Занятие 24. Практическая работа. Создание классов и их использование

## План занятия

## 1. Создание классов:

Python проектировался как объектно-ориентированный язык программирования. Это означает, что он построен с учетом следующих принципов:

– все данные в нем представляются объектами;

– программу можно составить как набор взаимодействующих объектов, посылающих друг другу сообщения;

– каждый объект имеет собственную часть памяти и может состоять из других объектов;

– каждый объект имеет тип;

– все объекты одного типа могут принимать одни и те же сообщения (и выполнять одни и те же действия);

– каждый объект программы создается на основе некоторого класса объектов.

В Python реализованы следующие способы работы с классами:

– можно определять собственные классы;

– наследоваться от встроенных и собственных классов (одного или нескольких);

– производные классы могут переопределять любые методы базовых классов.

Следовательно, создание приложения на Python должно начинаться с проектирования и создания классов. Классы могут располагаться или вначале кода программы, или импортироваться из других файлов-модулей (также в начале кода).

Для создания классов предусмотрена **инструкция class**. Это составная инструкция, которая состоит из **строки заголовка** и **тела**. **Заголовок** состоит из ключевого слова **class**, **имени класса** и, возможно, **названий**

**суперклассов** (понятие суперкласса будет рассмотрено позже) в скобках. Суперклассов может и не быть, в таком случае скобки не требуются.

**Тело класса** состоит из блока различных инструкций. Тело должно иметь отступ (как и любые вложенные конструкции в языке Python).

Схематично класс можно представить следующим образом:

```
class Имякласса:
   переменная = значение
   …
   def имяметода(self, ...):
      self.переменная = значение
      …
      …
```

Данная схема не является полной. Например, в заголовке после имени класса могут быть указаны суперклассы (в скобках), а методы могут быть более сложными.

## 2. Атрибуты класса

**Атрибуты** класса — это имена переменных вне функций и имена функций. Эти атрибуты наследуются всеми объектами, созданными на основе данного класса. Атрибуты обеспечивают свойства и поведение объекта.

Атрибуты класса бывают двух видов:

– атрибуты данных;

– атрибуты-методы.

**Атрибуты данных** обычно записываются сверху. Память для атрибутов выделяется в момент их первого присваивания — либо снаружи, либо внутри метода.

**Атрибут-метод** или просто **метод** – это функция находящаяся внутри класса, выполняющая определенную работу, которая, чаще всего, предполагает доступ к атрибутам созданного объекта.

**Методы** класса — это небольшие программы, предназначенные для работы с объектами. Методы могут создавать новые свойства (данные) объекта, изменять существующие, выполнять другие действия над объектами. Методы определяются служебным словом def:

**def** имяметода(self,[список параметров])

Доступ к атрибутам выполняется по схеме **имяобъекта.имяатрибута**.

Пример.

```
class Simple:
   u'Простой класс'
   var = 87
   def f(self): return "Hello world"
```

Здесь Simple.var и Simple.f — пользовательские атрибуты.

Первым аргументом каждого **метода** класса всегда является текущий экземпляр класса. Общепринято называть этот аргумент **self** (аналог слова this в C++). Аргумент **self** ссылается на экземпляр класса, для которого вызывается метод, устанавливая связь с конкретным объектом.

В методе __init__ (рассмотрено далее) self ссылается на только что созданный объект. При вызове метода self не указывается, Python добавляет его автоматически.

Объекты могут иметь атрибуты, которые создаются в теле метода, если данный метод будет вызван для конкретного объекта.

Пример создания класса:

```
#!/usr/bin/env python2.7
# -*- coding: utf-8 -*-
class First:                    #Определение класса
   color = "red"                #Определение атрибута и задание ему значения

   def out(self):               #Определение метода класса
      print (self.color + "!")  #Тело метода
```

### 3. Создание объектов:

Объекты создаются так:

имяобъекта = Имякласса()

Здесь скобки обязательны! После такой инструкции в программе появляется объект, доступ к которому можно получить по имени переменной, связанной с ним. При создании объект получает атрибуты его класса, т. е. объекты обладают характеристиками, определенными в их классах.

Количество объектов, которые можно создать на основе того или иного класса, не ограничено.

Объекты одного класса имеют схожий набор атрибутов, а вот значения атрибутов могут быть разными. Другими словами, объекты одного класса похожи, но индивидуально различимы.

Вызов метода для конкретного объекта в основном блоке программы выглядит следующим образом:

имяобъекта.имяметода(…)

Ниже приведен пример создания классов, методов класса, объектов, вызовов методов класса:

```
#!/usr/bin/env python2.7
# -*- coding: utf-8 -*-
class First:                #Определение класса First
   color = "red"            #Определение и задание значения атрибута

    def out(self):          #Определение метода out. Параметр self указывает
                            #на принадлежность метода определяемому
                             #классу (First)
       print (self.color + "!")


class Second:               #Определение класса Second
   color = "red"            #Определение и задание значения атрибута
   form = "circle"          #Определение и задание значения атрибута

# Определение метода changecolor. Параметр self указывает на
# принадлежность метода определяемому классу (Second)
    def changecolor(self, newcolor):
      self.color = newcolor
```

```
# Определение метода changeform. Параметр self указывает на
# принадлежность метода определяемому классу (Second)
    def changeform(self, newform):
        self.form = newform


obj1 = Second () #Создание объекта obj1 класса Second ()
obj2 = Second () #Создание объекта obj2 класса Second ()
print (obj1.color, obj1.form) # вывод на экран "red circle"
print (obj2.color, obj2.form) # вывод на экран "red circle"

obj1.changecolor ("green") # изменение цвета первого объекта
obj2.changecolor ("blue") # изменение цвет второго объекта
obj2.changeform ("oval") # изменение формы второго объекта

print (obj1.color, obj1.form) # вывод на экран "green circle"
print (obj2.color, obj2.form) # вывод на экран "blue oval"
```

**Объект класса** и **экземпляр класса** — это два разных объекта. Первый генерируется на этапе объявления класса, второй — при вызове имени класса. Объект класса может быть один – он используется как определение принадлежащих ему объектов, экземпляров класса может быть сколько угодно – это сущности, созданные на основе класса.

Часто объект (не путать с объектом класса) и **экземпляр класса** используются как синонимы.

## 4. Конструктор класса

Конструктор класса позволяет задать определенные параметры объекта при его создании. Таким образом появляется возможность создавать объекты с уже заранее заданными атрибутами. Конструктором класса является метод: **__init__(self)**

Например, создадим класс **Rectangle** (Прямоугольник) с наперед заданными атрибутами: цветом, длиной и шириной:

```
class Rectangle:
```

```
    def __init__(self, color="green", width=100, height=100):
        self.color = color
        self.width = width
        self.height = height

    def square(self):
        return self.width * self.height


rect1 = Rectangle()
print(rect1.color)
print(rect1.square())
rect1 = Rectangle("yellow", 23, 34)
print(rect1.color)
print(rect1.square())
```

## 5. Практическое задание

1. Напишите два скрипта представленных выше. Посмотрите, как они работают. Во второй программу добавьте еще одно свойство и один метод, позволяющий его менять. Создайте третий объект и измените все его свойства.