**Тема занятия №16: Объектно-ориентированное программирование**

**1. Введение**

Наследование позволяет объявить класс, который дублирует функциональность уже существующего класса. С помощью этой концепции вы сможете расширить возможности своего класса.

Наследование — одна из концепций объектно-ориентированного программирования (ООП).

**2. Наследование в Python**

Как было сказано выше, наследование позволяет объявить класс, который либо не отличается от существующего, либо содержит минимальные изменения. Новый класс называется дочерним, а тот, у которого он наследует функционал — родительским.

**Синтаксис**

**class BaseClass:**

**Тело родительского класса**

**class DerivedClass(BaseClass):**

Тело дочернего класса

Особенность наследования заключается в том, что оно позволяет не просто создать дубликат класса, но и расширить его функционал. Это очень полезно, потому что наследование позволяет повторно использовать уже написанный код.

Пример использования наследования

Полигон — замкнутая геометрическая фигура. У полигона 3 и более сторон.

Давайте объявим класс Polygon:

**class Polygon:**

**def \_\_init\_\_(self, no\_of\_sides):**

**self.n = no\_of\_sides**

**self.sides = [0 for i in range(no\_of\_sides)]**

**def inputSides(self):**

**self.sides = [float(input("Введите сторону " + str(i+1)+ " : ")) for i in range(self.n)]**

**def dispSides(self):**

**for i in range(self.n):**

**print("Сторона", i+1, " — ", self.sides[i])**

В этом классе объявлено несколько переменных. Одна хранит количество сторон — n. Вторая, sides — это список, в нём находятся размеры сторон.

Метод inputSides() принимает размер каждой стороны, а dispSides() выводит их на экран.

Треугольник — это полигон с 3 сторонами. Теперь мы можем создать класс Triangle, который наследует весь функционал Polygon. Благодаря этому все атрибуты класса Polygon становятся доступны в Triangle.

Так что нам не нужно объявлять все переменные и методы снова. Давайте создадим класс Triangle:

**class Triangle(Polygon):**

**def \_\_init\_\_(self):**

**Polygon.\_\_init\_\_(self,3)**

**def findArea(self):**

**a, b, c = self.sides**

**# Вычисляем полупериметр**

**s = (a + b + c) / 2**

**area = (s\*(s-a)\*(s-b)\*(s-c)) \*\* 0.5**

**print('Площадь треугольника равна %0.2f' %area)**

В классе есть и собственный метод findArea(). Он вычисляет площадь треугольника и выводит ее на экран. Попробуем запустить нашу программу:

**>>> t = Triangle()**

**>>> t.inputSides()**

**Введите сторону 1 : 3**

**Введите сторону 2 : 5**

**Введите сторону 3 : 4**

**>>> t.dispSides()**

**Сторона 1 — 3.0**

**Сторона 2 — 5.0**

**Сторона 3 — 4.0**

**>>> t.findArea()**

Площадь треугольника равна 6.

Как видите, мы не объявляли методы inputSides() и dispSides() в классе Triangle. Но вот использовать их мы можем!

Если какой-либо атрибут не найдется в дочернем классе, Python пойдет искать в родительской. Этот поиск происходит рекурсивно, если родительский класс одного класса является дочерним для другого.

**3. Переопределение методов**

Стоит заметить, что в примере метод \_\_init\_\_() был объявлен в обоих классах — и в Triangle, и в Polygon. Здесь и происходит переопределение классов. То есть, метод в дочернем классе переопределяет тот же самый метод из родительского класса. Это значит, что \_\_init\_\_() в Triangle становится предпочтительнее \_\_init\_\_() в Polygon.

При переопределении метода родительского класса нужно стремиться к его расширению, а не простому копированию. Это, например, происходит при вызове метода в родительском классе из дочернего (вызов Polygon.\_\_init\_\_() из \_\_init\_\_() в Triangle).

Лучше всего использовать встроенную функцию super(). Например, super().\_\_init\_\_(3) эквивалентно вызову Polygon.\_\_init\_\_(self, 3). Старайтесь использовать именно этот способ.

**ООП. Множественное наследование**

Выполните следующие задания:

***Задание № 1***

Задан класс Point, описывающий точку с координатами x, y на координатной плоскости. Используя механизм наследования, нужно расширить возможности класса Point путем добавления нового атрибута цвета. Для этого реализовать подкласс PointColor.

В классе Point реализовать следующие атрибуты:

* координаты точки;
* метод иницализации, который получает 2 параметра — координаты точки x, y;
* метод вычисления расстояния от точки до начала координат;
* метод getPoint(), который возвращает точку в виде списка.

В подклассе PointColor реализовать следующие атрибуты:

* цвет точки color;
* метод начальной инициализации, который получает 3 параметра: координаты точки и цвет;
* метод доступа к цвету color с именем getColor().

***Задание № 2***

Создать базовый класс «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай», «Хомяк». С помощью конструктора установить имя каждого животного и его характеристики. Реализуйте для каждого из классов методы:

* Sound — издает звук животного (пишем текстом в консоль);
* Show — отображает имя животного;
* Type — отображает название его подвида.

***Задание № 3***

Создать базовый класс Employer (служащий) с функцией Print(). Она должна выводить информацию о служащем. В случае базового класса это может быть строка с надписью This is Employer class.

Создайте от него три производных класса: President, Manager, Worker.

Переопределите функцию Print() для вывода информации, соответствующей каждому типу служащего.