CS417 Lab #12

Getting Started

Begin the lab by downloading these starting files:

- `roots1.py`
- `roots2.py`
- `roots3.py`
- `roots4.py`

Your Tasks

1. Run the program `roots1.py`. It solves a quadratic equation:

   $ax^2 + bx + c = 0$

   and finds the smaller of its real roots, using this well-known formula:

   $x = [-b - \surd(b^2 - 4ac)] / (2a)$

   Enter each of these inputs, and see the result. For inputs D, E, and F, the program will crash. Run it again.

   (Press Control-C to exit the program)

	Input	Result	Explanation
A:	1 3 1	`Root: -2.618`	smaller root
B:	1 -2 1	`Root: 1`	
C:	1 2 0	`Root: -2`	
D:	0 2 1	crashes: `division by zero`	`a` is zero
E:	1 2 3	crashes: `domain error for math.sqrt`	roots are complex
F:	1 2	crashes: `need more values to unpack`	missing `c`

---

2. Let's make the program more robust. Instead of crashing, it should simply complain, and ask for a new input. Download the next version, `roots2.py`.

   Notice that both `real_root` and `solve_quadratic` now return two values: the root, and a **success flag**.

Verify that `roots2.py` has this new behavior:

	Input	Result	Explanation
A:	1 3 1	`Root: -2.618`	smaller root
B:	1 -2 1	`Root: 1`	
C:	1 2 0	`Root: -2`	
D:	0 2 1	crashes: `division by zero`	`a` is zero
E:	1 2 3	complains, doesn't crash	roots are complex
F:	1 2	crashes: `need more values to unpack`	missing `c`

The program can handle negative discriminants, but not division by zero. It also can't handle missing coefficients.

Make these changes:

- In `real_root`, check `a`. If it is zero, `return (0, False)`, indicating a problem.
- In `solve_quadratic`, check the length of `fields`. If it's not 3, `return (0, False)`, indicating a problem.

3. Your program shouldn't crash, but it doesn't report report enough information when there is a problem. Instead of returning a boolean, let's return a **result code**. Download the next version, `roots3.py`.

   In the main function, there is an `if-elif-else` block which checks the result code, printing various messages.

   Make these changes:

   - In `real_root`, check `a`. If it is zero, `return (0, 2)`.
   - In `solve_quadratic`, check the length of `fields`. If it's not 3, `return (0, 3)`.

4. The program works, and is informative, but it is difficult to maintain. If we added another function `do_something`, it would have to return a code. Also, if `do_something` itself called another function `do_stuff`, it would need to check the returned code. A programmer would need discipline and good habits, when making changes to the code.

   Raising exceptions can help here. Download the next version, `roots4.py`.

   The program is simpler, and easier to maintain. Notice these changes:

- Both `real_root` and `solve_quadratic` simply return a value.
- In the `main` function, there is now a `try-except` block that handles all the error messages.
- In `real_root`, there is a `raise` statement that deliberately generates an error. This error is caught in the main function, so it does not crash the program.

Make these changes:

- In `real_root`, check `a`, and if it is zero, raise a `ValueError`, explaining the problem.
- In `solve_quadratic`, check the length of `fields`, and if it is too short, raise an `IndexError`, explaining the problem.

---

5. Most quadratic equations have two roots, not just one:

    - $x_1 = [-b - \surd(b^2 - 4ac)] / (2a)$
    - $x_2 = [-b + \surd(b^2 - 4ac)] / (2a)$

    Modify `real_root`:

    - compute both `x1` and `x2`.
    - instead of returning just `x1`, return a tuple with the two roots `(x1, x2)`.

---

6. [Bonus 10%] Python can handle complex numbers! They are available if you `import cmath`.

    Each complex number *a* + *bi* has two parts: *a* is the real part, and *b* the imaginary part. In python, if you know *a* and *b*, you can make a complex number thus:

    ```
    x = complex(a, b)
    ```

    Make these changes:

    - The function `real_root` should be renamed `roots`.
    - If the discriminant is negative, `roots` should return two `complex` numbers, instead of raising a `ValueError`
    - If the discriminant is zero, there is one real root. Return a tuple with *one* real value.
    - If `a` is zero, we can still solve the equation (it's actually not quadratic, so it's simpler). Return a tuple with one real value.
    - What if `a`, `b` are both zero? Is that a fatal error? If so, raise a `ValueError`.
    - What if `a`, `b`, `c` are all zero? Is that a fatal error? If so, raise a `ValueError`.

    Many things can go wrong in a program, and there are exceptions for all of them. If you search for "python predefined exceptions", you'll get this page:

`docs.python.org/3/library/exceptions.html`.

Turning your work in

To turn you work in, go to mycourses.unh.edu, find CS417 and lab #12, click the "Submit" button, and upload:

- `roots2.py`
- `roots3.py`
- `roots4.py`

At the end of the lab session, submit any work you have completed. You can submit again until midnight, with no lateness penalty.