Основы ООП

Обычное программирование (процедурное)

- Чаще всего под обычным понимают процедурное программирование, в основе которого – процедуры и функции. Функция – это мини-программа, которая получает на вход какие-то данные, что-то делает внутри себя и может отдавать какие-то данные в результате вычислений.

Обычное программирование (процедурное)

- Процедурное программирование идеально работает в простых программах, где все задачи можно решить, грубо говоря, десятком функций. Функции аккуратно вложены друг в друга, взаимодействуют друг с другом, можно передать данные из одной функции в другую.

Обычное программирование (процедурное)

- Но теперь представьте, что у вас этих функций — сотни. И изменений в них нужно делать десятки в день. И каждое изменение, как правило, заставляет функции вести себя более сложным образом и выдавать более сложный результат. И каждое изменение в одном месте ломает три других места. В итоге у вас будут нарождаться десятки клонированных функций, в которых вы сначала будете разбираться, а потом уже нет.

- Это называется спагетти-код, и для борьбы с ним как раз придумали объектно-ориентированное программирование.

- Объектно-ориентированное программирование (ООП) – это парадигма программирования, где различные компоненты компьютерной программы моделируются на основе реальных объектов. Объект – это что-либо, у чего есть какие-либо характеристики и то, что может выполнить какую-либо функцию.

- Преимущества и недостатки ООП Python

Рассмотрим несколько основных преимуществ объектно-ориентированного программирования:

- Объектно-ориентированное программирование подразумевает повторное использование. Компьютерная программа написанная в форме объектов и классов может быть использована снова в других проектах без повторения кода;
- Использование модулярного подхода в объектно-ориентированном программировании позволяет получить читаемый и гибкий код;
- В объектно-ориентированном программировании каждый класс имеет определенную задачу. Если ошибка возникнет в одной части кода, вы можете исправить ее локально, без необходимости вмешиваться в другие части кода;
- Инкапсуляция данных вносит дополнительный уровень безопасности в разрабатываемую программу с использованием объектно-ориентированного подхода;

Хотя объектно-ориентированное программирование обладает рядом преимуществ, оно также содержит определенные недостатки, некоторые из них находятся в списке ниже:

- Для создания объектов необходимо иметь подробное представление о разрабатываемом программном обеспечении;
- Не каждый аспект программного обеспечения является лучшим решением для реализации в качестве объекта. Для новичков может быть тяжело прочертить линию в золотой середине;
- С тем, как вы вносите все новые и новые классы в код, размер и сложность программы растет в геометрической прогрессии;

Класс

Класс в объектно-ориентированном программировании выступает в роли чертежа для объекта. Класс можно рассматривать как схему дома. Вы можете понять, как выглядит дом, просто взглянув на его карту.

Сам по себе класс не представляет ничего. К примеру, нельзя сказать что схема является домом, она только объясняет как настоящий дом должен выглядеть.

Ключевое слово class используется для создания класса в Python. Название класса следует за ключом class, за которым следует двоеточие. Тело класса начинается с новой строки, с отступом на одну вкладку влево.

```python
# Создаем класс Car
class Car:

    # создаем атрибуты класса
    name = "c200"
    make = "mercedez"
    model = 2008

    # создаем методы класса
    def start(self):
        print ("Заводим двигатель")

    def stop(self):
        print ("Отключаем двигатель")
```

- В примере выше мы создали класс под названием Car с тремя атрибутами: имя name, марка make и модель model. Наш класс также содержит два метода: start() и stop().

Объекты

Раннее мы поняли, что класс предоставляет чертеж объекта. Однако, чтобы на самом деле использовать объекты и методы класса, вам нужно создать объект из этого класса. Существует несколько методов и атрибутов класса, которые можно использовать вне объекта, мы рассмотрим их в следующем разделе.

Сейчас просто запомните, что по умолчанию, нам нужно создать объект класса перед тем, как мы сможем начать использовать его методы и атрибуты.

Объект также называется экземпляром. Тем не менее, процесс создания объекта класса называется инициализация. В Python, чтобы создать объект класса, нам просто нужно вписать название класса, с последующими открывающимися и закрывающимися скобками.

```python
# Создаем объект класса Car под названием car_a
car_a = Car()

# Создаем объект класса Car под названием car_b
car_b = Car()
```

Атрибуты класса

В предыдущей секции мы разобрались, как создавать объекты класса и как мы можем использовать эти объекты для получения доступа к атрибутам класса

В Python, каждый объект содержит определенные атрибуты по умолчанию и методы в дополнение к определенным пользователем атрибутами. Чтобы посмотреть на все атрибуты и методы объекта, используйте встроенную функцию под названием dir().

Атрибуты класса против атрибутов экземпляров

Атрибуты могут быть наглядно отнесены к двум типам:

- атрибуты класса
- атрибуты экземпляров

Атрибуты класса делятся среди всех объектов класса, в то время как атрибуты экземпляров являются собственностью экземпляра.

Помните, что экземпляр – это просто альтернативное название объекта.

Атрибуты экземпляра объявляются внутри любого метода, в то время как атрибуты класса объявляются вне любого метода.

```python
class Car:

    # создаем атрибуты класса
    car_count = 0

    # создаем методы класса
    def start(self, name, make, model):
        print("Двигатель заведен")
        self.name = name
        self.make = make
        self.model = model
        Car.car_count += 1
```

Давайте создадим объект класса `Car` и вызовем метод `start()`.

```python
car_a = Car()
car_a.start("Corrola", "Toyota", 2015)
print(car_a.name)
print(car_a.car_count)
```