Введение в объектно-ориентированное программирование, понятие классов и объектов

№ урока:	1	**Курс:**	Python Essential

Средства обучения:	Python 3; интегрированная среда разработки (PyCharm или Microsoft Visual Studio + Python Tools for Visual Studio)

Обзор, цель и назначение урока

После завершения урока обучающиеся будут иметь представление о парадигме объектно-ориентированного программирования, смогут создавать классы и объекты в программах на Python.

Изучив материал данного занятия, учащийся сможет:

- Понимать основы парадигмы объектно-ориентированного программирования
- Понимать назначение и различие между классами и объектами
- Иметь представление об инкапсуляции
- Создавать классы в программах на Python
- Создавать экземпляры классов в программах на Python
- Использовать так называемые «магические методы» для задания особого поведения и переопределения операторов

Содержание урока

1. Понятие ООП
2. Создание классов
3. Создание экземпляров классов
4. Инкапсуляция
5. Конструкторы и «магические» методы в Python

Резюме

- Парадигма программирования — это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию). Это способ концептуализации, определяющий организацию вычислений и структурирование работы, выполняемой компьютером.
- Важно отметить, что парадигма программирования не определяется однозначно языком программирования; практически все современные языки программирования в той или иной мере допускают использование различных парадигм (мультипарадигмальное программирование).
- Python является мультипарадигменным языком, он поддерживает множество различных парадигм как на уровне языка (императивное программирование, процедурное программирование, структурное программирование, объектно-ориентированное программирование, функциональное программирование), так и при помощи разного рода библиотек и фреймворков (например, для аспектно-ориентированного программирования).
- Императивное программирование — это парадигма программирования, которая, в отличие от декларативного программирования, описывает процесс вычисления в виде инструкций, изменяющих состояние данных. Императивная программа очень похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть это последовательность команд, которые должен выполнить компьютер.
- Процедурное программирование — программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка. Процедурное

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

программирование является отражением архитектуры традиционных ЭВМ, которая была предложена фон Нейманом в 1940-х годах. Теоретической моделью процедурного программирования служит абстрактная вычислительная система под названием машина Тьюринга.

- Структурное программирование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 1970-х годах Э. Дейкстрой и др. В соответствии с данной методологией любая программа строится без использования оператора goto из трёх базовых управляющих структур: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово, методом «сверху вниз».
- Объектно-ориентированное программирование (ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.
- ООП возникло в результате развития идеологии процедурного программирования, где данные и подпрограммы (процедуры, функции) их обработки формально не связаны.
- Первым языком программирования, в котором были предложены принципы объектной ориентированности, была Симула. В момент его появления в 1967 году в нём были предложены революционные идеи: объекты, классы, виртуальные методы и др., однако это всё не было воспринято современниками как нечто грандиозное. Тем не менее, большинство концепций были развиты Аланом Кэем и Дэном Ингаллсом в языке Smalltalk. Именно он стал первым широко распространённым объектно-ориентированным языком программирования.
- Основные понятия ООП:
    - Абстракция
    - Инкапсуляция
    - Наследование
    - Полиморфизм
    - Класс
    - Объект
- Абстрагирование — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция — это набор всех таких характеристик.
- Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе, и скрыть детали реализации от пользователя.
- Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.
- Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта. При использовании термина «полиморфизм» в сообществе ООП подразумевается полиморфизм подтипов; а использование параметрического полиморфизма называют обобщённым программированием.
- Класс является описываемой на языке терминологии исходного кода моделью ещё не существующей сущности (объекта). Фактически он описывает устройство объекта, являясь своего рода чертежом. Объект — это экземпляр класса. При этом в некоторых исполняющих системах класс также может представляться некоторым объектом при выполнении программы посредством динамической идентификации типа данных. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.
- Объект – сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса или копирования прототипа.
- Каждый объект является значением, относящимся к определённому классу. Класс представляет собой объявленный программистом составной тип данных, имеющий в составе:
    - Поля данных – параметры объекта (конечно, не все, а только необходимые в программе), задающие его состояние (свойства объекта предметной области). Иногда поля данных объекта называют свойствами объекта, из-за чего возможна путаница. Физически поля представляют собой значения (переменные, константы), объявленные как принадлежащие классу.

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

    - Методы – процедуры и функции, связанные с классом. Они определяют действия, которые можно выполнять над объектом такого типа, и которые сам объект может выполнять.
- Инкапсуляция обеспечивается следующими средствами
    - Контроль доступа
        - Поскольку методы класса могут быть как чисто внутренними, обеспечивающими логику функционирования объекта, так и внешними, с помощью которых взаимодействуют объекты, необходимо обеспечить скрытость первых при доступности извне вторых. Для этого в языки вводятся специальные синтаксические конструкции, явно задающие область видимости каждого члена класса. Традиционно это модификаторы public, protected и private, обозначающие, соответственно, открытые члены класса, члены класса, доступные только из классов-потомков, и скрытые, доступные только внутри класса. Конкретная номенклатура модификаторов и их точный смысл различаются в разных языках.
        - В Python есть два уровня доступа, соответствующие традиционным модификаторам public и private.
    - Методы доступа
        - Поля класса в общем случае не должны быть доступны извне, поскольку такой доступ позволил бы произвольным образом менять внутреннее состояние объектов. Поэтому поля обычно объявляются скрытыми (либо язык в принципе не позволяет обращаться к полям класса извне), а для доступа к находящимся в полях данным используются специальные методы, называемые методами доступа. Такие методы либо возвращают значение того или иного поля, либо производят запись в это поле нового значения. При записи метод доступа может проконтролировать допустимость записываемого значения и, при необходимости, произвести другие манипуляции с данными объекта, чтобы они остались корректными (внутренне согласованными). Методы доступа называют ещё аксессорами (от англ. access — доступ), а по отдельности — геттерами (англ. get — чтение) и сеттерами (англ. set — запись).
    - Свойства объекта
        - Псевдополя, доступные для чтения и/или записи. Свойства внешне выглядят как поля и используются аналогично доступным полям (с некоторыми исключениями), однако фактически при обращении к ним происходит вызов методов доступа. Таким образом, свойства можно рассматривать как «умные» поля данных, сопровождающие доступ к внутренним данным объекта какими-либо дополнительными действиями (например, когда изменение координаты объекта сопровождается его перерисовкой на новом месте).
- В терминологии Python члены класса называются атрибутами.
- Классы определяются при помощи ключевого слова class.
- Внутри определения класса, как правило, находятся определения атрибутов класса, но разрешены любые операторы.
- В Python всё является объектами – экземплярами каких-либо классов, даже сами классы, которые являются объектами – экземплярами метаклассов. Главным метаклассом является класс type, который является абстракцией понятия типа данных.
- Классы как объекты поддерживают два вида операций: обращение к атрибутам классов и создание (инстанцирование) объектов – экземпляров класса (instance objects).
- Обращение к атрибутам какого-либо класса или объекта производится путём указания имени объекта и имени атрибута через точку.
- Для создания экземпляров класса используется синтаксис вызова функции.
- Единственная доступная операция для объектов-экземпляров – это доступ к их атрибутам.
- Атрибуты объектов-экземпляров делятся на два типа: атрибуты-данные и методы.
- Атрибуты-данные аналогичны полям в терминологии большинства широко распространённых языков программирования.
- Атрибуты-данные не нужно описывать: как и переменные, они создаются в момент первого присваивания. Как правило, их создают в методе-конструкторе __init__.

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

- Метод – это функция, принадлежащая объекту. Все атрибуты класса, являющиеся функциями, описывают соответствующие методы его экземпляров, однако они не являются одним и тем же.
- Особенностью методов является то, что в качестве первого аргумента им передаётся данный экземпляр класса. Таким образом, если obj – экземпляр класса MyClass, вызов метода obj.method() эквивалентен вызову функции MyClass.method(obj). Соответствующий формальный параметр принято называть self.
- В Python 2 атрибуты-функции класса называются непривязанными методами (unbound methods), а методы экземпляров класса – привязанными методами (bound methods). Кроме того, первый позиционный аргумент (self) обязан быть экземпляром данного класса, тогда как в Python 3 он может быть чем угодно.
- Атрибуты класса являются общими для самого класса и всех его экземпляров. Их изменение отображается на все соответствующие объекты. Атрибуты-данные принадлежат конкретному экземпляру и их изменение никак не влияет на соответствующие атрибуты других экземпляров данного класса. Таким образом, атрибуты класса, которые не являются функциями, примерно соответствуют статическим полям в других языках программирования, а атрибуты-данные – обычным полям.
- Декоратор – это специальная функция, которая изменяет поведение функции или класса. Для применения декоратора следует перед соответствующим объявлением указать символ @, имя необходимого декоратора и список его аргументов в круглых скобках. Если передача параметров декоратору не требуется, скобки не указываются.
- Для создания статических методов используется декоратор staticmethod.
- Так как классы тоже являются объектами, то помимо атрибутов-функций они могут иметь и собственные методы. Для создания методов класса используется декоратор classmethod. В таких методах первый параметр принято называть не self, а cls. Методы класса обычно используются в двух случаях:
    - для создания фабричных методов, которые создают экземпляры данного класса альтернативными способами;
    - статические методы, вызывающие статические методы: поскольку данный класс передаётся как первый аргумент функции, не нужно вручную указывать имя класса для вызова статического метода.
- Все атрибуты по умолчанию являются публичными.
- Атрибуты, имена которых начинаются с одного знака подчёркивания (_) говорят программисту о том, что они относятся ко внутренней реализации класса и не должны использоваться извне, однако никак не защищены.
- Атрибуты, имена которых начинаются, но не заканчиваются, двумя символами подчёркивания, считаются приватными. К ним применяется механизм «name mangling», суть которого заключается в том, что изнутри класса и его экземпляров к этим атрибутам можно обращаться по тому имени, которое было задано при объявлении, однако на самом деле к именам слева добавляется подчёркивание и имя класса. Этот механизм не предполагает защиты данных от изменения извне, так как к ним всё равно можно обратиться, зная имя класса и то, как Python изменяет имена приватных атрибутов, однако позволяет защитить их от случайного переопределения в классах-потомках.
- Атрибуты, имена которых начинаются и заканчиваются двумя знаками подчёркивания, являются внутренними для Python и задают особые свойства объектов. С одним из подобных атрибутов мы уже имели дело ранее (документационная строка __doc__). Другим примером может служить атрибут __class__, в котором хранится класс данного объекта.
- Среди таких атрибутов есть методы. В документации Python подобные методы называются методами со специальными именами, однако в сообществе Python-разработчиков очень распространено название «магические методы». Также встречается и название «специальные методы». Они задают особое поведение объектов и позволяют переопределять поведение встроенных функций и операторов для экземпляров данного класса.
- Наиболее часто используемым из специальных методов является метод __init__, который автоматически вызывается после создания экземпляра и соответствует конструкторам в других языках программирования.
- Некоторые специальные методы:
    - __new__(cls[, ...]) – вызывается для создания нового экземпляра класса

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

- __init__(self[, ...]) – вызывается после создания нового экземпляра класса; аргументы этого метода совпадают с параметрами, которые указываются при инстанцировании класса;
- __del__(self) – деструктор – вызывается перед удалением объекта;
- __repr__(self) – возвращает строковое представление объекта, которое, если это возможно, должно быть корректным выражением, создающим аналогичный объект, иначе содержать его описание; вызывается функцией repr;
- __str__(self) – возвращает предназначенное для человека строковое представление объекта; вызывается функциями str, print и format;
- __bytes__(self) – вызывается функцией bytes() для создания байтовой строки;
- __format__(self, format_spec) – вызывается функцией format для получения форматированной строки согласно параметрам format_spec;
- __lt__(self, other), __le__(self, other), __eq__(self, other), __ne__(self, other), __gt__(self, other), __ge__(self, other) – определяют операции сравнения <, <=, ==, !=, >, >=;
- __hash__(self) – вызывается функцией hash() для получения числа – хеш-значения объекта;
- __bool__(self) (в Python 2 – __nonzero__(self)) – вызывается функцией bool() и при использовании объекта в условиях;
- __len__(self) – вызывается функцией len();
- __getattribute__(self, name) – вызывается для получения доступа к атрибутам экземпляров класса, может (вместе с __setattr__) использовать для реализации полноценной инкапсуляции;
- __getattr__(self, name) – в отличие от__getattribute__, вызывается только когда атрибут не найден; если класс реализует __getattribute__, вызывается только если __getattribute__ вызовет его явно или сгенерирует исключение AttributeError;
- __setattr__(self, name, value) – используется для присваивания значений атрибутам;
- __delattr__(self, name) – вызывается при удалении атрибутов.

Закрепление материала

- Что такое парадигма программирования?
- Назовите основные принципы объектно-ориентированного программирования.
- Что такое класс?
- Что такое объект?
- Что такое инкапсуляция?
- Что в Python не является объектом?
- Что такое атрибуты класса?
- Что такое атрибуты экземпляров класса?
- Чем отличаются обычные связанные методы, статические и методы класса?
- Что такое специальные («магические») методы?

Дополнительное задание

Задание
Создайте класс, описывающий автомобиль. Создайте класс автосалона, содержащий в себе список автомобилей, доступных для продажи, и функцию продажи заданного автомобиля.

Самостоятельная деятельность учащегося

Задание 1
Создайте класс, описывающий книгу. Он должен содержать информацию об авторе, названии, годе издания и жанре. Создайте несколько разных книг. Определите для него операции проверки на равенство и неравенство, методы __repr__ и __str__.

Задание 2

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

Создайте класс, описывающий отзыв к книге. Добавьте в класс книги поле – список отзывов. Сделайте так, что при выводе книги на экран при помощи функции print также будут выводиться отзывы к ней.

Задание 3
Используя ссылки в конце данного урока, ознакомьтесь с таким средством инкапсуляции как свойства. Ознакомьтесь с декоратором property в Python. Создайте класс, описывающий температуру и позволяющий задавать и получать температуру по шкале Цельсия и Фаренгейта, причём данные могут быть заданы в одной шкале, а получены в другой.

Задание 4
Ознакомьтесь со специальными методами в Python, используя ссылки в конце урока, и научитесь использовать те из них, назначение которых вы можете понять. Возвращайтесь к этой теме на протяжении всего курса и изучайте специальные методы, соответствующие темам каждого урока.

Рекомендуемые ресурсы

Документация по Python
https://docs.python.org/3/tutorial/classes.html – ООП в Python
https://docs.python.org/3/reference/datamodel.html#special-method-names – методы со специальными именами

Обзор специальных методов в Python
http://rafekettler.com/magicmethods.html

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке
https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование
https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование_на_Python
https://ru.wikipedia.org/wiki/Класс_(программирование)
https://ru.wikipedia.org/wiki/Объект_(программирование)
https://ru.wikipedia.org/wiki/Инкапсуляция_(программирование)

Информация для задания №3
https://ru.wikipedia.org/wiki/Свойство_(программирование)
http://www.programiz.com/python-programming/property
https://docs.python.org/3/library/functions.html#property

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com