CSE 231 **Spring 2016**

Computer Project 6

Assignment Overview

This assignment focuses on the design, implementation and testing of Python programs using functions.

It is worth 45 points (4.5% of course grade) and must be completed no later than 11:59 PM on Monday, February 29th (leap day!) .

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj06.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

Numbers can be interesting and since ancient times mathematicians have looked for interesting patterns. An integer n is *transposable*, if multiples of n form cyclic permutations of the digits of n. For example:

$$128205 \times 4 = 512820$$
$$142857 \times 2 = 285714$$
$$153846 \times 3 = 461538$$
$$190476 \times 4 = 761904$$

The colors are simply to show the start of the cycle. For example, notice in the first row that the number 128205 has been rotated once to the right so that in the product the digit 5 has moved from being the last (rightmost) digit to the first (leftmost) digit. In the second row the number 142857 has been rotated twice in the product. You are going to write a program that finds these special transposable numbers.

Assignment Specifications

For this particular exercise we will be making the following assumptions:

1. The number can have a single leading zero. (The sample output has some examples of such numbers.)

2. We will check if the number is *transposable* by multiplying the number by integer values from 2 to 9. If any of the multiples is a cyclic permutation of the number, then we will consider the number *transposable* and print it.

You will be provided with a text file that contains the *start* and *end* values. You need to check if the number is *transposable* for all values from `start` to `end`. The file will have one header line followed by a line with two integers separated by spaces. (Hint: you can use `find` and slicing to extract the two values.)

The program will contain the following functions:

```
rotate( string ) → string
is_transpose( string1, string2 ) → Boolean
get_transposability( number ) → None
get_transposability_zero( number ) → None
open_file()→ file object
process_file( ) → None
```

The notation above gives the name of each function, the number and type of its argument(s), and the type of its return value.

❖ `rotate( string ) → string`

- returns a new string that is created by rotating the input (`string`) by one character to the right. String slicing is your friend here.
- e.g.: 12345 → 51234

❖ `is_transpose( string1, string2 ) → Boolean`

- takes in two strings as parameters
- continually rotates string `s1` (as many times as is useful)
- returns `True` if `s1` is ever equal to `s2`, and returns `False` otherwise
- you should use your `rotate` function

❖ `get_transposability( number ) → None`

- checks if `number` is transposable by multiplying it by `2..9` and calling your `is_transpose`
- remember to convert the numbers to a strings because `is_transpose` expects strings
- if `number` is transposable, then print the number, the multiplier, and the multiple; see the sample output for formatting.

❖ `get_transposability_zero( number ) → None`

- prepends a leading zero and checks if the resulting number is transposable
- this process will be similar to that used in `get_transposability`

- if `number` is transposable, then print the number, the multiplier, and the multiple, nicely formatted as shown in the sample output.

❖ `open_file( )` → `file object`

- prompt the user for the name of the input file
- If unable to open that file, prompts again.

❖ `process_file( )` → `None`

- calls `open_file`
- extracts start and end values from file
- calls `get_transposability` and `get_transposability_zero` for all values from `start` to `end`

❖ The main part of your program will simply call `process_file`.

Assignment Notes

1. Items 1-9 of the Coding Standard will be enforced for this project.

2. The function names must be spelled exactly as shown.

3. Your program will be slower than previous programs so expect results to come slowly!

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 may 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.

- Divide-and-conquer: construct and test each function one at a time. A good ordering is the order they are described. For example, you can write the rotate function and test it completely in the IPython shell before moving on to the next function. Get help from your TA if you do not know how to do that.

- Cycle through the steps to incrementally develop your program:

    - Write and test one function at a time.
    - Test the program and fix any errors.
    - Hint: use the **handin system** to submit the current version of your solution.

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

Sample Output

```
Input file name: range.txt

Transposed numbers from 20000 to 500000

    021978 * 9 = 197802
    025641 * 4 = 102564
    032967 * 9 = 296703
    043956 * 9 = 395604
    047619 * 4 = 190476
    051282 * 4 = 205128
    054945 * 9 = 494505
    065934 * 9 = 593406
    076923 * 3 = 230769
    076923 * 4 = 307692
    076923 * 9 = 692307
    087912 * 9 = 791208
    095238 * 4 = 380952
    098901 * 9 = 890109
    102564 * 4 = 410256
    109890 * 9 = 989010
    128205 * 4 = 512820
    142857 * 2 = 285714
    142857 * 3 = 428571
    142857 * 4 = 571428
    142857 * 5 = 714285
    142857 * 6 = 857142
    153846 * 3 = 461538
    153846 * 4 = 615384
    179487 * 4 = 717948
    190476 * 4 = 761904
    205128 * 4 = 820512
    230769 * 3 = 692307
    230769 * 4 = 923076
    238095 * 4 = 952380
    285714 * 2 = 571428
    285714 * 3 = 857142
    307692 * 3 = 923076
    428571 * 2 = 857142
```