CSE 231 **Fall 2015**

Computer Project #4

Assignment Overview

This assignment focuses on the design, implementation and testing of a Python program that uses character strings for data decompression.

It is worth 45 points (4.5% of your course grade) and must be completed no later than 11:59 PM on Monday, October 12.

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj04.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **handin system** before the project deadline.

Assignment Background

We use data compression all the time: music is compressed by the MP3 algorithm, videos are compressed by the MP4 algorithm, and photos are compressed by the JPEG algorithm. A fundamental part of compression is to remove repeated patterns and replace them with a code. For example, "compress" was used repeatedly in this paragraph and could be replaced with a code (a pair of numbers). If you do that often enough, you can save a lot of space.

Consider the following traditional ditty:

Pease porridge hot,
Pease porridge cold,
Pease porridge in the pot,
Nine days old.

Some like it hot,
Some like it cold,
Some like it in the pot,
Nine days old.

Using a relatively simple algorithm, that compresses to the string:

```
"Pease porridge hot,\(20,15)cold,\(21,15)in the p(48,4)Nine days
(42,3).\\Some like it(82,6)(18,13)(80,6)(19,13)(78,26)"
```

Here, we are using the backslash character (`"\"`) to represent a carriage return (or new line character) and a pair of numbers to represent text that is repeated from earlier in the string: the first number tells you how far to go back and the second tells you how many characters are repeated from that starting point. You decompress the string by replacing each pair, in its turn, with a substring from earlier in the decompressed part of the string. Finally, you replace each backslash with a newline (`"\n"`).

For example, the pair (20, 15) is to be replaced by the substring obtained by going back 20 characters and copying 15 characters starting at that point. Going back 20 characters takes you to the beginning and then you copy 15 characters to get the substring `"Pease porridge "`. Replacing the pair with this substring results in the (partially decompressed) string:

```
"Pease porridge hot,\Pease porridge cold,\(21,15)…"
```

(To save space, we have used ellipses (…) to represent the rest of the compressed string.)

For the next pair, you go back 21 characters to the beginning of the second line where you again copy 15 characters, i.e. `"Pease porridge ".` Replacing the pair with this substring yields:

```
"Pease porridge hot,\Pease porridge cold,\Pease porridge in the
p(48,4)…"
```

Next, the pair (48,4) takes you back 48 characters to the `"o"` in `"hot"`. The substring obtained by copying four characters is `"ot,\"`. Replacing the pair with this substring gives you:

```
"Pease porridge hot,\Pease porridge cold,\Pease porridge in the
pot,\Nine days (42,3)…"
```

The pair (42,3) takes you back 42 characters to pick up the 3 characters `"old"(`from `"cold,"`). The decompressed string so far is:

```
"Pease porridge hot,\Pease porridge cold,\Pease porridge in the
pot,\Nine days old.\\Some like it(82,6)…"
```

And so on.

Last week Google announced a new compression algorithm named Brotli. So what? For one, it is 26% better than existing algorithms. Six years ago, Google's disk space was estimated at an exabyte (10^{18} bytes = 1,000,000 terabytes). The improved compression will save Google a quarter of that space, or about 250,000 terabytes. Since terabyte disk drives cost about $100 each, the better compression will save Google approximately $25 million. The savings might be much larger now since Google's disk space has certainly grown in the past six years.

Assignment Specifications

You will develop a Python program that allows the user to decompress any number of strings.

The program will prompt for a string to decompress. As long as the user inputs a non-empty string, the program will output the decompressed string. The decompressed string will have proper carriage returns that replace the backslashes in the input string.

When the user presses the enter/return key without first inputting a string, the program will display an appropriate termination message and halt.

The decompression algorithm looks for a left parenthesis `"("`. Note that the string method `find()` is useful to locate a specific character, such as a left parenthesis. Then, use `find`() to locate the next comma and grab the **slice** in between to extract the substring representing the first number (which, of course, needs to be converted to an integer). Use `find()` to extract the second number in a similar way. Now that you have the two numbers, work backwards in the string and use **slicing** to obtain the substring you need to copy. Repeat.

We will make two simplifying assumptions: the only parentheses and backslashes in the compressed string are for encoding (i.e. the original string does not contain any parentheses or backslashes).

Assignment Notes

1. To clarify the project specifications, sample output is provided at the end of this document.

2. The coding standard for CSE 231 is posted on the course website:

   http://www.cse.msu.edu/~cse231/General/coding.standard.html

3. Items 1-7 of the Coding Standard will be enforced for this project.

4. A useful pattern when working with strings is to start with an empty string and add characters to it — something that works nicely here to build the decompressed string.

5. Slicing is useful in multiple places in this assignment. For example, it is useful for extracting the string to be copied once you have extracted the pair of encoding numbers.

6. To simplify your program we will assume that parentheses are **only** used for encoding. That is, if you find a parenthesis, you know that you have an encoding string of the format `(first_number,second_number)`. In addition, there will be no spaces within the parentheses of the encoding string.

7. A useful first test case (from the famous soliloquy in Shakespeare's *Hamlet*) has only one substitution and is short enough to not have any carriage returns:

   ```
   to be, or not (14,6)
   ```

8. There are two different approaches to solving this problem:

   a. One uses a `while` statement for the main loop, using `find()` to look for indices of the pair of numbers within the encoding ordered pair. One thing to consider is when to stop the loop—how do you know when you don't have any left parentheses left?

   b. Using a `for` statement as the main loop is possible, but the logic is messier because you need Booleans to keep track of when you are gathering characters for the two numbers in the encoding ordered pair as well as a Boolean that keeps track of when you are working on extracting numbers so you recognize that comma as separating the numbers.

9. Use the following symbolic constant for the backslash character. It is a double backslash because a backslash has a special meaning (for example the `\n` used for carriage return). Therefore, Python will interpret the double backslash as a single backslash.

   ```
   BACKSLASH = "\\"
   ```

Suggested Procedure

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step can be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python statements, you must work on your own.

- Cycle through the following steps to incrementally develop your program:

  - Edit your program to add new capabilities one at a time.

  - Using the phrase from Hamlet's soliloquy is a good way to begin. In fact, you might start with it as a string defined in your program so you don't have to keep inputting it.

  - Use a divide-and-conquer approach to solving the problem. For example, start by simply finding and printing the first number of the first encoding pair—print it. Then find the second number of the first encoding pair—print it. Then use those two numbers to extract the string to be copied and print it. Then modify your program to handle the phrase from Hamlet's soliloquy. Get that working before trying to handle more complex cases.

  - Use the **handin** system to submit the current version of your program.

- Be sure to use the **handin** system to submit the final version of your program.

- Be sure to log out when you leave the room, if you're working in a public lab.

The last version of your solution is the program which will be graded by your TA.

*You should use the **handin** system to back up your partial solutions, especially if you are working close to the project deadline. That is the easiest way to ensure that you won't lose significant portions of your work if your machine fails or there are other last-minute problems.*

*You would also be wise to save a copy of your completed program in your CSE file space (the H: drive on the lab machines) **before** the project deadline. If you write your program at home and turn it in from home, you should copy it to your CSE file space **before** the deadline.*

*In case of problems with electronic submission, an archived copy in the **handin** system or the CSE file space is the only acceptable evidence of completion.*

Sample Output

```
Enter a string to decompress (or return to quit): To be or not to(13,3)

The decompressed string prints as:

To be or not to be

Enter a string to decompress (or return to quit):
 Pease porridge hot,\(20,15)cold,\(21,15)in the p(48,4)Nine days (42,3).\\Some like it(82,6)(18,13)(80,6)(19,13)(78,26)

The decompressed string prints as:

Pease porridge hot,
Pease porridge cold,
Pease porridge in the pot,
Nine days old.

Some like it hot,
Some like it cold,
Some like it in the pot,
Nine days old.

Enter a string to decompress (or return to quit):
 The Rain\------(6,6)\Pitter pa(7,4)\(14,14)Lis(7,2)n to t(60,2) r(60,4)(47,28)On (40,3) window (21,2)ne

The decompressed string prints as:

The Rain
------------
Pitter patter
Pitter patter
Listen to the rain
Pitter patter
Pitter patter
On the window pane

Enter a string to decompress (or return to quit):

Nothing to decompress.  Exiting program.
```