Activities Module 6: While-Loops

1. Use a Python while loop to find an integer that fulfills the following sets of congruences:
   $x \equiv 3 \pmod{11}, \quad x \equiv 4 \pmod{13}, \quad x \equiv 5 \pmod{17}, \quad x \equiv 6 \pmod{19}$
   Make sure that you do not forget to modify your loop variable `x`.

2. Using two nested Python for-loops and guessing that the solution is not larger than the product of the moduli (13×11), find a solution of the system of linear congruences
   $3 \times x + 4 \times y \equiv 5 \pmod{13}, \qquad 2 \times x - 3 \times y \equiv 4 \pmod{11}.$

   Hint: You will need to use nested for loops. A break-statement will only exit the innermost loop, so you are stuck with getting lots of different solutions.

3. (An ancient Chinese problem.) A band of 17 pirates stole a sack of gold coins. When they tried to divide the fortune into equal portions, three coins remained. In the ensuing brawl, one pirate was killed. The wealth was redistributed, but this time, 10 coins remained. Again, an argument arose during which one more pirate was killed. But now, the fortune was evenly distributed among the survivors. What is the smallest possible number of coins?

   [Hint: The number of coins divided by 17 gives remainder 3, divided by 16 remainder 10 and divided by 15 has remainder 0. Just try out all possible values for the number of coins starting with 1.]

4. Calculate the following sums. (The result is also given so that you can check your work). Use both for and while loops.
   1. $\sum_{\nu=0}^{10000} \frac{1}{\nu^2 + 2} = 2.07657$

   2. $\sum_{i=2}^{100} \frac{i+1}{i-1} = 109.355$

5. A person takes out a credit of 10,000 US$. The person pays a monthly interest rate of 0.4% of the loan. Each month, the person pays $200.00 back. These $100.00 serve the interest fee and the repayment of the loan. For example, the first month, the interest is 40$ ($10000 \times \frac{0.4}{100}$) and the rest (160$) is used as repayment. The next month, the outstanding loan has shrunk to 9840.00$, the interest is 39.36, and the repayment is 160.64. Write a program that prints the progress of repayment as a table, with the month, the amount of the loan still outstanding, the interest paid that month, and the repayment that month as columns. Later, we will learn how to print only two digits after the point of a floating point number.

1	9840	40.00	160.00
2	9679.36	39.36	160.64
3	9518.08	38.72	161.28
4	9356.15	38.07	161.93
5	9193.57	37.42	162.58

6. Write a program that uses *Heron's method* to determine the square root of a number *x* provided by the user such that the answer squared is fewer than 0.0001 away from *x.* *Heron's method* as you might remember starts with an initial guess of

```
guess = 1
```

and then improves the guess by setting

```
guess = 0.5*(guess + x/guess)
```

7. (Bonus Problem) We want to solve the equation $-x^6 + x^5 - x^3 + x^2 - x + 0.3 = 0$. If we plug in 0 for the expression on the left, we obtain 0.3. If we plug in 1, we obtain -0.7. We decide that there should be a solution between 0 and 1. We divide the interval between 0 and 1 in the middle and have a new point 0.5. If we evaluate the expression at 0.5, we get -0.059375. Therefore, there should be a solution between 0 and 0.5. Dividing the interval in half, we get the new midpoint 0.25. Since the value of the expression is 0.0976074, we decide that a solution should be between 0.25 and 0.5. We now evaluate at the midpoint, which is 0.375, and obtain 0.0175255. Since this is a positive number, we deduce that a solution lies between 0.375 and 0.5. The midpoint is 0.4375 and the evaluation of the expression there gives -0.020818. Therefore, a solution is between 0.375 and 0.4375. Continue this process with a Python program until the interval between the lower and the upper estimate is less than 0.0001 long. Notice that if the evaluation at the midpoint is positive, than the lower limit of the interval becomes the midpoint and otherwise the upper limit becomes the midpoint. Here is the graph of the expression.