ENGI 1006 HW1 (Chapters 0-2)

Assignment Due on 09/29/2016 at 11:59PM

Academic Honesty Policy:

You are expected to submit **your own work**. Homework assignments are to be completed individually, NOT in groups. Please do not include any of your code snippets or algorithms in public Piazza posts. You must not look at solutions from any source. If you have any questions at all about this policy please ask the course staff before submitting an assignment.

For all problems:

- **Make sure to use Python 3.5 for all Python programming. To check your Python version, open a terminal window and enter "python --version" (no quotes).**

- **Use the input() function to receive input and the print() function to produce output.**

- **Do not use any built-in function such as min( ), max( ), unless you are explicitly told otherwise. You may also not use anything we haven't covered yet.**

- **Leave a comment at the top of each file explaining the logic of your code at a high level. Make in-line comments to explain specific approaches where necessary.**

- **You will be graded on your code style. Please refer to the Python style guide available on Courseworks in resources folder.**

- **For each question, submit separate files corresponding to each question, named as follows:**

- **time_convert.py**
- **sum_min_and_max.py**
- **perfect_numbers.py**
- **prime_numbers.py**
- **decimal_to_base_2.py**
- **next_highest.txt**

Put all six files into a single folder called hw1. Compress your hw1 folder into a zip file named UNI_hw1.zip (eg. abc1234_hw1.zip). Make sure to use an underline, NOT a hyphen. Submit that zip file via courseworks.

1. **time_convert (15 pts):** Write a Python script that prompts the user to enter a duration in seconds and then display that duration in "hours, minutes, seconds." Assume user's input is a positive integer. You do not need to pluralize the time units, simply add a parenthesized "s" to each one as shown in the example.

   Input: 3672
   Output: 1 hour(s) 1 min(s) 12 sec(s)

2. **sum_min_and_max (15 pts):** Write a Python script that asks user to input a series of integers, one at a time, separated by an Enter. To terminate, user will input the character "X" (capitalized). After that, the script should print the sum, minimum and maximum of all numbers entered.

   Input: 7 (press Enter) -3 (press Enter) 20 (press enter) X (press enter)
   Output: Sum: 24, Min: -3, Max: 20

3. **perfect_numbers (15 pts):** A number is perfect if it is equal to the sum of its factors (except itself but including 1). For example, 28 (with factors 1, 2, 4, 7, 14) is perfect because 28 = 1 + 2 + 4 + 7 + 14. Write a Python script to determine if a number is perfect. Make it as efficient as possible. (Hint: try to reduce the number of calculations)

   Input: 28
   Output: True

4. **prime_numbers (15 pts):** Write a Python script that checks if a number is prime or not. Make it as efficient as possible. (Hint: try to reduce the number of calculations)

   Input: 6
   Output: False

   Input: 7
   Output: True

5. **decimal_to_base_2 (20 pts)**: In class, we have learned decimal representation (base 10) and binary representation (base 2). Write a Python script that converts a non-negative integer to base 2.

   Input: 3
   Output: 11

   Input: 12
   Output: 1100

6. **next_highest (20 pts)** Describe an algorithm in English that takes as input a 9 digit number where no digit appears twice, containing the numbers 1-9 (inclusive), and produces as output an arrangement of the same 9 digits corresponding to the next highest number. If no such number exists, the algorithm should indicate this. For example, if the input is 781623954 the output would be 781624359. You can use bullets to describe your algorithm or pseudocode similar to what we saw in class. **You do NOT have to write this algorithm in Python. Your algorithm will be graded for efficiency.**