Review for Final Exam
Speech to text
if you want to do Speech-to-Text you have to follow the instructions here:

https://pypi.python.org/pypi/SpeechRecognition/
SpeechRecognition 3.6.5

Library for performing speech recognition, with support for several engines and APIs, online and offline.

Speech recognition engine/API support:

- CMU Sphinx (works offline)
- Google Speech Recognition
- Google Cloud Speech API
- Wit.ai
- Microsoft Bing Voice Recognition
- Houndify API
- IBM Speech to Text

Quickstart: pip install SpeechRecognition. See the “Installing” section for more details.

To quickly try it out, run python -m speech_recognition after installing.

Project links:

- PyPI
- Source code
- Issue tracker
Installing

First, make sure you have all the requirements listed in the "Requirements" section.

The easiest way to install this is using `pip install SpeechRecognition`.

Otherwise, download the source distribution from PyPI, and extract the archive.

In the folder, run `python setup.py install`.

Requirements

To use all of the functionality of the library, you should have:

- **Python 2.6, 2.7, or 3.3+ (required)**
- **PyAudio 0.2.9+ (required only if you need to use microphone input, Microphone)**
- **PocketSphinx** (required only if you need to use the Sphinx recognizer, `recognizer_instance.recognize_sphinx`)
- **Google API Client Library for Python** (required only if you need to use the Google Cloud Speech API, `recognizer_instance.recognize_google_cloud`)
- **FLAC encoder** (required only if the system is not x86-based Windows/Linux/OS X)

The following requirements are optional, but can improve or extend functionality in some situations:

- On Python 2, and only on Python 2, some functions (like `recognizer_instance.recognize_bing`) will run slower if you have **Monotonic for Python 2** installed.
- If using CMU Sphinx, you may want to **install additional language packs** to support languages like International French or Chinese.

The following sections go over the details of each requirement.

**Python**

The first software requirement is **Python 2.6, 2.7, or Python 3.3+**. This is required to use the library.
PyAudio

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X / macOS.

PyAudio is inspired by:

- pyPortAudio/fastaudio: Python bindings for PortAudio v18 API.
- tkSnack: cross-platform sound toolkit for Tcl/Tk and Python.

What's new

March 18, 2017
PyAudio 0.2.11 is a new release with a bug fix related to memory management.

Many thanks to both Blaise Potard and Matthias Schaff for discovering the issue and for their patches! Thanks as well to Timothy Port for helping to correct a docstring.

January 16, 2017
PyAudio 0.2.10 is a new release with bug fixes related to the Python GIL. It also introduces a few automated unit tests.

Great thanks to Michael Graczyk for discovering the GIL-related issues and for submitting a patch!

October 19, 2015
PyAudio installation is better streamlined. Install PyAudio on most platforms using pip.

October 18, 2015
PyAudio 0.2.9 is a new release with bug fixes related to overflow error handling and IOError exception arguments.

Many thanks to Tony Jacobson for discovering and helping with the overflow error. Thanks also to Sami Liedes for reporting the IOError exception issue!
Installation

The current version is **PyAudio v0.2.11**. Install PyAudio using pip on most platforms. For versions prior to v0.2.9, PyAudio distributed installation binaries, which are [archived here](https://github.com/cdf/PyAudio/releases).

---

**Microsoft Windows**

Install using **pip**:

```
python -m pip install pyaudio
```

**Notes:**

- If pip is not already bundled with your installation of Python, get it [here](https://pip.pypa.io/en/stable/install/).
- pip will fetch and install PyAudio wheels (prepackaged binaries). Currently, there are wheels compatible with the official distributions of Python 2.7, 3.4, 3.5, and 3.6. For those versions, both 32-bit and 64-bit wheels are available.
- These binaries include PortAudio v19 v190600_20161030, built with MinGW. They support only the Windows MME API and do not include support for DirectX, ASIO, etc. If you require support for APIs not included, you will need to compile PortAudio and PyAudio.

---

**Apple Mac OS X**

Use **Homebrew** to install the prerequisite portaudio library, then install PyAudio using pip:

```
brew install portaudio
pip install pyaudio
```

**Notes:**

- If not already installed, download [Homebrew](https://brew.sh).  
- pip will download the PyAudio source and build it for your version of Python.  
- Homebrew and building PyAudio also require installing the Command Line Tools for Xcode [more information](https://developer.apple.com/xcode/getting-started).
Installing

First, make sure you have all the requirements listed in the "Requirements" section.

The easiest way to install this is using `pip install SpeechRecognition`.

Otherwise, download the source distribution from PyPI, and extract the archive.

In the folder, run `python setup.py install`.

Requirements

To use all of the functionality of the library, you should have:

- **Python 2.6, 2.7, or 3.3+** (required)
- **PyAudio 0.2.9+** (required only if you need to use microphone input, `Microphone`)
- **PocketSphinx** (required only if you need to use the Sphinx recognizer, `recognizer_instance.recognize_sphinx`)
- **Google API Client Library for Python** (required only if you need to use the Google Cloud Speech API, `recognizer_instance.recognize_google_cloud`)
- **FLAC encoder** (required only if the system is not x86-based Windows/Linux/OS X)

The following requirements are optional, but can improve or extend functionality in some situations:

- On Python 2, and only on Python 2, some functions (like `recognizer_instance.recognize_bing`) will run slower if you have `Monotonic for Python 2` installed.
- If using CMU Sphinx, you may want to **install additional language packs** to support languages like International French and Chinese.

The following sections go over the details of each requirement.

**Python**

The first software requirement is **Python 2.6, 2.7, or Python 3.3+**. This is required to use the library.
PocketSphinx is a lightweight speech recognition engine, specifically tuned for handheld and mobile devices, though it works equally well on the desktop.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>Updated man files</td>
<td>a year ago</td>
</tr>
<tr>
<td>include</td>
<td>More sane defaults for pocketsphinx and android demo</td>
<td>6 months ago</td>
</tr>
<tr>
<td>m4</td>
<td>More accurate check for python</td>
<td>2 years ago</td>
</tr>
<tr>
<td>model</td>
<td>Updated the en-us lm to en-us-70k-0.2-pruned</td>
<td>11 months ago</td>
</tr>
<tr>
<td>src</td>
<td>Merge pull request #66 from tedlefter/master</td>
<td>6 months ago</td>
</tr>
<tr>
<td>swig</td>
<td>Added threading support</td>
<td>3 months ago</td>
</tr>
<tr>
<td>test</td>
<td>Restored large vocabulary test</td>
<td>10 months ago</td>
</tr>
<tr>
<td>win32</td>
<td>Introduce SPHINX_DLL macros to enable DLL compilation. Compile without...</td>
<td>a year ago</td>
</tr>
</tbody>
</table>
include  More sane defaults for pocketsphinx and android demo
m4      More accurate check for python
model   Updated the en-us lm to en-us-70k-0.2-pruned
src     Merge pull request #66 from tedlefter/master
swig    Added threading support
test    Restored large vocabulary test
win32   Introduce SPHINX_DLL macros to enable DLL compilation. Compile without...
AUTHORS Removed outdated files, updated version
LICENSE Included license file in tarball
NEWS    Removes autogenerated files. Adds news for release
README  Fix fenced code blocks.
README.md Add README.md to render it right on GitHub.
autogen.sh Renamed COPYING to LICENSE
configure.ac Use config.h, no clutter in command line
indent.sh Add an indent.sh
pocketsphinx.pc.in Swig updates for javascript
pocketsphinx.sln Windows fixes

README.md
Package Manager
(From Wikipedia, the free encyclopedia)

A collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner.

A package manager deals with packages, distributions of software and data in archive files. Packages contain metadata, such as the software's name, description of its purpose, version number, vendor, and a list of dependencies necessary for the software to run properly. Upon installation, metadata is stored in a local package database. Package managers typically maintain a database of software dependencies and version information to prevent software mismatches and missing prerequisites. They work closely with software repositories, binary repository managers, and app stores.
Demos
Mac II Demo

https://www.youtube.com/watch?v=8bepzUM1x3w
Requirements of your demo:

- 4 minutes long
- show your code working
- show how you used **either**:
  - dictionaries,
  - lists,
  - modules,
  - libraries
- file i/o
Homework 10:

Package your code in a zip, with a readme.txt file that explains how to run it
Readme

A README file contains information about other files in a directory or archive of computer software. A form of documentation, it is usually a simple plain text file called READ.ME, README.TXT, README.md (for a text file using markdown markup), README.1ST – or simply README.

The contents typically include one or more of the following:

- Configuration instructions
- Installation instructions
- Operating instructions
- A file manifest (list of files included)
- Copyright and licensing information
- Contact information for the distributor or programmer
  - Known bugs
  - Troubleshooting
- Credits and acknowledgments
- A changelog (usually for programmers)
- A news section (usually for users)
Final Prep
Agenda

• Exam format & resources
• Exam topics
• Sample problems
• Course evaluations
Final Exam Format

• 1:50 long

• Definitions

• Trace the output and short answer questions (approximately 40% of the exam)

• Writing code (approximately 60% of the exam)
# Python Command Index

<table>
<thead>
<tr>
<th>Core Language Elements and Functions</th>
<th>Module Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>random.randint</td>
</tr>
<tr>
<td>chr</td>
<td></td>
</tr>
<tr>
<td>def</td>
<td></td>
</tr>
<tr>
<td>del</td>
<td></td>
</tr>
<tr>
<td>elif</td>
<td></td>
</tr>
<tr>
<td>else</td>
<td></td>
</tr>
<tr>
<td>except</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td></td>
</tr>
<tr>
<td>for</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td></td>
</tr>
<tr>
<td>global</td>
<td></td>
</tr>
<tr>
<td>if</td>
<td></td>
</tr>
<tr>
<td>import</td>
<td></td>
</tr>
<tr>
<td>in</td>
<td></td>
</tr>
<tr>
<td>input</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td></td>
</tr>
<tr>
<td>max</td>
<td></td>
</tr>
<tr>
<td>min</td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
</tr>
<tr>
<td>open</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>ord</td>
<td></td>
</tr>
<tr>
<td>print</td>
<td></td>
</tr>
<tr>
<td>range</td>
<td></td>
</tr>
<tr>
<td>return</td>
<td></td>
</tr>
<tr>
<td>str</td>
<td></td>
</tr>
<tr>
<td>str.lower</td>
<td></td>
</tr>
<tr>
<td>str.upper</td>
<td></td>
</tr>
<tr>
<td>try</td>
<td></td>
</tr>
<tr>
<td>while</td>
<td></td>
</tr>
<tr>
<td><strong>File Object Methods</strong></td>
<td></td>
</tr>
<tr>
<td>close</td>
<td></td>
</tr>
<tr>
<td>read</td>
<td></td>
</tr>
<tr>
<td>readlines</td>
<td></td>
</tr>
<tr>
<td>write</td>
<td></td>
</tr>
<tr>
<td><strong>List Methods</strong></td>
<td></td>
</tr>
<tr>
<td>append</td>
<td></td>
</tr>
<tr>
<td>index</td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td></td>
</tr>
<tr>
<td>remove</td>
<td></td>
</tr>
<tr>
<td>reverse</td>
<td></td>
</tr>
<tr>
<td>sort</td>
<td></td>
</tr>
<tr>
<td><strong>String Methods</strong></td>
<td></td>
</tr>
<tr>
<td>rstrip</td>
<td></td>
</tr>
<tr>
<td>split</td>
<td></td>
</tr>
<tr>
<td>isalpha</td>
<td></td>
</tr>
<tr>
<td>isdigit</td>
<td></td>
</tr>
<tr>
<td>islower</td>
<td></td>
</tr>
<tr>
<td>isupper</td>
<td></td>
</tr>
<tr>
<td>isspace</td>
<td></td>
</tr>
<tr>
<td>isalnum</td>
<td></td>
</tr>
<tr>
<td><strong>Dictionary Methods</strong></td>
<td></td>
</tr>
<tr>
<td>clear</td>
<td></td>
</tr>
<tr>
<td>keys</td>
<td></td>
</tr>
<tr>
<td>values</td>
<td></td>
</tr>
<tr>
<td>items</td>
<td></td>
</tr>
</tbody>
</table>
### Simple ASCII

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>Null</td>
<td>32</td>
<td>20</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>$</td>
<td>96</td>
<td>60</td>
<td>'</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Start of heading</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>Start of text</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>End of text</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>End of transmit</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>Enquiry</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>Acknowledge</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>Audible bell</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>Backspace</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>Horizontal tab</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>Line feed</td>
<td>42</td>
<td>2A</td>
<td>*</td>
<td>74</td>
<td>4A</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>Vertical tab</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Form feed</td>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Carriage return</td>
<td>45</td>
<td>2D</td>
<td>–</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>Shift out</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Shift in</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Data link escape</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Device control 1</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Device control 2</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>Device control 3</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Device control 4</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>Neg. acknowledge</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>Synchronous idle</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>End trans. block</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Cancel</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>End of medium</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>Substitution</td>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>Escape</td>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>[</td>
<td>123</td>
<td>7B</td>
<td>(</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>File separator</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td>)</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>Group separator</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>]</td>
<td>125</td>
<td>7D</td>
<td>}</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>Record separator</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>~</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>Unit separator</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>`</td>
<td>127</td>
<td>7F</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
Topics & Review
Problems
- feature
- software versioning
- throw and catch
- delimiter
- exception
- binary file
- return
- hard coding data
- traceback
- exception handling
- dictionary
- two-dimensional array
9. You have just been hired by a large university to write some automated test score software. Your job is to compute the average (mean) grade for all students using data obtained from a single String that has been formatted in a very specific way. Here is a sample String:

```plaintext
rawdata = "s1,95:s2,100:s3,76:s1,98:s4,0:s5,83:s2,89"
```

Students and their scores on a single test are organized using commas (i.e. “s1,95” represents that student “s1” earned a “95”). Many students exist within this String, and individual student records are separated by colons (i.e. “s1,95:s2,100” indicates that student “s1” earned 95 points and student “s2” earned 100 points).

Your job is to write a program that analyzes a String provided in this format to determine the average (mean) grade for all students along with the total # of tests that the student has taken. You can assume the following:

- A String “rawdata” is already available in your program
- This String is formatted using the specification described above
- There are no errors in this String (i.e. You will always find an Integer as a score value and the String will always correctly use “:” and “,” characters to separate data elements correctly)

Here is a sample running of your program.

```
s1 96.5 - 2 test(s)  
s2 94.5 - 2 test(s)  
s3 76.0 - 1 test(s)  
s4 0.0  - 1 test(s)  
s5 83.0 - 1 test(s)  
```
10. Imagine that you are programming in a very old version of Python that doesn’t have access to any of the list “methods” that we have been using in class. Your job is to write a series of **FUNCTIONS** that perform these tasks. Note that for you are not allowed to use any list methods to solve any of these problems, nor are you allowed to use built-in Python functions that work with lists, such as “max”, “min” or “in”. You can use any standard Python syntax though (for loops, while loops, +, +=, etc)

For this question you only need to write 3 of the 5 functions listed below. You can write all 5 for extra credit (if you have time)

# function: my_max  
# input: a list  
# processing: finds the largest item in the list  
# (WITHOUT USING THE “max” FUNCTION!)  
# output: returns the largest item in the list

# sample code
test = [5,3,2,9,7,1,3]
biggest = my_max(test)  # biggest = 9
# function:    find
# input:      a list and a desired item
# processing: finds the location of the desired item in the list
#             (WITHOUT USING THE “index” METHOD!)
# output:     returns the location of the desired item
#             returns -1 if the item does not exist

# sample code
test = [5,3,2,9,7,1,3]
location1 = find(test, 3) # location1 = 1
location2 = find(test, 20) # location2 = -1
# function: append
# input: a list and a desired item
# processing: adds the desired item to the end of the list
# (WITHOUT USING THE “append” METHOD!)
# output: returns a new list with the appended item

# sample code
test = [5,3,2,9,7,1,3]
newlist1 = append(test, "happy")  # newlist1 = [5,3,2,9,7,1,3,"happy"]
print (test)  # test hasn’t changed

# function: can_be_found_in
# input: a list and a desired item
# processing: reports whether the desired item exists in the list
# (WITHOUT USING THE “index” METHOD OR THE “in” KEYWORD!)
# output: returns True or False

# sample code
test = [5,3,2,9,7,1,3]
isthere = can_be_found_in(test, 2)  # isthere = True
# function:   insert
# input:      a list, a desired item and a location
# processing: adds the desired item at the specified location
#             (WITHOUT USING THE "insert" METHOD!)
# output:     returns a new list

# sample code
test = [5,3,2,9,7,1,3]
newlist2 = insert(test, "foo", 3) # newlist2 = [5,3,2,'foo',
                                    # 9,7,1,3]
print (test)                     # test hasn’t changed
11. You’ve been asked to write a very simple artificial intelligence “chat bot” that can be used to simulate conversation with a user. The “chat bot” works by repeatedly asking the user to enter in a line of text. It then analyzes the line to see if it can pick up any keywords that it can use to respond to the user.

Here is a dictionary that contains all of the keywords that the “chat bot” knows about:

```python
responses = {
    "angry":["Tell me more.", "Why does that make you angry?"],
    "sad":["I'm sorry to hear that.", "What can I do to help?"],
    "happy":["That's great!", "Tell me more.", "That's wonderful."]
}
```

If the “chat bot” cannot find a keyword in a line of text it should respond with the String “I don’t know how to respond to that”. Note that for this program you can assume that the user will always enter text without any punctuation.

Here’s a sample running of the program:

Tell me your problems, (q) to quit: i am so angry
Tell me more ...

Tell me your problems, (q) to quit: i am really angry
Why does that make you angry?

Tell me your problems, (q) to quit: because i am sad that class is over
What can I do to help?

Tell me your problems, (q) to quit: i want to keep programming in python
I don't know how to respond to that.

Tell me your problems, (q) to quit: i will be happy next semester
That's great!

Tell me your problems, (q) to quit: ☹️
Evaluations