Разработка программного обеспечения на языке Python

Лекция 5. Объектно-ориентированное программирование

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

Класс

В этой лекции мы подробно поговорим об объектно-ориентированном программировании на языке Python. Начнём с классов.

Несмотря на то, что Питон - мультипарадигменный язык (он сочетает в себе элементы структурного, объектно-ориентированного и функционального программирования), основной парадигмой все-таки является ООП.

Давайте для начала разберемся что такое класс. Класс - это шаблон. По нему определяется форма объекта. В нём указываются данные и код, который будет оперировать этими данными. То есть по сути класс - это ряд схематических описаний способа построения объекта. Здесь я подчеркну, что класс - это логическая абстракция. То есть физическое представление класса появится лишь после того, как будет создан объект этого класса.

Можно сказать, что класс - это новый тип, который определяется пользователем с помощью ключевого слова Класс.

После слова class идёт имя класса, далее, после двоеточия собственно содержимое класса. Внутри класса определяются его атрибуты, которые хранят различные характеристики класса, и методы - функции класса.

Создадим простейший класс Student.

```
class Student:
    pass

student1 = Student()
```

В данном случае определен класс Person, который условно представляет человека. В данном случае в классе не определяется никаких методов или атрибутов. Однако поскольку в нем должно быть что-то определено, то в качестве заменителя функционала класса применяется оператор pass. Этот оператор применяется, когда синтаксически необходимо определить некоторый код, однако мы не хотим его, и вместо конкретного кода вставляем оператор pass.

После создания класса можно определить объекты этого класса. Например, здесь мы определили объект student1.

```
class Student:
    def message (self):
        print("Я студент ИКИТа!")

student1 = Student()
student1.message()
```

Для создания объекта применяется специальная функция - конструктор, которая называется по имени класса и которая возвращает объект класса. То есть в данном случае вызов Student() представляет вызов конструктора. Каждый класс по умолчанию имеет конструктор без параметров.

Методы класса фактически представляют функции, которые определенны внутри класса и которые определяют его поведение. Например, определим функцию, которая будет выводить сообщение на консоль «Я студент ИКИТа».

Вы можете увидеть в функции message параметр self. Этот параметр обозначает ссылку на текущий объект. Через эту ссылку внутри класса мы можем обратиться к функциональности текущего объекта. Но при самом вызове метода этот параметр не учитывается.

Ниже в коде, через имя объекта, мы можем вызвать метод через точку: название объекта точка название метода: student 1.message.

При создании объекта student1 мы использовали конструктор по умолчанию. Но при необходимости мы можем явным образом определить в классах конструктор с помощью специального метода, который называется __init__() (по два прочерка с каждой стороны). Добавим в наш класс конструктор:

```
class Student:
    def __init__(self):
        print("Создание объекта Student")
    def message (self):
        print("Я студент ИКИТа!")

student1 = Student()
student1.message()
```

В качестве первого параметра конструктор, как и методы, также принимает ссылку на текущий объект - self. Обычно конструкторы применяются для определения действий, которые должны производиться при создании объекта. Например, в данном случае при создании объекта student1 будет выводиться на консоль фраза «Создание объекта Student»

Еще одними значимыми членами класса является атрибуты. Атрибуты представляют некоторую характеристику класса. Добавим в наш класс атрибуты: имя, группа, список изучаемых дисциплин.

```
class Student:
    def __init__(self,name,group):
        self.name = name
        self.group = group
        self.discipline = []
    def message (self):
        print(self.name + " студент группы " + self.group +
              " изучаю дисциплины " + str(self.discipline))

dis = ["Алгебра","Геометрия","Программирование"]

student1 = Student("Иван","КИ22-01")
student1.discipline = dis
student1.message()
```

Внутри конструктора устанавливаются два атрибута - name и group (имя и группа студента_

Атрибутам self.name и self.group присваивается значение переменной name и group. Атрибут discipline получает значение пустого списка.

Далее, уже после класса в коде мы можем создать список и присвоить значение этого атрибута равным этому списку.

Если мы определили в классе конструктор __init__, мы уже не сможем вызвать конструктор по умолчанию. Теперь нам надо вызывать наш явным образом определенный конструктор __init__, в который необходимо передать значение для параметров name и group.

Итак, в этом видео мы разобрали классы, рассмотрели, как можно определить класс и какие бывают члены класса.

Инкупсуляция

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◄ Задание 10. 38 попугаев

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

Задание 11. Привет из 90-х.Тамагочи ►

Контакты +7(391) 206-27-05
info-ms@sfu-kras.ru

Скачать мобильное приложение

Инструкции по работе в системе