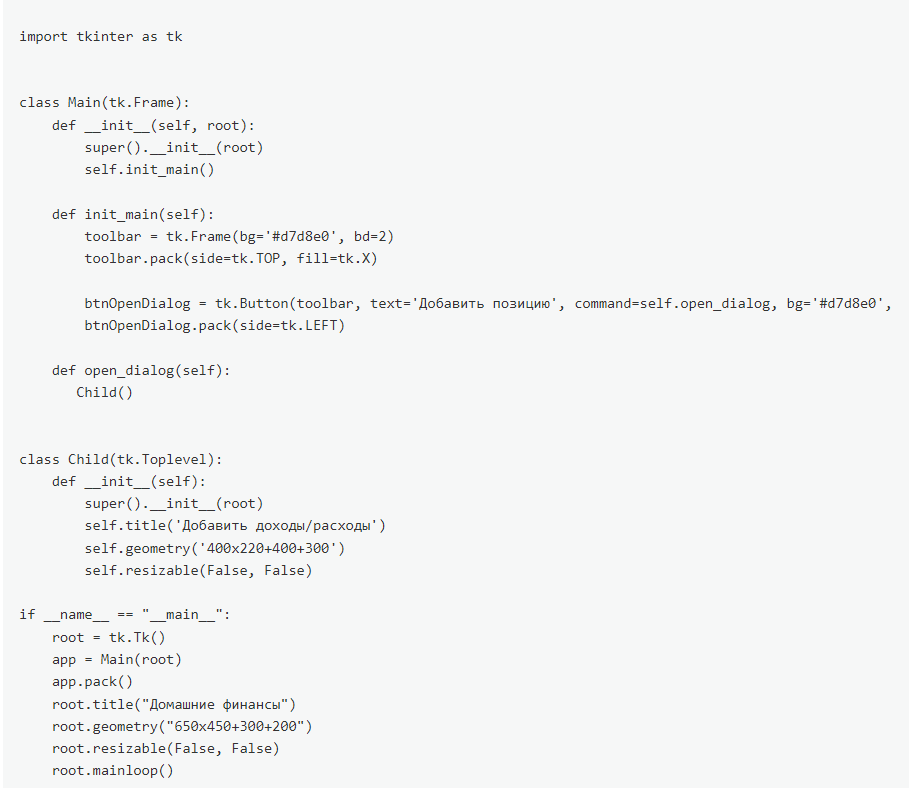
**Тема занятия №26: Программы с интерфейсом (GUI)**

**1. ООП стиль tkinter**

Самым важным в подходе является определение «сущностей», которые и реализуют большую часть функционала, совместно с взаимодействиями.

Например:



Тут мы видим, что есть «главный» объект интерфейса, и «дочерний».

Более подробный пример:

Объектно-ориентированное программирование – очень удобный тип программирования, который позволяет работать с данными, как с объектами. Это позволяет с легкостью модифицировать определенные объекты кода, осуществлять откат, если возникают какие-то ошибки. Такие программы легко расширяются дополнительными функциями, если разработчик предусматривает такую возможность. И в целом, объектно-ориентированное программирование позволяет существенно упростить разработку программы.

Сегодня рассмотрим, как осуществляется работа с объектами в Tkinter. Ведь все мы понимаем, что каждый класс – это объект, верно?

Использование классов для структурирования данных

Очень просто демонстрировать использование классов в Python на примере приложения со списком контактов. Несмотря на то, что интерфейс будет отличаться, нам необходимо определить доменную модель. В нашем случае – каждый контакт.

Какая информация туда включается?

Имя и фамилия. Очевидно, что здесь должно храниться какое-то значение.

Адрес электронной почты. Допустим, у нас он следующий: alexx345@gmail.com

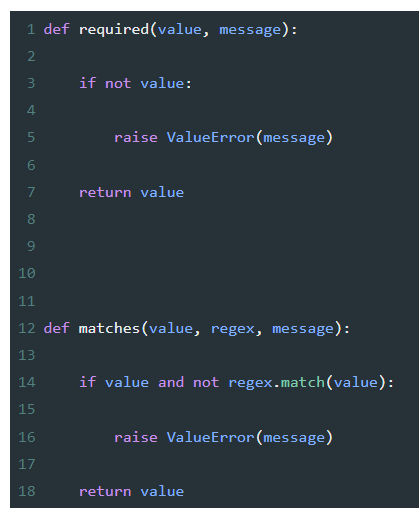
Номер телефона. Он также указывается в определенном формате. Например, таком: (345) 6789012

Это абстракция – набор обобщенных характеристик, которые и отличают элемент определенного класса. В нашем случае таковым будет Contact.

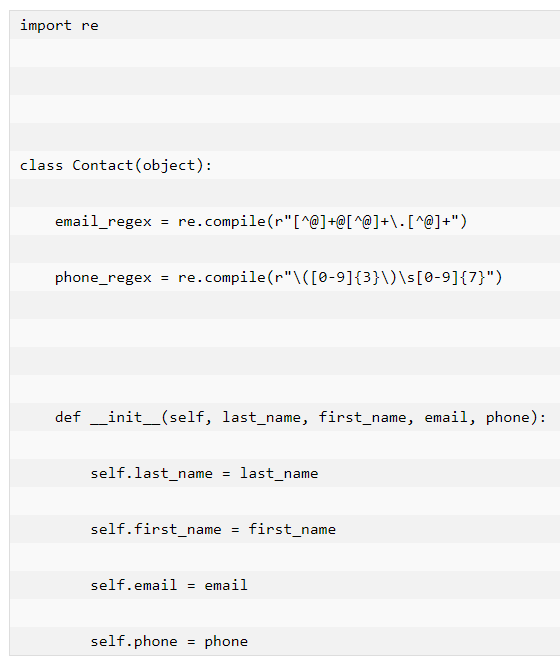
Сперва необходимо определить служебных функций, используемых для валидации обязательных полей.

Также важно определить те из них, которые выполняют проверку на соответствие полей ввода электронной почты и номера телефона тем форматам, которые указаны выше.

В результате, получится следующий код:



Затем необходимо определить наш класс Contact, а также метод \_\_init\_\_, в котором укажем все параметры полей, в которые будет вводиться информация. Также скомпилированные регулярные выражения необходимо сохранить, так как они будут применяться для всех элементов во время проверки полей.



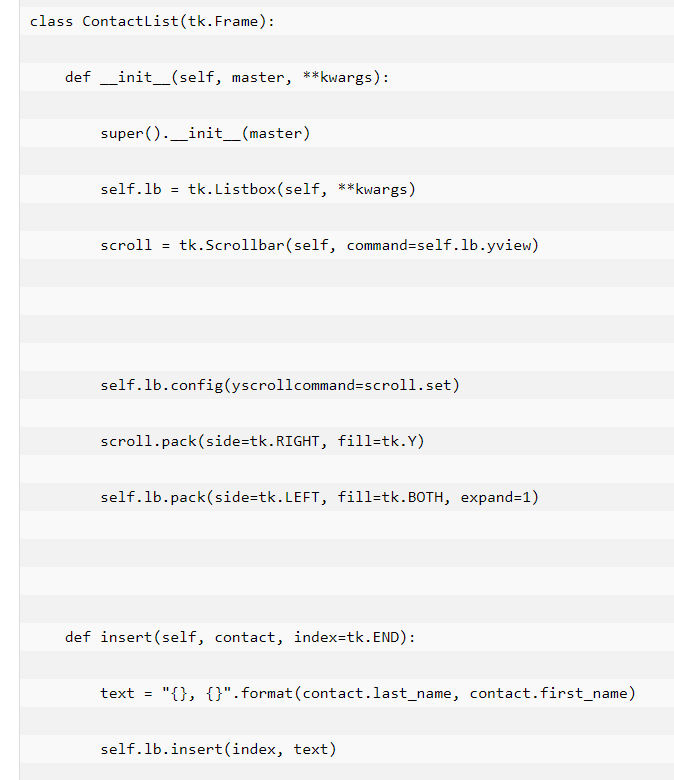
Далее этот готовый код нужно будет использовать. Для этого необходимо сохранить его в отдельном файле, и потом на него ссылаться.

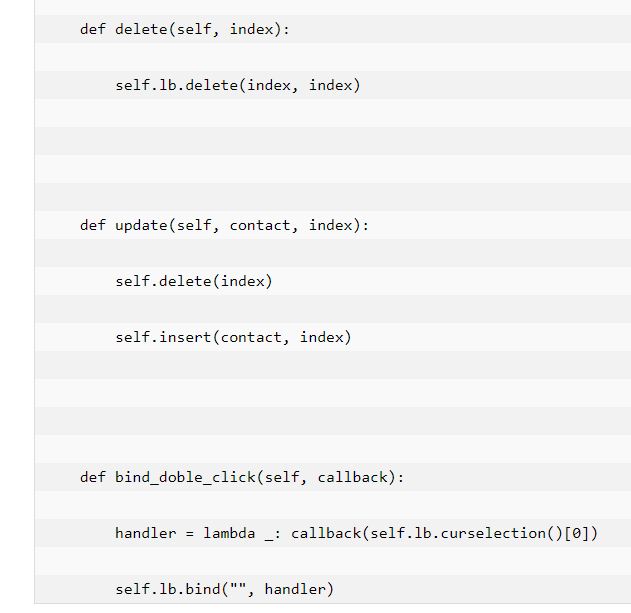
Как создавать виджеты для отображения информации?

Чтобы создавать приложения было комфортно, необходимо разбивать его на отдельные классы. Тяжело сделать качественно приложение, если весь код предусмотрен лишь в одном классе. А вот при модульной организации приложения появляется возможность создавать отдельный элемент для каждой задачи.

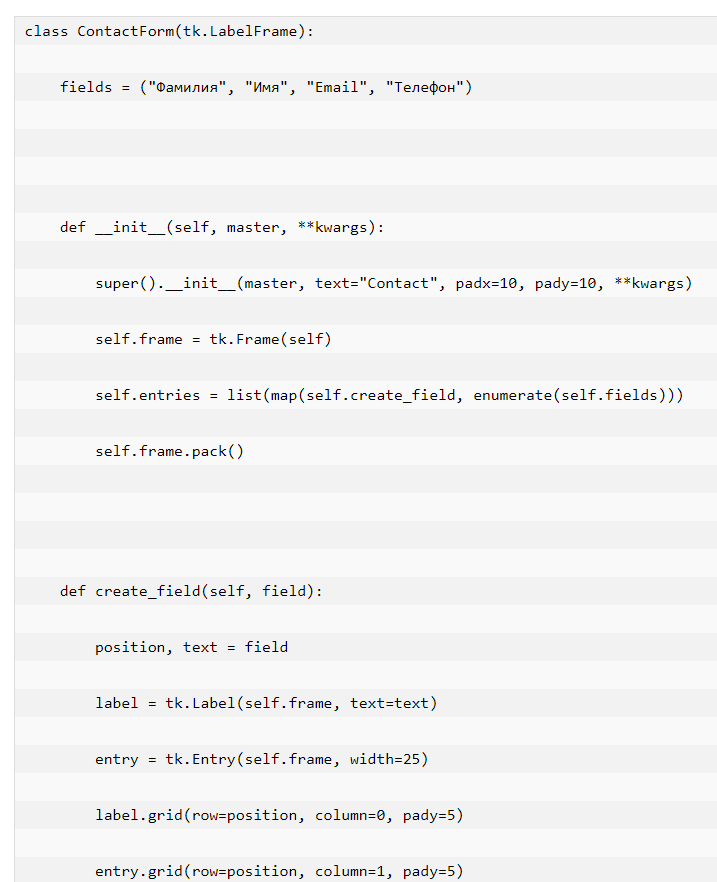
В прошлом примере мы создали класс Contact. Его необходимо импортировать.

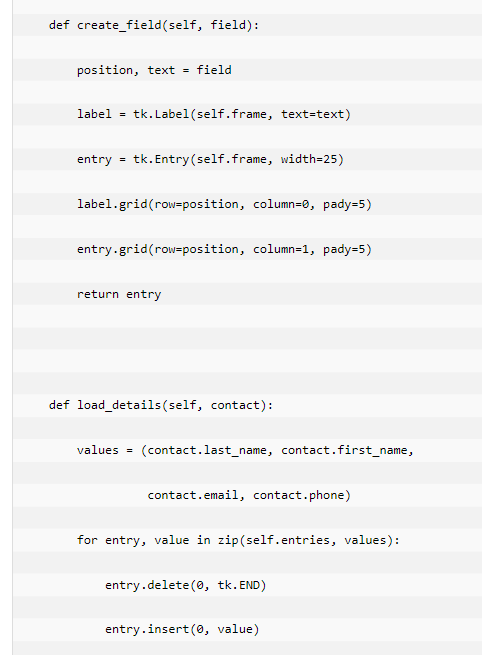


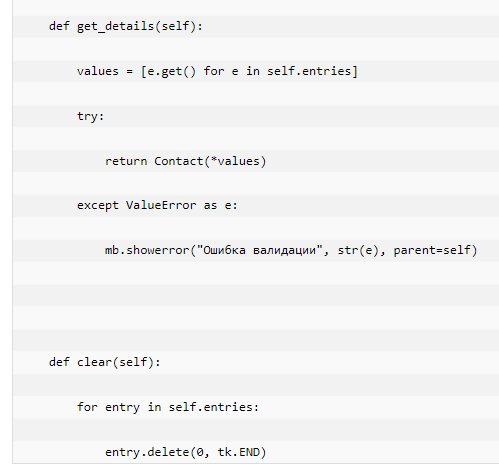




Чтобы была возможность просматривать и модифицировать контакты, необходимо создать отдельную форму. Для начала попробуем взять в качестве базового класса LabelFrame с Label и Entry для каждого поля







Наш виджет предоставляет возможность добавить функцию обратного вызова для клика левой кнопкой мыши два раза. Также с его помощью можно получить индекс клика в качестве аргумента функции. Все это позволяет спрятать особенности внутреннего класса Listbox.

