**Упражнение №1**

Напишите «абстрактный» класс Box и используйте его для определения некоторых методов, которые должен иметь любой объект коробки: add для добавления любого количества элементов в коробку, empty для извлечения всех элементов из коробки и возврата их в виде списка, и count, для подсчета предметов, которые в данный момент находятся в коробке.

Напишите простой Item класс, у которого есть name атрибут и value атрибут — вы можете предположить, что все элементы, которые вы будете использовать, будут Item объектами.

Теперь напишите два подкласса Box, которые используют разные базовые коллекции для хранения элементов: ListBox должны использовать список и DictBox должны использовать словарь.

Напишите функцию, repack\_boxes которая принимает любое количество ящиков в качестве параметров, собирает все содержащиеся в них элементы и перераспределяет их как можно более равномерно по всем ящикам. Порядок неважен. Есть несколько способов сделать это. Протестируйте свой код с ListBox 20 элементами, ListBox с 9 элементами и DictBoxс 5 элементами. У вас должно получиться две коробки по 11 предметов в каждой и одна коробка с 12 предметами.

**Упражнение №2**

Напишите простую программу, которая перебирает список пользовательских данных (кортежи, содержащие имя пользователя, адрес электронной почты и возраст) и добавляет каждого пользователя в каталог, если ему не менее 16 лет. Вам не нужно хранить возраст. Напишите простую иерархию исключений, которая определяет разные исключения для каждого из этих условий ошибки:

* имя пользователя не уникально
* возраст не является положительным целым числом
* пользователю меньше 16 лет
* адрес электронной почты недействителен (достаточно простой проверки имени пользователя, @символа и доменного имени)

Поднимите эти исключения в своей программе, где это уместно. Всякий раз, когда возникает исключение, ваша программа должна перейти к следующему набору данных в списке. Распечатайте разные сообщения об ошибках для каждого типа исключений.

Подумайте, где еще было бы неплохо использовать пользовательский класс и какой тип коллекции будет наиболее подходящим для вашего каталога.

Вы можете считать адрес электронной почты действительным, если он содержит один @символ и имеет непустое имя пользователя и доменное имя — вам не нужно проверять наличие допустимых символов. Можно предположить, что возраст уже является целочисленным значением.

**Упражнение №3**

Кратко опишите возможный набор классов, которые можно использовать для представления музыкальной коллекции (например, внутри музыкального проигрывателя), сосредоточив внимание на том, как они будут связаны по композиции. Вы должны включить классы для песен, исполнителей, альбомов и плейлистов. Подсказка: запишите четыре имени класса, нарисуйте линию между каждой парой классов, которые, по вашему мнению, должны иметь отношения, и решите, какой вид отношений будет наиболее подходящим.

Для простоты вы можете предположить, что у любой песни или альбома есть одно значение «исполнитель» (которое может представлять более одного человека), но вы должны включать альбомы-сборники (которые содержат песни разных исполнителей). «Исполнитель» сборника может иметь особое значение, например, «Разные исполнители». Вы также можете предположить, что каждая песня связана с одним альбомом, но может существовать несколько копий одной и той же песни (которые включены в разные альбомы).

Напишите простую реализацию этой модели, которая ясно показывает, как составлены различные классы. Напишите пример кода, чтобы показать, как вы будете использовать свои классы для создания альбома и добавления всех его песен в список воспроизведения. Подсказка: если два объекта связаны друг с другом двунаправленно, вам придется решить, как должна формироваться эта связь — один из объектов должен быть создан раньше другого, поэтому вы не можете связать их друг с другом в обоих направлениях. одновременно!

**Упражнение №4**

Дан листинг:

class Learner:

def \_\_init\_\_(self):

self.classes = []

def enrol(self, course):

self.classes.append(course)

class Teacher:

def \_\_init\_\_(self):

self.courses\_taught = []

def assign\_teaching(self, course):

self.courses\_taught.append(course)

class Person:

def \_\_init\_\_(self, name, surname, number, learner=None, teacher=None):

self.name = name

self.surname = surname

self.number = number

self.learner = learner

self.teacher = teacher

jane = Person("Jane", "Smith", "SMTJNX045", Learner(), Teacher())

jane.learner.enrol(a\_postgrad\_course)

jane.teacher.assign\_teaching(an\_undergrad\_course)

Перепишите Person класс, реализовав вызываемые дополнительные методы, enrol которые assign\_teaching скрывают делегирование. Эти методы должны вызывать соответствующее сообщение об ошибке, если делегирование не может быть выполнено, поскольку соответствующий атрибут не установлен.