Today's task is to write a program that creates a list of cyclops numbers. A "cyclops number" is defined as a number with an odd number of digits, with one zero, whose center digit is zero. Here are the first 32:

```
0, 101, 102, 103, 104, 105, 106, 107, 108, 109, 201, 202, 203, 204, 205, 206, 207, 208,
209, 301, 302, 303, 304, 305, 306, 307, 308, 309, 401, 402, 403, 404...
```

See how all of these numbers have a zero as the center digit? Cyclops numbers are listed on the Online Encyclopedia of Integer Sequences (http://oeis.org/A134808).

Write a program that asks the user for a starting value and an ending value and then outputs all the cyclops numbers that occur from the start up to and including the end value. Please make sure to do the following by the time you are done:

- Save your program as day29_cyclops on the Z: drive in your Python folder.
- Use a try/except structure to make sure the user enters integers.
- Make sure the second integer is larger than the first and that both integers are positive. You do this part with an if statement inside the try/except section, not with its own try/except block.
- Find and print all the cyclops numbers from the lower bound to the upper bound. Don't forget to check the upper bound, that's one of the requirements.
- Spell your prompts properly. Use variable names that make the code easy to read and understand.
- Print the numbers in 5 columns, spaced nicely using the .rjust string method.
- Make the columns work for any length numbers in the range the user enters (for example, see how 11011 is lined up with the 804 column in the sample output below.) You know the longest number won't be longer than the second integer the user enters, which helps with this.
- How can you tell a number is a cyclops number? First, turn it into a string: number = str(n). Then count how many zeros are in there using the count("0") method: number.count("0"). If there is a single zero it might be a cyclops number.
- Next check if the zero is the center character. Hmm... there can only be a center character if there are an odd number of digits, so you should check if there are an odd number of digits using the % operator and the len() command.
- If there are an odd number of digits you can access the center character in a string by slicing using square brackets like this (this code has the number we are testing named **number**, already turned into a string):

```
if number[int(len(number)/2)]=="0":
```

- For full credit you need to make your program handle any length of number.
- At the end print a count of the cyclops numbers found that also reports the starting and ending numbers checked. Something like "n cyclops numbers found from x to y"

See next page for sample output.

Sample output:

```
Please enter the starting value (a positive integer): 800
Please enter the ending value (a larger positive integer): 11012
      801      802      803      804      805
      806      807      808      809      901
      902      903      904      905      906
      907      908      909    11011    11012

20 cyclops numbers found from 800 to 11012
```

If you'd like extra credit, after you satisfy all of the above requirements print out the subset of cyclops numbers that you found that are **palindromic** cyclops numbers (http://oeis.org/A138131), that is, cyclops numbers which are palindromes (the same number forwards and backwards). For the above run, the extra note would be:

```
3 were palindromic cyclops numbers:
      808      909    11011
```

For more extra credit, also print out which ones are **prime** palindromic cyclops numbers.