Chapter 4

Lab 4: The Quadratic Equation and Prime Numbers

4.1 Introduction

In this lab, we will make more extensive use of conditional statements to implement algorithms which solve the quadratic equation, identify prime numbers, and add fractions. An optional challenge problem, The Lucky Number of Euler, explores how the quadratic equation can generate prime numbers.

4.2 Preparation

This lab will rely on the material from Sections 1.2.1 to 1.2.4 of the Scientific Python Lecture notes. We'll now be making frequent use of conditional statements:

```
def compare(a,b):
    if (a==b):
        print("a equals b")
    elif (a<b):
        print ("a is less than b")
    else:
        print ("a is greater than b")
```

Notice that Python uses `==` for comparison. You will get a syntax error if you use `a=b` instead.

We'll also use of the modulo operator % extensively. The modulo operation $a\%b$ returns the remainder from the integer division a/b.

```
# is b a factor of a?
def isfactor(a,b):
    if (a%b == 0):
        return True;
    return False
```

Why does `a%b == 0` mean that b is a factor of a?

△ **Jupyter Notebook Exercise 4.1:** Consider this verbose code snippet:

```
for i in range(100):
    print("on index ", i)
```

Which prints the current index on every iteration. Use the modulo operator to modify the code so that it only prints the index every 10 iterations. This is a classic trick!

We will also be using while loops, which repeat a block of code until a condition is met:

```
count=0
while(count<10):
    print(count)
    count = count+1
```

4.3 Quadratic Formula

The Quadratic equation:

$$ax^2 + bx + c = 0$$

has solutions which are given by the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4.1}$$

The number of unique real solutions depends on the quantity in the square root, which is called the discriminant:

$$b^2 - 4ac$$

If this is positive there are two real solutions, it it is one there is one real solution, and if it is negative there are no real solutions. For now, let's assume that a, b, and c are all integers.

In this case, the solution to the quadratic equation can be calculated as follows:

```
D := b^2 - 4ac
if (D=0):
    calculate the single solution from quadratic
    print single solution
if (D<0):
    print no solutions
if (D>0):
    calculate both solutions from quadratic formula
    print both solution
```

A test case with one real solution is:

$$(x-1)(x-1) = x^2 - 2x + 1.$$

A test case with two real solution is:

$$(x-1)(x+1) = x^2 - 1.$$

A test case with zero real solutions is:

$$(x-i)(x+i) = x^2 + 1.$$

△ **Jupyter Notebook Exercise 4.2:** Implement a function `quad(a,b,c)` which reports the solutions to the quadratic equation and verify it with the test cases shown.

△ **Jupyter Notebook Exercise 4.3:** Calculate one more case with integer solutions and use it to test your function more thoroughly.

△ **Jupyter Notebook Exercise 4.4:** As we discussed in a previous lab, Python can work with imaginary numbers, but you will need to import `cmath` and use `cmath.sqrt()` rather than `math.sqrt()`. Make this change and verify that it works with all the examples, including the one with no real solutions.

△ **Jupyter Notebook Exercise 4.5:** (Optional) The condition that the discriminant is exactly zero (`D==0`) is problematic when applied to floating point numbers. (Why?) In example is for a=.1 b=.3 c =0.225 which should have only one real solution. Test you function with this test case. Modify the conditionals in your function to account for floating point precision and test it.

4.4 Prime Numbers

A prime number is a number that has two and only two factors: itself and one. One is not prime, but two is. We can determine if a number a is prime as follows:

```
if (a<2):
    return false
i := 2
while (i ≤ √a):
    if (a%i=0):
        return false
    i := i + 1
return true
```

Why is there no need to check for factors larger than $\sqrt{a}$?

△ **Jupyter Notebook Exercise 4.6:** Implement a function `isprime(a)` which returns `True` if the integer a is prime and `False` if not.

Suppose that next we want to find the first n prime numbers greater than or equal to a number A. We can simply check if A, $A+1$, $A+2$, and so on are prime until we find the first n. But we do not know how many numbers we will have to check before finding n that are prime. This is a case for a `while` loop.

△ **Jupyter Notebook Exercise 4.7:** Find the first n prime numbers greater than A using a while loop and your `isprime` function. Test it for $n = 10$ and $A = 0$, then for $A = 1000000000$. Try that with paper and pencil!

Notice how we broke this problem of finding primes into two parts: determining whether or not a number is prime or not, then testing and counting prime numbers. We thoroughly tested the first part before using it in the second. This is an essential approach to solving computational problems:

break complicated tasks down into smaller tasks which can be tested separately. With that in mind...

△ **Jupyter Notebook Exercise 4.8:** Write a routine to simplify the fraction

$$\frac{a}{b}$$

as much as possible. You should return two integers as a Tuple, like this:

```
def simplify(a,b):
    # Your code
    return a_simp,b_simp
```

It should use the following algorithm

```
1   a_simp := a
2   b_simp := b
3   i := 1
4   while i less than or equal to the minimum of a_simp and b_simp:
5       if i is a factor of a_simp and b_simp:
6           a_simp :=a_simp / i
7           b_simp :=b_simp / i
```

Test this algorithm with (3,12), (51,9), and (3,11), with code like:

```
a,b = simplify(3,12)
print("{}/{}".format(a,b))
```

△ **Jupyter Notebook Exercise 4.9:** Write a function which computes the fraction

$$\frac{n}{d} = \frac{a}{b} + \frac{c}{d}$$

from integer inputs a,b,c and d and returns integers n and d. Returning $n > d$ is allowed, but n/d should be a simplified fraction with a greatest common factor of one.

```
# function which  adds fractions
def addfrac(a,b,c,d):
    n=d=0
    #your code...
    return n, d

# calling function:
n,d =addfrac(1,2,1,3)
print("{0}/{1}".format(n,d))
```

Note that you don't have to worry about simplifying the fraction until the final step, for which you can use the `simplify()` routine you wrote in the previous exercise.

4.5 The Lucky Number of Euler

Euler discovered that the formula:

$$k^2 + k + 41$$

produces prime numbers for $0 \leq k \leq 39$. Perhaps you can beat Euler at his own game!

Consider quadratics of the form:

$$k^2 + ak + b$$

For each integer value of a and b, there is a maximum number n such that the quadratic formula produces prime numbers for $0 \leq k < n$

△ **Jupyter Notebook Exercise 4.10:** (Optional) Find the the values of a and b which produce the largest number of prime numbers. Restrict yourself to $|a| \leq 1000$ and $|b| \leq 1000$.

If you do complete this optional problem, then nice work, hot shot, but remember that Euler found his without using a computer!