CS3101-1 Python, Fall 2014: Problem Set 5

Daniel Bauer

Total points: 20
Due date: Oct 14th, 11:59pm

Submission instructions:

Place the files for all problems in a directory named `[your_uni]_week[`X`]`, where X is the number of the problem set. For instance if your uni is xy1234 and you are submitting the problem set for the first week, the directory should be called `xy1234_week1`. Either zip or tar and gzip the directory (using `tar -c xy1234_week1 | gzip > xy1234_week1.tgz`) and upload it to your Drop Box on the Courseworks page for this class.

Please pay attention to the general guidelines/homework policy on the course website.

Part1 (6 points) - Searching the File System

You are a Computer Forensics expert with the NYPD. You have been tasked with searching through a suspect's computer for any incriminating evidence that he was selling marijuana.

Use the 'os' and 'os.path' module to write a Python program ('part1.py') that will recursively search through a directory hierarchy and print out the absolute paths of all files that have a filename related to marijuana. Such a filename contains one of the words 'marijuana, marihuana, cannabis, weed' or both the words 'mary' and 'jane' together.

The program should take the name of the base directory (i.e. where to start the search) as a command-line argument.

Make sure your string comparisons are case-insensitive.

Part 2 (8 points) - Object Serialization and Command Line Arguments

In this problem, we will use the 'pickle' module to save and load secret messages hidden in inconspicuous Python objects.

(a) Write a module 'gen_key', that contains a class Key with base class dict. This class represents a simple substitution cipher. When Key instances are initialized they should randomly map each of the 26 English letters [A-Z] and whitespace to a unique ciphertext number between 0 and 26[1].

For instance a Key instance could encode the following mapping:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
0	24	18	23	2	15	26	14	10	13	1	11	17	4	21	22	20	6	3	19	12	8	16	9	7	25	5

Write a main function that creates a Key instance and uses pickle (or cPickle) to save it to a file (specified on the command line).

Note that all modules that will unpickle a Key will need to import the Key class from the 'gen_key' module.

[1] Look at the `shuffle(sequence)` function in the 'random' module. Look at the built-in function `zip(seq1, seq2)` to do this in two lines.

(b) Write a module 'encode', that contains a function encode, which takes as parameters a plaintext message and a key instance and encodes the message in the following simple unary representation. The message is represented as a list and each element of the list represents a letter. Each letter is a list of objects of any type, such that the length of the list represents the ciphertext number for the letter (according to the key).

For instance, the message 'MARY' encoded with the key above would look like this:

```
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 [],
 [1, 1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1, 1, 1]]
```

Write a main function that reads in a plaintext message from the keyboard, uses encode to encode the message, and then uses pickle (or cPickle) to save the encrypted message into a file. Output file and Key should be passed as command line arguments.

(c) Write a module 'decode', that contains a function decode, which takes as parameters an encoded message object, of the type described above, and a Key object, and returns a plaintext string. Write a main function that unplickes a message object and a Key object from a file, uses decode to decode the message and prints the plaintext. Input file and the pickle file containing the Key should be passed as command line arguments.

Part 3 (6 points) - Exceptions

There are two problem-specific sources of errors in the encoder and decoder from Part 2.

(a) When encoding a message, the plaintext may contain symbols other than 26 letters or space.

In the 'encode' module, create an exception class EncodingException (choose any base class you find appropriate) and raise an exception on this type in the encode function when an input symbol cannot be encoded.

Modify the main method to handle this exception (by asking the user to revise his input until a valid input was read).

(b) The object to be decoded is not a valid encrypted message.

This can be case for a number of reasons.

1. The message object is not a list at all. In this case the main method of the 'decode' module should handle the TypeError, so that the program prints an error message and terminates gracefully.
2. The outer list contains a non-list element. This cases you should handle the TypeError by just skipping the encoded symbol.
3. There are too many elements in one of the inner lists. Handle this case like case 2.

Implement this behavior and provide examples for objects causing each type of exception.