Объектно-ориентированное программирование

Практикум по программированию

Принципы ООП

1. **Полиморфизм**: в разных объектах одна и та же операция может выполнять различные функции. Простым примером полиморфизма может служить функция *count()*, выполняющая одинаковое действие для различных типов обьектов: *'abc'.count('a')* и *[1, 2, 'a'].count('a')*. Оператор плюс полиморфичен при сложении чисел и при сложении строк.
2. **Инкапсуляция**: можно скрыть ненужные внутренние подробности работы объекта от окружающего мира.
3. **Наследование**: можно создавать специализированные классы на основе базовых. Это позволяет нам избегать написания повторного кода.
4. **Композиция**: объект может быть составным и включать в себя другие объекты.

Терминология ООП

- **Класс** (Class): Определенный программистом прототип программируемого объекта с набором атрибутов (переменных и методов), которые описывают данный объект. Доступ к атрибутам и методам осуществляется через точку
- **Переменная класса** (Class variable): Переменная, доступная для всех экземпляров данного класса. Определяется внутри класса, но вне любых методов класса.
- **Экземпляр класса** (Instance): Отдельный объект-представитель определенного класса.
- **Переменная экземпляра класса** (Instance variable): Переменная определенная внутри метода класса, принадлежащая только к этому классу.
- **Метод** (Method): Особая функция, определенная внутри класса.
- **Наследование** (Inheritance): Передача атрибутов и методов родительского класса дочерним классам.
- **Перегрузка функций** (Function overloading): Изменение работы метода, унаследованного дочерним классом от родительского класса.

Создание класса

```
class Elevator:
    """ Simple elevator class """
    # Переменная класса. Сколько людей было перевезено ВСЕМИ лифтами
    people_lifted = 0

    # Конструктор класса. Вызывается при создании экземпляра класса
    def __init__(self,name):
        self.name = name
        # переменная класса. Количество людей перевезенных КОНКРЕТНЫМ лифтом
        self.people_lifted = 0

    # Метод перевозки людей
    def lift(self):
        print ("{} lifted someone".format(self.name))
        # Увеличиваем количество людей перевезенных ЭТИМ лифтом
        self.people_lifted += 1
        # Увеличиваем количество людей перевезенных ВСЕМИ лифтами
        Elevator.people_lifted += 1

    # Метод печатающий информацию о конкретном лифте
    def info(self):
        print (self.name, "lifted", self.people_lifted, "people out of", Elevator.people_lifted)
```

Создание экземпляров класса

Чтобы создать экземпляр класса следует любой переменной присвоить значение имени класса, указав в скобках аргументы, которые принимает метод __init__().

```
elevator_1 = Elevator("OTIS")
elevator_2 = Elevator("PHILLIPS")
```

Получение доступа к атрибутам и методам

Чтобы получить доступ к атрибутам класса в Python следует после объекта поставить точку и написать имя переменной или метода, которые вы хотите использовать:

```
# Везем человека в лифте под именем OTIS
elevator_1.lift()
# Везем двоих человек в лифте под именем PHILLIPS
elevator_2.lift()
elevator_2.lift()
# Получаем информацию по лифту под именем OTIS
elevator_1.info()
# Получаем информацию по лифту под именем PHILLIPS
elevator_2.info()
```