Наследование и полиморфизм

№ урока:	2	**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 расширят своё представление о парадигме объектно-ориентированного программирования и её реализации в языке Python, смогут понимать и использовать принципы наследования и полиморфизма.

Изучив материал данного занятия, учащийся сможет:

- Понимать механизмы наследования и множественного наследования
- Иметь представление о полиморфизме
- Создавать иерархии классов в программах на Python
- Понимать особенности полиморфизма в динамически типизированных языках и использовать так называемый принцип «утиной типизации»

Содержание урока

1. Наследование и полиморфизм
2. Множественное наследование
3. Порядок разрешения методов, линеаризация класса
4. Утиная типизация

Резюме

- *Наследование* — механизм языка, позволяющий описать новый класс на основе уже существующего (родительского, базового) класса. Класс-потомок может добавить собственные методы и свойства, а также пользоваться родительскими методами и свойствами. Позволяет строить иерархии классов. Является одним из пяти типов ассоциации. Является одним из основных принципов объектно-ориентированного программирования.
- Наследование обеспечивает в ООП полиморфизм и абстракцию данных.
- Класс, от которого произошло наследование, называется *базовым*, *суперклассом* или *родительским* (англ. base class, superclass, parent class). Классы, которые произошли от базового, называются *потомками*, *наследниками*, *дочерними*, *производными классами* или *подклассами* (англ. child class, heir class, derived class, subclass).
- В ряде языков программирования все классы явно или неявно наследуются от некого базового класса. Smalltalk был одним из первых языков, в которых использовалась эта концепция. К таким языкам относятся Objective-C (NSObject), Perl (UNIVERSAL), Eiffel (ANY), Java (java.lang.Object), C# (System.Object), Delphi (TObject).
- В Python таким базовым классом является класс object.
- В версиях до 2.2 некоторые объектно-ориентированные возможности Python были заметно ограничены. Например, было невозможно наследовать встроенные классы и классы из модулей расширения. Свойства (property) не выделялись явно. Начиная с версии 2.2, объектная система Python была существенно переработана и дополнена.
- В целях совместимости с существующим кодом в Python 2 существуют две системы типов: классы нового типа (new-style classes) и классы старого типа (old-style classes, classic classes).
- Поддержка классов старого типа нужна только для поддержки кода, написанного для старых версий Python. В новом коде следует использовать классы нового типа.

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

- Для создания класса нового типа следует унаследовать его от любого другого класса нового типа. Все стандартные классы являются классами нового типа. Базовым из них является класс object.
- Если не указывать базовый класс или унаследовать его от другого класса старого типа, то данный класс является классом старого типа.
- В Python 3 классов старого типа не существует. Все классы являются классами нового типа в терминологии Python 2. Все классы по умолчанию наследуются от класса object, даже если это явно не указывать.
- При множественном наследовании у класса может быть более одного предка. В этом случае класс наследует методы всех предков. Достоинство такого подхода в большей гибкости, однако он может быть потенциальным источником ошибок.
- При объявлении класса в Python список базовых классов указывается через запятую в круглых скобках после имени данного класса.
- Если в данном классе метод или атрибут был переопределён, а требуется доступ к соответствующему атрибуту суперкласса, это можно совершить двумя способами: путём явного обращения к атрибуту необходимого класса и, при необходимости, передачи параметра self, соответствующего экземпляру данного класса, либо при помощи инстанцирования специального класса super (выглядит, как вызов функции).
- В Python 2 как параметры конструктора super передаются имя текущего класса и ссылка на экземпляр текущего класса (self). В Python 3 можно не указывать ничего и данные параметры будут получены автоматически.
- super позволяет получить доступ к предыдущему классу в цепочке MRO (method resolution order ¬– порядок разрешения методов), которая в случае простого наследования выглядит так же, как и цепь наследования, а в случае множественного наследования строится интерпретатором при помощи алгоритма C3-линеаризации, который позволяет построить устойчивый список из самого класса и всех его предков (родителей и пророди телей) в котором по порядку слева направо будет производиться поиск метода (линеаризация класса), такой что в линеаризации класса-потомка соблюдается тот же порядок следования классов-пророди телей, что и в линеаризации класса-родителя (свойство монотонности), и в линеаризации класса-потомка соблюдается тот же порядок следования классов-родителей, что и в его объявлении (свойство локального старшинства). Это позволяет решить многие проблемы множественного наследования, такие как проблему ромбовидной иерархии классов, присущие другим языкам, которые поддерживают множественное наследование, но не определяющих порядок разрешения методов, например, C++.
- Если вызываемый метод отсутствует в текущем классе, то он автоматически ищется среди базовых классов при помощи MRO.
- Цепь MRO можно изменить путём переопределения специального атрибута __mro__. Также в метаклассе type определён метод mro, который возвращает значение данного атрибута.
- MRO и super доступны только для классов нового типа.
- В Python есть две встроенные функции, которые работают с наследованием:
- isinstance(obj, cls) проверяет, является ли obj экземпляром класса cls или класса, который является наследником класса cls;
- issubclass(cls, base) проверяет, является ли класс cls наследником класса base.
- В языках программирования и теории типов полиморфизмом называется способность функции обрабатывать данные разных типов.
- Существует несколько видов полиморфизма. Два наиболее различных из них были описаны Кристофером Стрэчи в 1967 году: это специальный полиморфизм (или «ad hoc полиморфизм») и параметрический полиморфизм. Кратко специальный полиморфизм описывается принципом «много реализаций с похожими интерфейсами», а параметрический полиморфизм — «одна реализация с обобщённым интерфейсом».
- *Полиморфизм* является фундаментальным свойством системы типов. Различают статическую неполиморфную типизацию (потомки Алгола и BCPL), динамическую типизацию (потомки Lisp, Smalltalk, APL; Python) и статическую полиморфную типизацию (потомки ML). Параметрический полиморфизм и динамическая типизация намного существеннее, чем специальный полиморфизм, повышают коэффициент повторного использования кода, поскольку определенная единственный раз функция реализует без дублирования заданное поведение

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

для бесконечного множества вновь определяемых типов, удовлетворяющих требуемым в функции условиям. С другой стороны, временами возникает необходимость обеспечить различное поведение функции в зависимости от типа параметра, и тогда необходимым оказывается специальный полиморфизм.

- Параметрический полиморфизм повсеместно используется в функциональном программировании, где он обычно обозначается просто как «полиморфизм». В сообществе же объектно-ориентированного программирования под термином «полиморфизм» обычно подразумевают ad hoc полиморфизм методов классов, связанных в иерархию наследования; а использование параметрического полиморфизма называют обобщённым программированием, или иногда «статическим полиморфизмом».
- В объектно-ориентированном программировании полиморфизм подтипов (или полиморфизм включения) представляет собой концепцию в теории типов, предполагающую использование единого имени (идентификатора) при обращении к объектам нескольких разных классов, при условии, что все они являются подклассами одного общего надкласса (суперкласса). Полиморфизм подтипов состоит в том, что несколько типов формируют подмножество другого типа (их базового класса) и потому могут использоваться через общий интерфейс.
- Объектно-ориентированные языки программирования реализуют полиморфизм подтипов посредством наследования, то есть определения подклассов.
- Неявная типизация, латентная типизация или утиная типизация (англ. Duck typing) — вид динамической типизации, применяемой в некоторых языках программирования (Perl, Smalltalk, Python, Objective-C, Ruby, JavaScript, Groovy, ColdFusion, Boo, Lua, Go, C#), когда границы использования объекта определяются его текущим набором методов и свойств, в противоположность наследованию от определённого класса. То есть считается, что объект реализует интерфейс, если он содержит все методы этого интерфейса, независимо от связей в иерархии наследования и принадлежности к какому-либо конкретному классу.
- Название термина пошло от английского «duck test» («утиный тест»), который в оригинале звучит как:
  «If it looks like a duck, swims like a duck and quacks like a duck, then it probably is a duck».
  («Если это выглядит как утка, плавает как утка и крякает как утка, то вероятно это утка».).
- Утиная типизация решает такие проблемы иерархической типизации, как:
  - невозможность явно указать (путём наследования) на совместимость интерфейса со всеми настоящими и будущими интерфейсами, с которыми он идейно совместим;
  - экспоненциальное увеличение числа связей в иерархии типов при хотя бы частичной попытке это сделать.
- Другим близким подходом является структурные подтипы в OCaml, где типы объектов совместимы, если совместимы сигнатуры их методов, независимо от объявленного наследования, причём всё это проверяется во время компиляции программы.

Закрепление материала

- Что такое наследование?
- Что такое множественное наследование?
- Каким образом в Python указать, что один класс наследуется от другого класса? От нескольких других классов?
- Какие виды классов существуют в Python 2? В Python 3?
- Что такое MRO?
- Каким образом получить доступ к методу базового класса, если он был переопределён в данном?
- Что такое полиморфизм?
- Что такое утиная типизация?

Дополнительное задание

Задание

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 иерархию классов транспортных средств. В общем классе опишите общие для всех транспортных средств поля, в наследниках – специфичные для них. Создайте несколько экземпляров. Выведите информацию о каждом транспортном средстве.

Самостоятельная деятельность учащегося

Задание 1
Создайте класс Editor, который содержит методы view_document и edit_document. Пусть метод edit_document выводит на экран информацию о том, что редактирование документов недоступно для бесплатной версии. Создайте подкласс ProEditor, в котором данный метод будет переопределён. Введите с клавиатуры лицензионный ключ и, если он корректный, создайте экземпляр класса ProEditor, иначе Editor. Вызовите методы просмотра и редактирования документов.

Задание 2
Опишите классы графического объекта, прямоугольника и объекта, который может обрабатывать нажатия мыши. Опишите класс кнопки. Создайте объект кнопки и обычного прямоугольника. Вызовите метод нажатия на кнопку.

Задание 3
Создайте иерархию классов с использованием множественного наследования. Выведите на экран порядок разрешения методов для каждого из классов. Объясните, почему линеаризации данных классов выглядят именно так.

Рекомендуемые ресурсы

Документация Python
https://docs.python.org/3/tutorial/classes.html#inheritance
https://docs.python.org/3/tutorial/classes.html#multiple-inheritance
https://www.python.org/download/releases/2.3/mro/

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке
https://ru.wikipedia.org/wiki/Наследование_(программирование)
https://ru.wikipedia.org/wiki/Множественное_наследование
https://ru.wikipedia.org/wiki/C3-линеаризация
https://ru.wikipedia.org/wiki/Полиморфизм_(информатика)
https://ru.wikipedia.org/wiki/Утиная_типизация

Статья о порядке разрешения методов в Python
http://habrahabr.ru/post/62203/

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