

Задачи

Задача 1.

Создайте класс Soda (для определения типа газированной воды), принимающий 1 аргумент при инициализации (отвечающий за добавку к выбираемому лимонаду).

В этом классе реализуйте метод show_my_drink(), выводящий на печать «Газировка и {ДОБАВКА}» в случае наличия добавки, а иначе отобразится следующая фраза: «Обычная газировка».

Задача 2.

Николаю требуется проверить, возможно ли из представленных отрезков условной длины сформировать треугольник.
Для этого он решил создать класс TriangleChecker, принимающий только положительные числа.

С помощью метода is_triangle() возвращаются следующие значения (в зависимости от ситуации):

– Ура, можно построить треугольник!;
– С отрицательными числами ничего не выйдет!;
– Нужно вводить только числа!;
– Жаль, но из этого треугольник не сделать.

Домашнее задание

Задание 1.

Преобразуйте класс Time, написанный на вебинаре, в класс DateTime. Он должен хранить в себе информацию о дне, часах и минутах. Дни могут быть неограниченно большими, а также отрицательными.
Измените методы для сложения и вычитания так, чтобы они корректно складывали время и дни.

Дополнительно: Добавьте месяцы и года, учтите високосный год, добавьте как можно больше методов перегрузки.

АПЛАНА
ГРУППА КОМПАНИЙ

Входит в ГК Аплана

АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning и очное обучение

Направления обучения:

- Информационные технологии
- Информационная безопасность
- ИТ-менеджмент и управление проектами
- Разработка и тестирование ПО
- Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск, Хабаровск, Красноярск, Тюмень, Нижний Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис в Москве

Разработка программного обеспечения и информационных систем

Ежегодные награды Microsoft, Huawei, Cisco и другие

Программы по импортозамещению

Ресурсы более 400 высококлассных экспертов и преподавателей

Сеть региональных учебных центров по всей России

Крупные заказчики

СБЕР

РЖД

Билайн

TATNEFT

НОРНИКЕЛЬ

100+
сотрудников

АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:
Москва, Варшавское шоссе 47, корп. 4, 7 этаж
Тел: +7 (495) 150-96-00
academy@it.ru
academyit.ru