CSCI 141
Spring 2021
Assignment 2
Topics: Variables, Boolean logic, Conditionals

For this assignment, you will complete three programming problems. There is also an optional challenge problem at the end for those interested in a nominal amount of extra credit.

Getting Started

Review the labs and lecture slides to review. Topics needed to complete this assignment will be covered well before the deadline. As usual, seek help early if you get stuck: come talk to me or the TAs during office hours, or visit the CS mentors for help. Please keep track of approximately how much time you spend on both portions of this assignment. You will be asked to report your estimate on Canvas after you submit.

Reminder: You can discuss this homework with your peers. However, the answers to the questions and programming solution MUST be your own. You cannot copy another person's code, you cannot have another person tell you what code to type, etc. If any part of this is unclear, please come see me.

1 A Quadratic Formula Solver

Continuing from A1 on your mission to help students "check their answers" on their math homework, you are now working on a product for somewhat more advanced students. In particular, you've been tasked with writing a module that can be used to verify students' calculations involving the quadratic formula.

Write a program called `quadratic.py` that takes three floating-point numbers as command line arguments representing the coefficients a, b, and c in the quadratic equation $y = ax^2 + bx + c$, and print out the roots of the parabola represented by this equation.

It's been a while since I took algebra class, so in case you, too need a refresher: a quadratic equation of the form $y = ax^2 + bx + c$ takes the shape of a parabola; the roots of the quadratic are the values of x for which $y = 0$. The so-called "quadratic formula" for finding these roots (i.e., the values of x where the parabola intersects the x axis) is:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

How do I compute the square root of a number in Python? There are a couple ways to do it, but the one I recommend here uses a fact you may remember from math class: $\sqrt{x} = x^{(\frac{1}{2})}$. In other words, raising a number to the power of 0.5 is equivalent to taking its square root.

Some facts to remember about this formula:

- The $\pm$ symbol means "plus or minus"; that means that in general there are two roots, one of which is computed using a plus here and the other is computed using a minus.
- If a is 0, the formula would result in division by zero, which is undefined.

- If $b^2 - 4ac$ (called the *discriminant*) is negative, the result of the square root is undefined and there are no roots (actually there are roots but they are complex; for purposes of this problem let's keep it real).
- In the special case that the parabola's minimum (or maximum, if a is negative) is on the x axis, there's only one root, and the "plus" version of the formula will give the same value as the "minus" version.

Your program should behave as follows to handle the special cases described above:

- If a is 0, your program should output a message saying that the coefficients do not describe a quadratic.
- If the discriminant is negative, your program should output a message telling the user that there are no real roots.
- Otherwise, your program should print out both roots on the same line, separated by a space.
- If there's only one root, your program may simply print the same value twice. If that root is zero, it's okay if it prints one of the zeros with a negative sign[1]

A few sample runs of `quadratic.py` are shown in Figure 1.

```
>>> %Run quadratic.py 1 0 -4
  2.0 -2.0
>>> %Run quadratic.py 1 1 -4
  1.5615528128088303 -2.5615528128088303
>>> %Run quadratic.py 1 1 1
  Negative discriminant: no real roots.
>>> %Run quadratic.py 0 1 10
  a is zero: not a quadratic
>>> %Run quadratic.py -2 1 10
  -2.0 2.5
>>>
```

Figure 1: Sample outputs for `quadratic.py`. The command line arguments provided in the "Program arguments" box of Thonny are shown in gray text in the shell.

Testing

You should be sure to make sure your code behaves correctly in the following situations:

- A case where a is zero
- A case the discriminant is negative
- Cases where a is positive and negative
- A case where the same root gets printed twice.

[1] It turns out that because of how floating point numbers are stored, 0.0 and -0.0 are stored as two different numbers! Don't worry though: `0.0 == -0.0 => True`, as you would hope.

2 Flower Power Up: the game

Suppose that you are a programmer for a game development company called *FlowerPower*. The text adventure game being prepared for launch involves a character meandering through the forest, during which they find and pick up flowers.

Your task is to write code for a portion of the game in which the role-playing character encounters a perfumer who wants to exchange some of the gathered flowers for emeralds. The perfumer exchanges flowers for emeralds according to her secret formula (explained below).

The chief game designer has given you the below pseudocode that explains the mechanics that your python program should implement. She has also instructed you to **use no more than 10 `if` and `elif` keywords** (`else` keywords are not included in this count). You are allowed to be as creative as you like with the text, as long as the numbers are correct.

- Prompt the player to specify how many stargazer lily flowers were found and picked up
- Prompt the player to specify how many plumeria flowers were found and picked up
- Include a narrative of how the player is meandering through the forest
- The perfumer asks the player how many of the stargazer lily flowers they'd like to trade
- The perfumer asks the player how many of the plumeria flowers they'd like to trade
  - If the player specifies that they want to trade more flowers (of either kind) than have been collected, the perfumer gets mad and ends the conversation (the program ends; it should not throw an error).
  - If the player specifies to trade a total of zero flowers (i.e., the sum of both flower types), the perfumer is disappointed and ends the conversation (the program ends; it should not throw an error).
  - If the player wants to trade their flowers, then the perfumer is happy and will offer emeralds according to the following exchange rules (the perfumer's secret formula):

Number Plumeria Player is Willing to Trade	Number Stargazer Lily Player is Willing to Trade	Emeralds Offered by Perfumer
Fewer than 6	Fewer than 11	Twice the number of stargazer lily offered for trade
6 or more	Fewer than 11	Three times the number of plumeria offered for trade
Fewer than 20	Multiple of 12 but NOT a multiple of 24	The number of plumeria offered for trade
20 or more	Multiple of 12 but NOT a multiple of 24	Four times the number of plumeria offered for trade
Any	A number of stargazer lily flowers different than any of the 4 above choices	Five times the number of stargazer lily offered for trade

- The perfumer should ask the player if they want to make the exchange. If the player enters `y`, `yes`, or `Yes`, the program should output the number of emeralds that the player walks

away with, as well as the number of plumeria and stargazer lily flowers that the player retains. Otherwise, the program should output the number of plumeria and stargazer lily flowers the player walks away with.

Planning your code

In order to convert the chart into code that gives the right results, you need to plan. An excellent way to begin is to try and turn the table into a flow chart. Then, where you see the same check (e.g., "Is the stargazer lily trade , 11?") in multiple places, try to re-organize to consolidate. Once you have a working flow chart you can more easily turn it into a series of nested if, else, and elif statements.

Then, in testing you can refer to the flow chart to make sure you're testing each possible branch. Your flow chart is part of your pseudocode so you can work with other students on it.

Program Arguments

To make it faster and easier to test, give yourself the option to pass in all the numbers using the program arguments. Use an `if` statement at the top of the code to check if there are more than 1 argument: `len(sys.argv) > 1`, and if so, set the main variables from the program arguments. Else, write out each prompt in full and wait for the user to enter input. You can see examples of this in Figure 2.

```
>>> %Run flower_power.py
How many stargazerlily have you picked up? 45
How many plumeria have you picked up? 12
As you meander through the forest, you round a corner and a  perfumer
    appears out of nowhere and hits you over the head with her wooden spoon. "Watch
    where you're going!" she says. She peers into your bag and her demeanor changes
    immediately. "I have emeralds I can give you for those flowers..."
How many stargazer lily are you willing to trade? 32
How many plumeria are you willing to trade? 7
The perfumer offers you 160 emeralds.
Do you accept the trade? (y / n) y
You make the exchange, and walk away with 160 emeralds,
13 stargazerlily, and 5 plumeria.
>>> %Run flower_power.py 45 12 32 7
The perfumer offers you 160 emeralds.
Do you accept the trade? (y / n) y
You make the exchange, and walk away with 160 emeralds,
13 stargazerlily, and 5 plumeria.
>>> %Run flower_power.py 20 0 0 0
The  perfumer twitches and says, "If you don't want to trade, then get out of my woods!"
```

Figure 2: Sample outputs for `flower_power.py`

Testing

For this problem, we've provided you with some test cases in the table below. Note that these sample inputs are not guaranteed to be an exhaustive test suite. Your code will be graded on a different set of tests from the ones given below, so you can't count on these tests finding all possible mistakes. You should test your program on your own combinations of inputs, making

```
>>> %Run flower_power.py 4 5 1 2
The perfumer offers you 2 emeralds.
Do you accept the trade? (y / n) n
You walk away with 4 stargazer lily
and 5 plumeria.
```

Figure 3: Sample output for turning down a trade `flower_power.py`

```
>>> %Run flower_power.py 2 4 2 5
I don't trade for imaginary flowers!
```

Figure 4: Sample output for trying to trade more than you have in `flower_power.py`

sure that you have tried out all possible paths that your code might take. In other words, make sure you try out numbers that test **every** possible scenario your program could encounter.

Stargazer Lilys Found / Willing to Trade	Plumerias Found / Willing to Trade	Perfumer Offers	Accept?	Player's Final Stargazer Lily/Plumeria/Emeralds
11/6	30/22	66 emeralds	Yes	5/8/66
100/0	40/5	0 emeralds	Yes	100/35/0
10/10	5/6	perfumer runs away	NA	NA
10/10	6/5	20 emeralds	No	10/6/0
20/0	0/0	Unwilling to trade	NA	NA
13/12	9/8	8 emeralds	Yes	1/1/8

Submission

Check over the rubric to make sure you aren't missing anything. Submit the following Python files with your solutions to each of the programming problems. Be sure the names of the files uploaded to Canvas match these exactly (if you resubmit, canvas will append a number—this is fine): `quadratic.py` and `flower_power.py`.

Finally, fill out the A2 Survey on Canvas with an estimate of the number of hours you spent working on all parts of this assignment.

Rubric

`quadratic.py` **(15 points)**

Author, date, and program description given in a comment at the top of the file	1 point
Code is commented adequately and variables are appropriately named	2 points
Correct output for $a = 0$	3 points
Correct output when discriminant is negative	3 points
Correct output when the quadratic has two distinct real roots	4 points
Correct output when the quadratic has only one real roots	2 points

`flower_power.py` **(35 points)**

Author, date, and program description given in a comment at the top of the file	1 point
The code is commented adequately and variable names are appropriately named	2 points
The program uses no more than 10 `if` and `elif` keywords	5 points
The program correctly prompts for the flower input	2 points
Program can take input via the command line OR, if no command line arguments are given, user prompts	5 points
The emerald and remaining flower counts are correct after making a trade	6 points
The remaining flower counts are correct when the trade is not made	4 points
The program responds correctly if the player specifies they want to trade 0	4 points
The program responds correctly if the player wants to trade more flowers than have been picked up	6 points
Total	50 points

Challenge Problem: Mean and Median of Three Values

Some assignments will come with an optional challenge problem. In general, these problems will be worth very small amounts of extra credit: this one is worth two points. Though the grade payoff is small, you may find them interesting to work on and test your skills in Python and algorithm development. The skills and knowledge needed to solve these problems are not intended to go beyond those needed for the base assignment, but less guidance is provided and more decisions are left up to you. The A2 challenge problem is as follows:

Write a program called `threestats.py` that takes three numbers as command line arguments and prints the mean and median of the values as `float`s. The median should be printed first, followed by the mean on a second line, as shown in the sample output in Figure 5.

Note that this program gets its input via command line arguments, not the `input` function. If you need a refresher on command line arguments, review the Lab 2 handout. In the figure, the arguments appear on the console, but I specified them by typing them into the "Program arguments" text box at the top of the Thonny window.

```
>>> %Run threestats.py 4 4 10
  Median: 4.0
  Mean: 6.0
>>>
```

Figure 5: Sample output for `threestats.py`. Note that 4 4 10 are command line arguments specified in the Program arguments box in Thonny.

As usual, you do not need to handle improper user input: assume that the command line arguments can be correctly converted to `float`s.

To get the extra credit, your program must follow some extra guidelines:

- You may not use any Python functions other than those we've covered in class, namely type conversion functions, and `print` (you shouldn't need `input`).
- You may not use loops, lists, or any other constructs we haven't covered; this problem can be solved using boolean expressions, conditional statements, and print function calls.
- There are multiple ways to go about this, and all the ones I know of look quite cryptic when written in code; be sure to include comments to help a reader of your code understand your approach.
- For full credit, make sure your code works when two or more of the numbers are equal.

Testing is left entirely up to you.

2.1 Rubric

- No credit if the program does not follow the above guidelines
- 1 point for correct output on three distinct inputs
- 1 point for correct output on inputs where two or more values are equal