Python
Lab 1

Work on as many programming projects as you have time to complete. There is no grade; the number and difficulty of projects is intended to enable students at different levels of programming background and skill to gain some benefit from this lab.

Coding style.

Always start with a documentation string ("docstring"). In Spyder, replace the lines between the two lines of triple quotes with a brief description of the program, followed by two lines:
Author: Your Name
Changelog: Initial version YYYY-MM-DD

Please choose variable names that are descriptive. Please do not "hard code" literals. If I ask you to write a program that works on integer N, you can set N in your code but assign it to a variable. When we learn more about I/O you will learn how to obtain input from the user, the command line, or a file, but for today it is sufficient to set the values in your script.

For Beginners

Project 1

Write a program that:
1. Creates a list of temperatures [0, 10, 20, 30, 40, 50]
2. Prints the number of items in the list
3. Prints the index of temperature 30
4. Adds another temperature 60 at the end
5. Loops through the list of temperatures and converts them from Celsius to Fahrenheit, printing each value in degrees C and F (use print C,F)

Project 2

Write a program that
1. Creates the temperature list 0 through 60 by 10 using a loop rather than by typing in all the values.
2. As each degree C value is added, converts it to F and adds the F value to a list for the Fahrenheit equivalents.
3. Makes another loop which prints out C and F similarly to Project 1, i.e. both on the same line, but does it by indexing into the two lists.

Project 3

Generate a list of temperatures from -40 to 100 inclusive by increments of 5 degrees Celsius.   Create another list of the corresponding Fahrenheit temperatures. Create a list of temperatures in degrees Fahrenheit which are greater than zero but for which the corresponding temperature in Celsius is less than zero. Print the elements of this last list.

For Intermediates

Project 4

A. Write a program that obtains the sum of the numbers from 1 to some specified positive (>0) integer N. Do not use the Gauss formula, do this via "brute force."

Print the number, its sum as obtained from your work, and the correct answer from the Gauss formula sum(N)=N(N+1)/2

Test your program with N=1, N=25, N=1000

B. Modify your program to print a table of the sums of the first 25 numbers. Print a header that indicates the columns are Integer and Sum. Try to line up your output as best you can (this will be easier once we have studied formatted output). You can use spaces, indicated by double quotes with some number of spaces in between, to separate the two values on each line of your table.

Project 5

The Collatz conjecture is a fun little exercise in number theory. Given a positive integer, if it is odd multiply it by 3 and add 1. If it is even divide by 2. Repeat this procedure until the result is 1. The Collatz conjecture is that the sequence will always reach 1. No exceptions have been found…yet.

The number of steps required to reach 1 is called the stopping time.

A. Write a program that will find and print the stopping time for the first N positive integers. Count the starting number itself as one of the steps. Print a table of N and stopping time similar to Problem 1B.

Test your program for N=30 and N=50.

B. Modify your program to print the starting number, its stopping time, and the maximum value of the sequence of numbers. Hint: if you use a list you will be able to use the len() and max() intrinsic (built-in) functions. Confirm that you get the same stopping numbers as before.

For Experts

Project 6

Write a program to compute the day of the week for any date of the Gregorian calendar. Here is the formula:

W=(C+Y+L+M+D) mod 7

Y is the last two digits of the actual year and D is the actual day.

You need to obtain the value of C from the following rule for the years:

If year is in the 1400s, 1800s, 2200s, C=2
If year is in the 1500s, 1900s, 2300s, C=0
If year is in the 1600s, 2000s, 2400s, C=5
If year is in the 1700s, 2100s, 2500s, C=4

Months are numbered from 1 in the usual way, but (from January) M is 0, 3, 3, 6, 1, 4, 6, 2, 5, 0, 3, 5

The only tricky part of this algorithm is L, the number of *leap days* that have occurred since the beginning of the century of the given date. To obtain this:
1. Integer divide the last two digits of the year by 4 to obtain the number of “ordinary” leap years in the century up to that year, not counting the century year itself if applicable.
2. Obtain the remainder of the two digits and 4. If it is not a century year and the remainder is 0 the year is a leap year, otherwise it is not. If the year itself is a century year see Step 3.
3. If the century (1400, 1500, etc.) was evenly divisible by 400 then the century year is a leap year, otherwise it is not. Thus 2000 was a leap year but 1900 was not. So add 1 for centuries divisible by 400 and 0 otherwise.
4. If your date is January 1-February 29 of a leap year, subtract 1.

Try to devise a method to obtain the last two digits on your own.

Print the day of the week as a word (Monday, Tuesday, etc.). Remember that Sunday is the first day of the week and it will be counted as 0 in this algorithm.

Hint: use an appropriate data structure to store the names of the days and the values of M.

Test your program first with your own birth date. Then test with the following dates:
Today's date
December 25, 1642 (Note: this is Newton's birth date in the Julian calendar, but use it as a Gregorian date)
October 12, 1492
January 20, 2000
December 11, 2117
November 25, 2402

Project 7

The algorithm for converting a number in base 10 to another base is as follows:

1. Find the remainder of the number divided by the base.
2. Divide the number by the base using integer division. If the result is greater than zero, replace the old value of the number by the result of the integer division and store the remainder previously obtained as the new *leftmost* digit for the base and repeat. If the result is 0 the process is complete.

A. Write a program to convert the first 51 integers, starting at 0 and ending at 50, to octal (base 8). Print a table of the decimal number and its octal equivalent.
Hint: construct a list of digits as you work through the integer divisions. The elements of the list should be strings so you'll need to convert from integer to string. To change from a list of individual strings to a single string for printing, use the join function as follows:
```
  "". join(digits)
```
That is two (regular, not "smart") double quotes with nothing between them, followed by a period, followed by `join` and in parentheses, the name of the list you have created.

B. Modify your program to handle bases >10. Use the letters of the alphabet to represent digits 10, 11, 12, … as A, B, C, … Hint: the char(<number>) built-in converts from an integer to its representation in the *ASCII collating sequence*. Note that A is number 65, i.e. chr(65)="A". The rest of the alphabet follows in numerical sequence to 96, then the lower-case letters begin at 97. Please use upper case letters.

The only widely used base greater than 10 is hexadecimal (base 16). Print a table of 0 to 50 as hexadecimal numbers.

If you have time, try base 20 (vigesimal), which has been used in some historic cultures such as the Mayans, some African tribes, and some areas in and around India. (We presume that for this base, people counted on toes as well as fingers.) Use the letter symbols for digits >10 even though they aren't historically or culturally correct.