Homework 1

Intro to Programming (Term I/2017–18)

built on 2017/09/13 at 15:54:22 ***due:*** *thu sep 21 @ 11:59pm*

This assignment will introduce you to the programming environment you will be using for this class and to programming in Python, as well as to our general infrastructure. In this assignment, you will install Anaconda (or your Python IDE of choice), write a few simple Python programs, solve a number of programming puzzles, and hand them in.

Be sure to read this problem set thoroughly, especially the sections related to collaboration and the hand-in procedure.

Overview:

Problem	File Name
1.	–
2.	`age.py`
3.	`types.txt`

Problem	File Name
4.	`decipher.txt`
5.	`greater.py`
6.	`secret.txt`

Collaboration

We interpret collaboration very liberally. You may work with other students. However, each student ***must*** write up and hand in his or her assignment separately. Let us repeat: You need to write your own code. You must not look at or copy someone else's code. You need to write up answers to written problems individually. The fact that you can recreate the solution from memory will be taken as proof that you actually understood it, and you may actually be interviewed about your answers.

Be sure to indicate who you have worked with (refer to the hand-in instructions).

Logistics

We're using a script to grade your submission before any human being looks at it. Sadly, the script is not as forgiving as we are. *So, make sure you follow the instructions strictly.* It's a bad omen when the course staff has to manually recover your file because the script doesn't like it. Hence:

- Save your work in a file as described in the task description. This will be different for each task. **Do not save your file(s) with names other than specified.**
- You'll zip these files into a single file called `a1.zip` and you will upload this one zip file to Canvas before the due date.
- Before handing anything in, you should thoroughly test everything you write.
- At the beginning of each of your solution files, write down the number of hours (roughly) you spent on that particular task, and the names of the people you collaborated with as comments. As an example, each of your files should look like this:

```
# Assignment XX, Task YY
# Name: Eye Loveprograming
# Collaborators: John Nonexistent
# Time Spent: 4:00 hrs

... your real program continues here ...
```

- The course staff is here to help. We'll steer you toward solutions. Catch us in real-life or online on Canvas discussion.

Task 1: Install Anaconda and Python on Your Computer (10 points)

We will be using an IDE called Spyder in class. The easiest way to obtain Spyder and other useful packages is to install a prepackaged Python distribution. We recommend Anaconda, available from `https://www.anaconda.com/download/`. You are welcome to try other IDEs (e.g., PyCharm, Wing101).

You will want to download the distribution appropriate for your platform (Windows/Mac OS X/Linux). We're using Python 3.6 for this class; make sure you download the right version[1].

Task 2: Write a Simple Python Program (10 points)

For this task, save your work in `age.py`

The goal of this programming task is simply to get you more comfortable with using an IDE and to begin using simple elements of Python.

You will write a program that does the following, in order:

1. Ask the user to enter his/her name
2. Print out "'Hi, "' followed by name the user has entered.
3. Ask the user to enter a year of birth.
4. Print out age that is based on the user input using the simple formula (2017 – year of birth).

An example of an interaction with your program is shown below (the words printed in blue are from the computer, based on your commands; the words in black are a user's input—the colors are simply there to help you distinguish the two components):

```
Enter your name:
**Jackie
Hi, Jackie
Enter your year of birth:
**1992
Your age is 25.
```

Please stick to this script to facilitate automatic grading. Keep in mind that the user may enter names and years of birth that differ from the example run above.

Hints

To complete this task, you will need to learn to instruct the computer (1) to print out the values you specify and (2) to read input from the console. Python 3 has predefined functions for these; they are called **print** and **input**, respectively. You will also want to store your input for further processing. For this, you'll keep it in a variable. You can review these concepts and learn more about them from the following links:

- **print** — `https://cscircles.cemc.uwaterloo.ca/`
- Variables — `https://cscircles.cemc.uwaterloo.ca/1-variables/`
- **input** — `https://cscircles.cemc.uwaterloo.ca/5-input/`

[1] We used 2.7 in previous terms.

Task 3: Types and Values (10 points)

For this task, save your work in `types.txt`

Assume that we first execute the following assignment statements in Python's interactive mode:

```
width = 14
height = 11.0
```

For each of the following expressions, write the value of the expression and the type (of the value of the expression). The point of this exercise is to give you practice in predicting the type and the outcome of a Python expression, without actually running it.

1. `width/3`
2. `width/2.0`
3. `height//3`
4. `height//3.0`
5. `height*width/2`
6. `width/2*height`
7. `3 + 4 * 5`
8. `3//4 + 4*5`
9. `3/4 + 4 * 5`
10. `3/4.0 + 4 * 5`

Task 4: Decipher What's Going On (10 points)

For this task, save your work in `decipher.txt`

This problem will give you practice reasoning about Python programs. To this end, resist the urge to find answers by trying out all possibilities in Python. Instead, it is important to exercise logical reasoning to arrive at your answers (before checking them by running the code). There are two puzzles.

Program I: Find *all* settings of `x` and `y` of type **bool** that will cause this code to print `True`.

```
x = ... # bool
y = ... # bool
print (x or not y)
```

Program II: Find the *smallest* integer `x` such that the following code will print `True`.

```
x = ...
print (x >= 40 and (x%20)//10 == 1)
```

Task 5: Greater: Me or My Twisted Self (10 points)

For this task, save your work in `greater.py`

This task will get you a little more comfortable with numeric and Boolean expressions. You will write a program that does the following, in order:

1. Ask the user to enter a 4-digit integer (that is, a whole number between 1000 and 9999, inclusive). Say this number is A. We'll assume that our user is nice, so we know for fact that A really has 4 digits, no more, no less.
2. Create a new number B by twisting A: move the most-significant digit of A to the least significant spot. For example, if $A = 1425$, this step moves 1 to the end, so $B = 4251$. As another example, if $A = 9143$, this step moves 9 to the end, so $B = 1439$.
3. Print on a single line the number A, followed by a greater sign (>), followed by the number B, followed by a question mark (?), followed by True or False, indicating whether $A > B$ for real. Example: in the example above, for $A = 1425$, which gives $B = 4251$, you'll see the following text on the screen:

```
1425 > 4251 ? False
```

For further examples, an example interaction with your program is shown below (the words printed in blue are from the computer, based on your commands; the words in black are a user's input—the colors are simply there to help you tell apart the two components):

```
Enter a 4-digit integer:
**1234
1234 > 2341 ? False
```

Another example:

```
Enter a 4-digit integer:
**3147
3147 > 1473 ? True
```

(*Hint:* What happens when you twist numbers such as 4250?)

Task 6: Your Secret Code (10 points)

For this task, save your work in `secret.txt`

According to the oracle of big snakes, the following number M is magical

$$M = 15485867^{17942+4691} + 179424691^{1548+5867}$$

because both 15,485,867 and 179,424,691 are prime numbers, which are sacred in some circles. If you write it out, M has just 162,730 digits! Let's look at a few digits at the beginning and at the end of M:

$$M = 64675185142543772724 \cdots \text{middle part omitted} \cdots 31913400555894024038$$

To tap into its magic, you will follow a simple procedure (use Python to help you):

(1) First, you will need to remember your student ID number (e.g., 5784016). Determine the last 3 digits of your ID number. We'll call this k. As an example, if your number is 5784016, the last 3 digits are 016, which is numerically equivalent to $k = 16$. Last we checked, no one in this class has an ID number ending in 000, so $k > 0$.

(2) Then, you will form your secret code by locating two digits of M. To find the first digit, which we'll call a, you will examine the digits of M from left to right. Set a equal to the k-th digit of M that you encounter. The first (i.e., left-most) digit here is 6, the second digit is 4, and so on.
To determine the second digit, which we'll call b, you will examine the digits of M from right to left. Set b equal to the k-th digit of M that you encounter. The first (i.e., right-most) digit here is 8, the second digit is 3, and so on. Finally, create your secret magic code by writing down b followed by a.
Example: Continuing with $k = 16$ from above, we find that $a = 7$ (no, it's not 2) and that $b = 3$ (no, it's not 4). Hence, your secret magic code is 37.

Your Task: Determine your secret magic code using your student ID number. Document the steps you do to arrive at the answer, as well as the answer itself, in a file that you will hand in.