Объектно-ориентированное программирование (ООП)

Объектно-ориентированное программирование (ООП) — это способ организации программы, позволяющий использовать один и тот же код многократно. В отличие от функций и модулей, ООП позволяет не только разделить программу на фрагменты, но и описать предметы реального мира в виде удобных сущностей — объектов, а также организовать связи между этими объектами.

Объектно-ориентированные языки определяются следующими концепциями:

- инкапсуляция;
- наследование;
- полиморфизм;
- композиция.

Композиция – объект содержит другие объекты и активизирует их для выполнения соответствующих распоряжений. Каждый компонент может быть реализован в виде класса, который определяет собственное поведение и взаимосвязи.

Роботы по приготовлению пиццы являются совокупностями компонентов, которые работают вместе как одна команда. Скажем, чтобы наш робот был успешным, ему могут понадобиться манипуляторы для раскатывания теста, двигатели для перемещения к духовому шкафу и т.п.

Основным понятием ООП является класс — сложный тип данных, включающий набор переменных и функций для управления значениями, хранящимися в этих переменных. Переменные называют атрибутами (или свойствами), а функции — методами. Атрибуты классов обеспечивают поведение (данные и функции), которое наследуется всеми экземплярами, сгенерированными из них.

Класс позволяет создать неограниченное количество экземпляров, основанных на этом классе. Экземпляры представляют конкретные элементы в предметной области программы. Атрибуты экземпляров хранят данные, которые варьируются для каждого отдельного объекта.

Класс описывается с помощью ключевого слова class по следующей схеме:

Раздаточный материал № 91

class <Название класса> [(<Класс1>[, …, <КлассN>])]:
    ["""Строка документирования """]
    <Описание атрибутов и методов>

Инструкция создает новый объект и присваивает ссылку на него идентификатору, указанному после ключевого слова class. Это означает, что название класса должно полностью соответствовать правилам именования переменных. Все выражения внутри инструкции class выполняются при создании класса, а не его экземпляра.

Создание атрибута класса аналогично созданию обычной переменной. Метод внутри класса создается так же, как и обычная функция, — с помощью инструкции def. Методам класса в первом параметре, который обязательно следует указать явно, автоматически передается ссылка на экземпляр класса. Общепринято этот параметр называть именем self, хотя это и не обязательно. Доступ к атрибутам и методам класса внутри определяемого метода производится через переменную self с помощью точечной нотации.

Чтобы использовать атрибуты и методы класса, необходимо создать экземпляр класса:

Раздаточный материал № 92
<Экземпляр класса> = <Название класса> ([<Параметры>])

При обращении к методам класса:

Раздаточный материал № 93
<Экземпляр класса>.<Имя метода> ([<Параметры>])

При вызове метода не нужно передавать ссылку на экземпляр класса в качестве параметра, как это делается в определении метода внутри класса. Ссылку на экземпляр класса интерпретатор передает автоматически.

Обращение к атрибутам класса осуществляется аналогично:

Раздаточный материал № 94
<Экземпляр класса>.<Имя атрибута>

Наследование – это способность класса наследовать или расширять свойства другого класса в процессе выполнения. Это свойство позволяет производному классу получать свойства или черты базового класса. Наследование решает проблему повторного использования кода.

В Питоне классы образуют иерархию, в которой на верхнем уровне находится класс **Object**, и все остальные классы, в том числе созданные программистом в своей программе встраиваются в общее дерево иерархии классов.

Основные понятия:

1. Подкласс (производный, дочерний) - это класс, который наследует свойства от другого класса (обычно базового класса).

2. Суперкласс (родительский, базовый класс) - это класс, из которого происходят другие подклассы.

3. Подкласс обычно наследует/расширяет или полностью переопределяет некоторые методы базового класса.