Practice problems

Here is a collection of some relatively straightforward problems that let you practice simple nuts and bolts of programming. Each problem is intended to be a separate program.

1. Write a function `getInteger` that interactively asks the user to enter an integer, and then returns this value. This function should handle the exception that the user enters text instead. Print an appropriate error message, and let the user try again. You will need to enclose your try-catch (or try-except) code inside a loop, because you don't know in advance how many attempts the user will need before getting it right. Call this function from your main program so that you can test it. You may find this function useful in future programs!

2. This is a continuation of the previous problem. Now that you have a robust way to interactively input an integer, modify `getInteger` so that it also verifies that the number entered by the user is in the range a..b, where a and b are parameters passed to `getInteger`. If the input is out of range, you should print an error message and ask the user to try again. Also, if a > b, `getInteger` should raise an exception, because this condition would be impossible to satisfy.

3. Ask the user for an integer value, and find the sum of the digits of this number. Once you get this program working, you can add error checking that you already know how to do.

4. Use a loop to print the first 25 Fibonacci numbers. Do you need to store these values in a data structure such as an array or list?

5. Ask the user to enter a year (e.g. 1945) and tell whether this is a leap year or not.

6. Write a program that prints all of its command-line arguments. Beside each (i.e. to the left), print the index in the array, so that each command-line argument is clearly labeled.

7. Ask the user for the current time here, and output the time in Los Angeles and London. For simplicity, just use a 12 hour clock and don't worry about am/pm. If the user enters "9:01" your outputs should be 6:01 and 2:01 respectively.

8. Read an input file containing some win-loss records. On each line there will be two numbers separated by a hyphen, with no whitespace. For example,

```
17-3
1-14
5-6
8-0
12-10
```

Do not assume a certain number of digits for each number. Add up the total number of wins and losses, and print these numbers out. Also, indicate the percentage of games won, rounded to the nearest tenth of a percent.

9. Write a function that computes the quadratic formula. It should solve an equation of the form $ax^2 + bx + c = 0$. The parameters to the function are the numbers a, b, and c. Since this formula gives 2 answers, we will need to return an object or list containing both values. Call this function to find the solutions to $x^2 + 10x + 21 = 0$ and $x^2 – 3x – 3 = 0$. You may assume that the solutions will be real.

10. Let's estimate the value of π by means of a Monte Carlo simulation. Here's how to do it. Imagine that you have a square that has a side length of 2. Inside this square is a unit circle. Basically all we are going to do is throw darts onto the square. Some of the darts will lie inside the circle, and some outside. The edge of the circle is so thin that we'll ignore the possibility of landing on it. The ratio of the circle's area to the square's area is π/4.

    Ask the user how many darts to throw. For each dart, ask the computer for random real values of x and y. (x, y) will then be one of the points on the square dart board. Keep track of how many are in the circle, and output your estimate of π.

    Of course we already know the value of π, but this general technique is useful when we have more irregular shapes than a circle that we want to estimate the area of.

11. Simulate a lottery drawing. Have the computer pick 6 random numbers in the range 1-50. Ask the user to "buy" a ticket by guessing what the 6 random numbers are. Tell the user how many of the selected numbers are correct.

    Hints: You should use some kind of array or list data structure to hold both the winning numbers as well as the user's selected values. You may find it convenient to sort the lists. Both Java and Python have built-in ways to sort lists. You also need to make sure that the six winning numbers are all different! However, you may assume that the user's input will all be distinct. In testing your program, you may want to temporarily allow for cheating by printing the lottery drawing before asking the user for input.

12. Here's a 2-D array application. Let's allow the user to play a simple form of Battleship. This is a board game of 8 rows and columns. On this board there are various naval vessels: an aircraft carrier that is 5 units long, a battleship 4 units long, a submarine 3 units long, and a destroyer 2 units long.

    You can use a 2-D array, or you could use 8 strings of 8 characters apiece. Place the ships in the board. For example, your board may start out like this: board = ["…SSS.." , etc. representing that the submarine is in the first row.

    Interactively let the user pick 5 locations (row, column) to drop bombs. For each bomb, indicate if it is a hit or a miss; if a hit say which vessel has been hit.

13. Let's explore some exceptions that may occur in Python or Java. Write a program that contains the following errors. Write down the name of the exception that occurs in each case. Then, write code that catches the run-time error.
    a. Asking for an element that is beyond the end of a string
    b. Asking for an element that is beyond the end of a list or array
    c. Converting a string to an int, but the string does not contain digits
    d. Opening a file that is read-only or that does not exist
    e. Dividing by zero
    f. Taking the square root or log of a negative number

14. Suppose a text file contains a set of grades for students taking a class. All the students took three tests, and we need to write a program that can process these scores two dimensionally. First, read the file, and place the numbers in a 2-D list or array. Then, output the average score for each student, as well as the class average score on each test. Here is some sample input:
    46 96 70
    86 87 90
    94 99 100
    81 82 74
    83 65 88
    71 72 94
    85 75 51
    87 93 74
    78 78 78
    83 65 66
    79 80 82
    83 84 98

15. Phone number words. Download this list of words of the English language from my Web site: cs.furman.edu/~chealy/words2.txt. We would like to help people remember a telephone number by converting it to a word, if we can. Ask the user to enter a 7-digit phone number. Note that this means 3 digits, a hyphen, and then 4 more digits. Of course, you don't need to hyphen in your computation, but it's there in the input. Scan the file words2.txt to see if there are any 7-letter words that are equivalent to this phone number. If so, then print them out. Otherwise, try 6-letter words for the first 6 digits or last 6 digits of the user's telephone number, meanwhile keeping the remaining digit intact. The scheme for converting numbers to letters is as follows:

    2 = A, B or C
    3 = D, E or F
    4 = G, H or I
    5 = J, K or L
    6 = M, N or O
    7 = P, R or S
    8 = T, U or V
    9 = W, X or Y

    You may assume that the user's input is legitimate, and that it only contains digits 2-9. For example, we can encode the phone number 776-4726 as PROGRAM.

16. Often, it's handy to have readily available a table of values that you need for some application. Here are a couple of examples.
    a. Write a program that prints an ASCII table. In other words, for each integer from 1 to 127, print out this integer, along with its corresponding character. For example, the ASCII code for 'A' is 65. So, you should see the letter A next to the number 65. Format your table in multiple columns so that the entire table could be printed out on one sheet of paper, and be viewable entirely on the screen at once. The output needs to be in "column-major" order. In other words, the information about integer n+1 needs to appear directly underneath the information about integer n, unless integer n is at the bottom of a column, in which case integer n+1 would appear at the top of the next column.
    b. Write a program that prints the integers from 1 to 256 in both decimal (base 10) and hexadecimal (base 16) notation. Again, make sure your output is in columns so that all the output can appear on one page or one screenful, and arrange the data in column-major order. You will probably need more columns in this program than in part (a).

17. Consider the following question of probability: What is the probability that two 3-digit numbers selected at random will have a product ending in the digit 0? In a discrete structures class you learn how to solve such a problem analytically. But it would be nice to verify your answer empirically, and we can do so by simulating the problem with a computer program.

a. We can solve the problem through brute force. In other words, we can enumerate every possible case. Write a program that tries every possible combination of 3 digit numbers. For each pair, keep track of how many yield a product ending in zero. Here is some pseudocode:

```
for a = 100 to 999
   for b = 100 to 999
      if a*b ends in 0
         ++count
Divide the count by the total number of iterations
```

b. Next, we wonder if the number of digits has any bearing on the answer. Let's next consider 7-digit numbers. In this case, brute force would not be practical, because you would need 81 trillion iterations! Instead, we need to select a sample of numbers at random. One million trials should give us a sufficiently precise answer. Implement this pseudocode. Is the output the same as in part (a)? Hint: If you are working in C++ or Java, you need to use the long data type, because the product of two 7-digit numbers cannot fit inside an int.

```
for trial = 1 to 1,000,000
   a = random 7 digit number
   b = random 7 digit number
   if a*b ends in 0
      ++count
Divide the count by the total number of iterations
```

18. Let's practice with loops, functions, and making a choice among multiple possibilities. In Java, you will be using a switch statement here, but in Python you will practice the if/elif/else statement.

    Let's suppose you run a hotel. People love your hotel so much, that many want to stay for a long time. Write a program that asks the user for a check-in date and a check-out date. The program will then tell the user how many nights are in the reservation. You may assume that both dates are in the same year, and that it is not a leap year. Assume that when the user enters a date, it will be entered correctly in the format month/day, as in 9/14.

    To support your program, you should write a separate function that will return the number of days in a month. For example, if the month number is 2, the function should return 28. This function should contain a switch statement (if Java) or an if/elif/else statement (in Python). This function should be called from a loop, because the user may want to stay in the hotel for several months.

19. Here is a short program featuring loops and random numbers. Let's write a program that implements a guessing game. The computer will pick a random integer in the range 1..100. It will

then give the user 7 chances to guess this number. Each time the user's guess is incorrect, the computer should say if the guess was too high or too low. The user wins and the game immediately halts if any of the user is able to correctly guess the secret number by the 7th guess. The user loses if all 7 guesses are wrong. Why do you think we limit the number of guesses to 7?

20. Ask the user to enter a positive integer less than 4000. The program will then output this number using Roman Numerals.

    A more interesting problem is to be able to go the other way: given a number entered as Roman Numbers, write it out in ordinary base 10 notation.

21. (This exercise may take a bit longer to solve.) Write a program that counts from 1 to 999,999, and writes its output to a file. But there is a catch: your program must print both the numeral as well as the English spelling of the number. Print one number per line, with the numeral first. The numeral should be right justified with a field width of 6 (since we could have up to a 6-digit number). The spelling should be left justified. Print a space between the numeral and the spelled-out form.

    Once you have completed this program, let's do some French immersion. Create a second version that spells the numbers out in French instead of English.

    Hint: This problem will be more interesting and less tedious if you can invest some time to think of ways to reduce repetitive code in your implementation.

22. If you have not done so already, download my list of English words from cs.furman.edu/~chealy/words2.txt. We are interested in knowing which words can be typed exclusively by the left hand or exclusively by the right hand. Assume that the left hand types these fifteen letters: Q, W, E, R, T, A, S, D, F, G, Z, X, C, V, B. The right hand takes care of the other eleven letters. Let's call these words lefty and righty words, respectively. Write a program to answer these questions:
    a. List all of the lefty and righty words. How many of each are there? What percentage of all words are of each type?

    b. Find the longest lefty word(s) and the longest righty word(s). Note that there could be a tie for the longest.

    c. Eye charts only use 10 letters of the alphabet. They are: C, D, E, F, L, N, O, P, T, Z. How many words can be spelled exclusively by letters that can appear on an eye chart? What percentage of all words is this?

d. Find the longest word using only letters that can appear on an eye chart. Note that there could be a tie for the longest word.

23. Write a program that performs the long division algorithm taught to kids in elementary school. The program will ask the user for the integers representing the numerator and denominator of some fraction. Also ask for the number of desired digits of precision in the answer. The program should accept any integer number of digits. Then, the program will carry out the long division algorithm, printing digits and keeping track of the status of the divisor as it goes along. For example, the user might want the first 700 digits of 1/7.