Homework #2 Data Structures — Due: Feb. 6 (Saturday at 11:59 PM)

Objects: Practice designing a program and "reviewing" Python file and dictionary usage. (Use the Python Summary and the **example programs found** at: www.cs.uni.edu/~fienup/cs1520s16/homework/example_programs_hw2.zip)

File Encryption/Decryption Program: You are to **design** and write a program that allows a user to interactively select between encrypting or decrypting files:

- encrypting a user-specified text-file (`.txt` extension) using a simple Morse-code encryption scheme to generate a new encoded file with the same name, except with a `.zzz` extension. The Morse-code encryption scheme substitutes most characters in a message by a series of dots and dashes according to the below table. (see example programs: `pickle_load_example.py` and `pickle_dump_example.py`)

Character	Code	Character	Code	Character	Code	Character	Code
. period	.-.-.-	0	-----	C	-.-.	O	---
, comma	--..--	1	.----	D	-..	P	.--.
? question mark	..--..	2	..---	E	.	Q	--.-
: colon	--..--	3	...--	F	..-.	R	.-.
@ at sign	.--.-.	4	....-	G	--.	S	...
/ slash	-..-.	5	.....	H	....	T	-
- hyphen	-....-	6	-....	I	..	U	..-
′ apostrophe	.-----.	7	--...	J	.---	V	...-
″ quotation	.-..-.	8	---..	K	-.-	W	.--
= equal sign	-...-	9	----.	L	.-..	X	-..-
end of message	.-.-.	A	.-	M	--	Y	-.--
new line	.-.-	B	-...	N	-.	Z	--..

Additionally, 3 spaces occur between each character and 7 spaces occur between words. For example, the lines of text "Meet by Union?\nBob\n" is encoded as the following string (with spaces shown as '~~b~~'):
"--~~bbb~~.~~bbb~~.~~bbb~~-~~bbbbbbb~~-...~~bbb~~-.--~~bbbbbbb~~ ..-~~bbb~~-.~~bbb~~..~~bbb~~---~~bbb~~-.~~bbb~~..--..
~~bbbbbbb~~.-.-~~bbbbbbb~~-...~~bbb~~---~~bbb~~-...~~bbbbbbb~~.-.-~~bbbbbbb~~.-.-."
Any characters not in the above table are dropped and not encoded.

- decrypting a user-specified Morse-code file (`.zzz` extension) to generate a new text-file with the same name, except with a `.txt` extension.

Your program also needs to: (See www.cs.uni.edu/~fienup/cs1520s16/homeworks/example_programs_hw2.zip)

- validate a correct selection whether to encrypt, decrypt, or exit.
- validate that the user specified file to encrypt/decrypt exists and force the user to reenter until they specify an existing file. Feel free to make the program more user-friendly by listing all files of the appropriate type `.txt` or `.zzz` extensions.
- check if the user specified file name for the encoded (or decoded) file exists before opening it for write. If it already exists, ask the user if they are okay with it being wiped out. If they are not, ask them to pick a different file name to receive the encoded (or decoded) text.

Your program's interaction should looks something like: (Student input shown in **bold**.)

```
Welcome to the Morse-code Encryption/Decryption Program

Would you like (e)ncrypt a file, (d)ecrypt a file, or e(x)it (enter e, d, or x)? encrypt
Sorry, that's an invalid choice.  Please enter only e, d, or x: e

Enter the text-file name to encrypt: mesage.txt
Sorry the file 'mesage.txt' does NOT exist -- please try again!
Enter the text-file name to encrypt: messaeg.txt
Sorry the file 'messaeg.txt' does NOT exist--please try again!
Enter the text-file name to encrypt: message.txt

WARNING:  The file 'message.zzz' already exists!
Is it okay to wipe it out (y/n)?  n
Enter the file name that should be used (.zzz extension will automatically be added): newFile

The file 'message.txt' was successfully encrypted to Morse-code and saved in the file 'newFile.zzz'
```

Save your program in a file called morseCode.py

When you write your program, be sure you:

- think about the functional-decomposition (top-down) design **before you start to write code**! You will need to turn in a design document (see below for details).
- use meaningful variable names with good style (i.e., useCamelCase)
- use comments (""" Multi-line Comment """) at the start of the program **and** immediately after each function `def`inition describing what they do (see lab1 `diceOutcomes.py` program)
- use a main function (see lab1 `diceOutcomes.py` program) located at the top of program with a call to it at the bottom to start execution
- use global constants where appropriate with good style (ALL_CAPS_AND_UNDERSCORES). (Put your global constants after your initial comments describing the program and before your main function definition so they can be found and changed easily in future versions of your program.)
- allow the user to enter the file name to be analyzed and verify that the file name exsists (`import os.path`)

Submit a single zipped file called hw2.zip containing the following:

- **morseCode.py** (your Python program)
- **design.doc** (or design.pdf, or design.txt, or design.rtf) a document describing the design of your program including a functional-decomposition diagram with text describing each function (see lab1 description)
- morseCodeObjects.p (file containing the pickled versions of the morse-code dictionaries)

The steps for the homework submission system are:

1. Design, write, debug, and test your program in the hw2 folder. When you are ready to submit your homework, zip the whole folder by right-clicking on it and selecting `Send to | Compressed (zipped) folder`. This will create a new file called `hw2.zip` which you will submit electronically. (This assumes Windows OS....)
2. Log on to the submission system at: **https://www.cs.uni.edu/~schafer/submit/which_course.cgi**

   (It is very likely that you will get some security certificate warnings when trying to use this. You may add an exception and accept the existing security certificate.) Use the same CatID user-name and password you use to log on the lab computers.
3. Select the course and section number of "CS 1520, Data Structures, Fienup". Click the "Continue".
4. Select the homework that you wish to submit: "HW 2: Morse Code". Click the "Continue" button.
5. Specify how many extra files you want to submit. Just leave it at 0. Click the "Continue" button.
6. Upload your program by Browsing and selecting your `hw2.zip` file. Click the "Continue" button.
7. The next page reports on the status of the upload(s). You can always continue to upload a better version of the program until the deadline. The newer file will replace an older file of the same name.

(If you miss the deadline, you'll need to submit it as above, but select "Late Homeworks" in step 4 above.)