For this lab, you will practice writing loops. Refer to the lecture notes from Monday (the problems are similar to what we have done), and do as many problems as you can.

Problem 1. π

The following is the Leibniz formula for π:

$$\sum_{0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}.$$

Write a program (using a `for`- or a `while`- loop) to compute

$$\sum_{0}^{1000} \frac{(-1)^n}{2n+1},$$

and then print an approximation for π using the result of the computation.

See `https://en.wikipedia.org/wiki/Leibniz_formula_for_%CF%80` for details on why this works.

Problem 2. Simplifying Fractions

Write a function with the signature `simpify_fraction(n, m)` which prints the simplified version of the fraction $\frac{n}{m}$. Hint: use a similar technique to the one we used when determining whether a number is prime. You do not need to use a complicated algorithm to compute the greatest common divisor (although you certainly can do that!). For example, `simplify_fraction(3,6)` should print `1/2`, and `simplify_fraction(8, 4)` should print `2`.

Problem 3. Some more π

Now we would like to figure out how many terms in the summation above we need to add up in order to approximate π to `n` significant digits. Write a function that returns the number of the terms required to obtain an approximation of π using the Leibniz formula that agrees with the actual value of π to n significant digits (i.e., the first n digits are the same in π and in the approximation.)

The best approximation of π using a `float` is available in `math.pi` (execute `import math` to be able to use it.)

Part of your job is to figure out whether two numbers agree to n significant digits. To figure that out, for a float `x`, consider what `int(x*(10**n))` means.

Problem 4. Calendar

Only do this problem after your work has been checked.

Part (a) Tomorrow's Date

Write a function with the signature `next_day(y, m , d)` which prints the date that follows the date `y/m/d`.

Reminder:

> According to the Gregorian calendar, which is the civil calendar in use today,years evenly divisible by 4 are leap years, with the exception of centurial years that are not evenly divisible by 400. Therefore, the years 1700, 1800, 1900 and 2100 are not leap years, but 1600, 2000, and 2400 are leap years. *(Source: the US Naval Observatory website.)*

Part (b) Counting Days

Write a function that prints out, in order, all the dates between `fY/fM/fD` and `tY/tM/tD`. Using the same idea, write a function that returns the *number* of days between two dates.

Problem 5. Euclid's Algorithm

Only do this problem after your work has been checked.

A more efficient way of simplifying fractions than what we suggested for Problem 2 is Euclid's algorithm: `http://www.cut-the-knot.org/blue/Euclid.shtml`. Implement and test Euclid's algorithm, and compare the number of steps it takes to the number of steps that the more naive algorithm takes.