Final Project: Contacts

CSE/IT 107L

NMT Department of Computer Science and Engineering

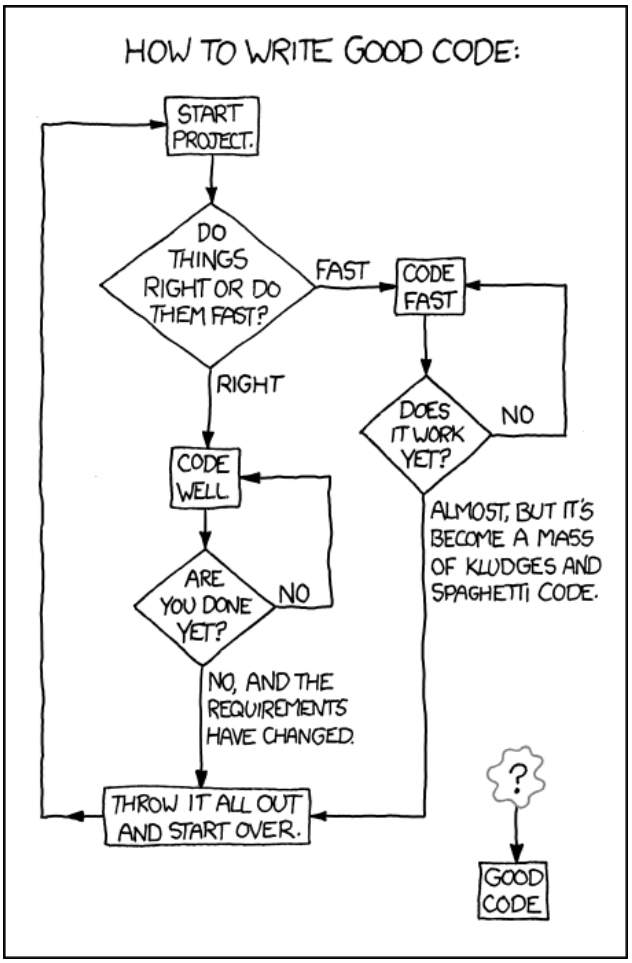

Figure 1: XKCD on how to write good code `https://xkcd.com/844/`.

Contents

Grading

60% Code Implementation

10% Design

10% Readme

10% Docstring Comments

05% Code follows PEP 8 and PEP 257 style guides

05% Tarball is named correctly. i.e. `cse107_firstname_lastname_contacts.tar.gz`

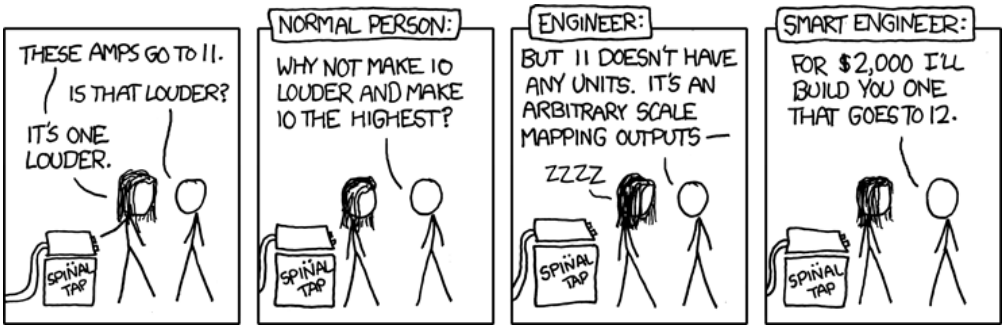

Figure 2: XKCD on engineers `https://xkcd.com/670/`.

1 The Problem

You are an employee of NMTLabs, a small software consultation firm for noble gas mass spectrometry. At this week's developer meeting you were tasked with developing an application to manage the firm's contacts. The CEO and CTO of NMTLabs are looking for something that is text-based and easy to use, so the firm can quickly start calling contacts for the latest round of venture capital fundraising. Heretofore the firm has been using a simple CSV file to save all of its contacts. Everyone in the organization is looking for an improved solution and are counting on you to have a working prototype.

The CTO has specified a set of tasks your contacts application must perform. Please examine the requirements carefully, implement as many of the tasks as possible including the Extra Credit (E.C worth 1% each)

1.1 Example Session

An example session using the application is shown below. A user starts the application, enters the commands, about, info, list and finally exit. The application displays a farewell message and quits. Note line 1 is a terminal command.

```
1 $ python3 contacts.py
2 Welcome to the Contact application
3 Please enter a command: about
4 Contacts App. Developed by Jake Ross for CSE107 2015
5
6 Please enter a command: info
7 contacts path: contacts.txt
8 number contacts: 3
9
10 Please enter a command: list
11 Name       : Phone     Company email
12 ---------------------------------------------
13 Elon Musk  : 453-6723  SpaceX  emusk@spacex.com
14 Larry Page : 853-0653  Google  lpage@gmail.com
15 Tim Cook   : 133-0419  Apple   tcook@apple.com
16
17 Please enter a command: exit
18 Goodbye
```

2 Assignment

Make your code clean and readable. You will be graded on style and functionality. **You must follow PEP8 and PEP257**.

2.1 Design

Provide a description of how you plan to solve this problem. Draw figures, flowcharts, and tables with detailed annotations (notes). Consider and describe typical use cases. For example,

1. The application starts.

2. The user wants to add a contact.
3. The program sequentially asks for required information validating each inputted value.
4. The program adds contact to its stored contacts.
5. The added contact information is displayed to the user.

2.2 README

Write a `README.txt` file. This is common practice in the open source community and provides a consistent location for users to find introductory information about your application. This file should contain a brief description of what your code does and some information on how to run it. You should provide a list of valid commands, their function, and examples of their use.

2.3 Contacts Format

A contact is something that stores least the following attributes:

- Name
- Phone
- Company
- Email
- Note

Here is a sample CSV file describing many contacts.

```
Name, Phone, Company, Email, Note
Elon Musk, 453-6723, SpaceX, emusk@spacex.com
Larry Page, 853-0653, Google, lpage@gmail.com
Tim Cook, 133-0419, Apple, tcook@apple.com
```

2.4 Application Commands

- Get input from the user.
- Think of and display a unique name for your application
- Display a welcome message when the application starts.

To make testing easier, add some default contacts so you don't have to constantly load a CSV file or manually enter contacts.

2.4.1 The `exit` Command

- Close the program.
- Display a goodbye message when the application quits.

2.4.2 The `about` Command

- Print developer information. Developer name, date created, ... etc

2.4.3 The `info` Command

- Print number of contacts
- Print number of companies
- Print number of contacts per company

2.4.4 The `list` Command

- List all contacts.

2.4.5 The `remove` Command

- Report contacts updated
- Warn user if trying to remove contact that does not exist

2.4.6 The `note` Command

- Allow user to edit a note associated with a specified contact
- Allow user to see current note

2.4.7 The `add` Command

- Allow user to add a contact. There should be a prompt for each field of a contact.

2.4.8 The `load` Command

- Load a default contacts file when the application starts
- Warn user if entered invalid file

2.4.9 The `save` Command

Use this command to write a line of a CSV file:

```
savefile.write("{}, {}, {}, {}, {}\n".format(
    name, phone, company, email, note))
```

2.4.10 The `commands` Command

- Load a set of commands from a file and execute them. An example command file might look like the following.

```
add joe
phone: 555-5555
add jane
phone: 555-5555
email: jane@jane.jane
remove bob
```

2.5 Extra Credit

- Use a random welcome and goodbye message chosen from a set of available messages.
- Verify CSV save path can be written to, i.e parent directory exits
- Validate user input in add or edit commands. Verify phone, email correct formats. i.e. (xxx)xxx-xxxx
- Add the `lookup` Search for contacts by substring. Offer case-insensitive search.
- Allow multiple notes for a given contact. User should be able to list, add, remove notes.
- Stay in each mode before asking user for another command. For example if user wants to add a bunch of contacts, enter 'add' mode, add contacts manual, and explicitly exit add mode with 'exit'
- Group contacts. Add ability to manage groups of contacts. Contacts may exist in multiple groups. Add an additional command set for working with groups, i.e add, remove, list groups, list contacts in group etc.
- Import contacts from a csv file
- Export contacts to YAML, XML, JSON, etc...
- Associate an image with each contact. Add a command to display the contact's image.
- Add user login. Each user has a distinct set of contacts that is automatically loaded when the user logs in.
- Use Object Oriented Programming to implement your project.

3 Submitting

You should submit your code as a tarball. It should contain all files used in solving the problems presented in this lab. If you want to include hand drawings, scan and include as PDFs. The submitted file should be named

`cse107_firstname_lastname_contacts.tar.gz`

Upload your tarball to Canvas.