CS 3423 – Systems Programming
Spring 2020
Steven O'Hara, Sam Silvestro

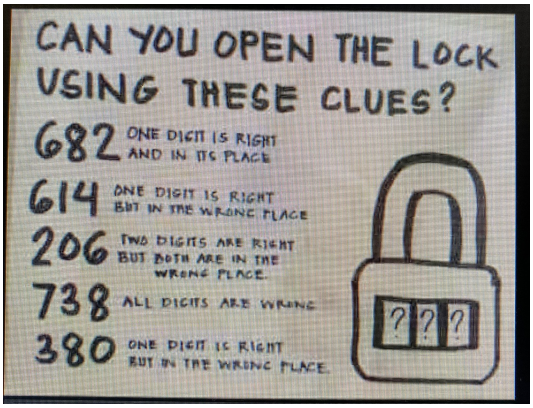
Version 1.0.1, 2/15/2020

Assignment 4

Python Combination Lock Puzzle

Note: that you may only use Python to complete this assignment. No other tools or non-standard (i.e. third-party external) Python libraries may be used to complete your work. Furthermore, as always, you are to complete this assignment in its entirety on your own; you do not have a partner in this course.

There is a popular puzzle where you try to guess the combination to a lock given some clues. For our purposes, the lock has exactly three wheels, each with values 0 through 9.



You must provide the clues on the command line and they must be in the form `XYZ-R-W` where:

- `XYZ` = a possible combination of 3 digits, each digit a member of the set $\{x \in \mathbb{Z} \mid 0 \leq x \leq 9\}$
- `R` = number of wheels that show the correct digit in the correct place within the sequence, each digit a member of the set $\{1,2,3\}$
- `W` = number of wheels that show the correct digit but in the wrong place within the sequence, each digit a member of the set $\{1,2,3\}$

The expectation is that you will use a brute force form of technique to solve this puzzle. Iterate through all 1,000 possible combinations and observe which one(s) correctly meet all specified clues. Do *not* try to solve this puzzle the way a human would, using logic.

Additional rules:

1. **"Digit reuse" is *not part* of the solution algorithm.**
    a. For example, given the rule `998-2-1` there are *no solutions*. This is because the `R=2` value indicates that two numbers are correct and in their correct locations, while the `W=1` value indicates that the one remaining number is correct but in the incorrect location. This is an impossibility, as there would only be one digit/location left, but it is apparently incorrect. Because it cannot be both correct and in the wrong position (as it is occupying the one available position remaining), this leads to the conclusion that no known solution exists.

    b. **In contrast**, *if* "digit reuse" *were* allowed (which it is *not*), this would allow for overlap between the digits specified by the `R` and `W` digits. In this case, this same problem (`998-2-1`) would indicate the existence of three solutions: 898, 988, and 999. These would be possible because we could interpret the `R=2` value to indicate that "99x", "9x8", or "x98" could each represent where the two correctly placed digits are located. Meanwhile, the `W=1` value could indicate that *either* of the two `R=2` digits could *also* be correct digits but placed in the wrong position. This would effectively allow for double or triple-counting of digits.

        For example, is we assume "99x" constitutes the two correctly placed digits, the `W=1` could indicate that one of these same two 9's is also the correct digit third digit, but is located in an incorrect position, thus allowing for the solution "999".

        Similarly, given "9x8" as a pattern built from the R=2 rule, we could assume that *either* of these 9 or 8 digits are the elusive third digit that is correct but is in an incorrect location. This leads to the solutions "999" and "988".

        The third and final solution "898" would be constructed by assuming a pattern of "x98", which is again made possible by the `R=2` rule, while allowing for the double-counting of either the 9 or the 8 as the final digit that is correct but in the wrong location. Thus, this would allow for the solutions "998" and "898".

        Combined, this process would result in a solution set of $\{898, 988, 999\}$.

        See the additional examples listed below for more information.
2. The number of clues may vary. If no clues are given, you must print an error message.
3. You *must* use a regular expression to validate the `XYZ-R-W` syntax, as defined above.
4. You are not *required* to validate the values of `XYZ`, `R`, and `W` for internal consistency (for example, whether `R+W<=3`, although you may do so if you wish.
5. You must print the list of clues given, all on a single line.
6. You must count the number of solutions found and print the numeration of each solution, on per line.
7. If no solutions are found, you must print a message to that effect, indicating there is solution.

Runtime *examples*:

```
$ python3 lock.py
ERROR: Must provide at least one pattern of the form XYZ-R-W

$ python3 lock.py 874-1-2
Trying 874-1-2
*** Solution #1 is 478
*** Solution #2 is 784
*** Solution #3 is 847

$ python3 lock.py 874-1-1 875-1-2
Trying 874-1-1 875-1-2
*** Solution #01 is 578
*** Solution #02 is 857

# This is the sample constructed from the image shown above
$ python3 lock.py 682-1-0 614-0-1 206-0-2 738-0-0 380-0-1
Trying 682-1-0 614-0-1 206-0-2 738-0-0 380-0-1
*** Solution #1 is ...
```

(there is exactly one answer, which has been hidden)

```
$ python3 lock.py 682-1-0 614-0-1 206-0-2 738-0-0 380-0-1 314-1-0
Trying 682-1-0 614-0-1 206-0-2 738-0-0 380-0-1 314-1-0
No solutions found.

$ python3 lock.py 682-0-0 614-0-0 206-0-0 738-0-0
Trying 682-0-0 614-0-0 206-0-0 738-0-0
*** Solution #1 is 555
*** Solution #2 is 559
*** Solution #3 is 595
*** Solution #4 is 599
*** Solution #5 is 955
*** Solution #6 is 959
*** Solution #7 is 995
*** Solution #8 is 999

$ python3 lock.py 874-1-1 239-0-1 110-0-0 554-0-0
Trying 874-1-1 239-0-1 110-0-0 554-0-0
*** Solution #1 is 378
*** Solution #2 is 827
*** Solution #3 is 897
*** Solution #4 is 978

python3 lock.py 874-1-1 239-0-1 110-0-0
Trying 874-1-1 239-0-1 110-0-0
*** Solution #1 is 378
*** Solution #2 is 384
*** Solution #3 is 472
*** Solution #4 is 473
```

```
*** Solution #5 is 724
*** Solution #6 is 794
*** Solution #7 is 827
*** Solution #8 is 842
*** Solution #9 is 843
*** Solution #10 is 897
*** Solution #11 is 978
*** Solution #12 is 984

$ python3 lock.py 682-1-0 614-0-1 w06-0-2 738-0-0 380-0-1
ERROR: invalid argument: w06-0-2

# The 4 is invalid, must be 0-3
$ python3 lock.py 682-1-0 614-0-1 206-0-2 738-4-0 380-0-1
ERROR: invalid argument: 738-4-0
```

Submission:

Please ensure your script's output formatting matches the syntax of the examples above as closely as possible (including spacing, which are all spaces, and no tabs present).

Submit a compressed copy of your assignment online via Blackboard. Your ZIP file, named *abc123*.zip, will include your personal *abc123* myUTSA ID and should contain only your single executable Python file.