Lab 4: Spam! Spam! and more Spam!

CSE/IT 107

1 Introduction

The purpose of this lab is to build on the last two labs.

Coding Conventions

Follow PEP-8 recommendations for your code.

Problems

Make sure your source code files are appropriately named. Make sure your code has a main function; use `boilerplate.py` you created in Lab 1.

McCarthy's 91 Function

John McCarthy, an AI and Lisp pioneer, devised the following function known as McCarthy's 91 Function $M(n)$. For positive integers:

if $n > 100$,

$$M(n) = n - 10$$

if $n <= 100$,

$$M(n) = M(M(n + 11))$$

Write a *recursive* function that determines $M(n)$. For n ranging from 1 to 101, print out `M(n)`. Your output should look something like this:

```
M(102) = 92
```

Use a format statement for your output. Name your source code `mccarthy.py`

Russian Peasant Multiplication

You can determine the product of two integers by doing the following:

If A and B are the two integers to multiply, you repeatedly multiply A by 2 and divide B by 2, until B cannot be divided further. That is, B becomes zero. Remember this is integer division.

During each step, whenever B is odd, you add the corresponding A value to the product you are generating. When B is zero, the sum of A values that had corresponding odd B values is the product.

For example, to find the product 20 * 17 using this method:

A B Comment

20 17 Add A to product, B is odd

40 8 Ignore A, B is even

80 4 Ignore A, B is even

160 2 Ignore A, B is even

320 1 Add A to product, B is odd

Stop as B is zero. The product is the sum 20 + 320 = 340.

Write a program that implements the Russian Peasant algorithm. Get input from the command line. Name your source code `russian.py`.

Collatz Conjecture

The Collatz conjecture, an unsolved math problem, is that given the following formula and an initial positive integer, the generated sequence *always* ends in 1. The sequence of numbers the formula generates is known as the hailstone sequence. The formula is:

> if the number is even is even divide it by 2.
>
> if the number is odd, multiply by 3 and add 1
>
> quit when the number is 1.

Write a program that plots the hailstone sequence and optionally prints the sequence and summary statistics about the sequence.

(a) Accept three inputs on the command line -n [int], which is the integer to determine the hailstone sequence for; `-p` if the user wants to print the sequence as a comma separated list; and `-s` if the use wants to print some summary statistics about the hailstone. The `-p` and `-s` options are optional.

(b) Write a function that returns the hailstone sequence for a given n.

(c) Write a function that plots the hailstone sequence (see below).

(d) Write a function that prints the hailstone sequence.

(e) Write a function that prints summary statistics about the hailstone (see below).

(f) Write a function that finds the number of even terms in the sequence.

(g) Write a function that finds the number of odd terms in the sequence.

Name your source code `collatz.py`

Plotting

To plot the line $y = x$, for you can do the following in code:

```
    import matplotlib.pyplot as plt
    n = range(11)
    plt.plot(n, n, 'bo-')
    plt.show()
```

The third argument of the `plot()` is a format string. The default is `'b-'`, which means to print a blue solid line. If you want to use circular markers for the values you can use `'bo'`. You can combine the color, line style, and marker choice. So to print a blue line with blue circular markers, use `'bo-'`. For more formating options (colors, markers, and line styles), do a `help(plot)` in ipython. Make sure to start ipython with the `--pylab` option.

Summary Statistics

You will print out the length of the sequence, the maximum element of the sequence, the number of even terms in the sequence, and the number of odd terms in the sequence. To find the length of the list use `len(list)`; to find the maximum element of the list use `max(list)`. You will write functions to determine the number of even and odd terms.

Sample Output

Your program output should look like:

```
In [56]: run collatz.py -n 50010000 -p -s
50010000 25005000 12502500 6251250 3125625 9376876 4688438
2344219 7032658 3516329 10548988 5274494 2637247 7911742
3955871 11867614 5933807 17801422 8900711 26702134 13351067
40053202 20026601 60079804 30039902 15019951 45059854 22529927
67589782 33794891 101384674 50692337 152077012 76038506
38019253 114057760 57028880 28514440 14257220 7128610 3564305
10692916 5346458 2673229 8019688 4009844 2004922 1002461
```

```
3007384 1503692 751846 375923 1127770 563885 1691656 845828
422914 211457 634372 317186 158593 475780 237890 118945
356836 178418 89209 267628 133814 66907 200722 100361 301084
150542 75271 225814 112907 338722 169361 508084 254042 127021
381064 190532 95266 47633 142900 71450 35725 107176 53588 26794
13397 40192 20096 10048 5024 2512 1256 628 314 157 472 236 118
59 178 89 268 134 67 202 101 304 152 76 38 19 58 29 88 44 22
11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

length of sequence = 138
max term = 152077012
number of odd terms 44
number of even terms 94
```

And the plot looks like this:

figure_1.png

Rock-paper-scissors-lizard-Spock

On Moodle is a file named `rock-spock.py`. The file contains source code for the game rock-paper-scissors-lizard-Spock. See `http://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock` for the rules of the game.

Unfortunately, the game is not finished. Your job is to finish the program. First run the code to get a feel for what it does. Then add code to do the following:

(a) Finish the function to check that the player enters a correct move.

(b) Finish the function that returns the computer play as a string. Valid strings are `rock, paper, scissors, lizard`, and `spock`. Import the `random` library and use the function `randint()` to generate a random integer between [1, 5]. Associate the values 1 through 5 with a string and return the computers random play.

(c) Finish the function that determines the winner. Print 'The Player Wins' or 'The Computer Wins' along with the phrases:
Scissors cuts paper
Paper covers rock
Rock crushes lizard
Lizard poisons Spock
Spock smashes scissors
Scissors decapitates lizard

Lizard eats paper

Paper disproves Spock

Spock vaporizes rock

Rock crushes scissors

Of course there are 5 ties as well. If there is a tie, print 'A tie'. In all there are 25 cases. This is mainly cut and paste after you set up the first one. Use nested if statements to determine the winner and the correct output. Why is this a better approach than using a series of and statements?

Luhn's algorithm

Luhn's algorithm (`http://en.wikipedia.org/wiki/Luhn_algorithm`) provides a quick way to check if a credit card is valid or not. The algorithm consists of three steps:

(a) Starting with the second to last digit (tens column digit), multiply every other digit by two.

(b) Sum the digits of the products together with the sum of the undoubled digits. The sum of the digits of the products means if the doubled value is 14, the sum of digits is 1 + 4 = 5.

(c) If the total sum modulo by 10 is zero, then the card is valid; otherwise it is invalid.

For example, to check that the Diners Club card 38520000023237 is valid, you would start at 3, double it and double every other digit to give, writing the credit card number as separate digits:

6, 8, 10, 2, 0, 0, 0, 0, 0, 2, 6, 2, 6, 7

Next you would sum the digits of the products with the sum of the undoubled digits:

$6 + 8 + (1 + 0) + 2 + 0 + 0 + 0 + 0 + 0 + 2 + 6 + 2 + 6 + 7 = 40$

Note for 10, since it was doubled you sum its digits (1 + 0).

The last step is to check if $40 \ mod \ 10 = 0$, which it does. So the card is valid.

Write a program that implements Luhn's Algorithm for validating credit cards. Name your source code `luhn.py`

(a) Use the command line to enter the credit card number. Just use argv[1]. No other processing is needed.

(b) Write a function that converts the string to a list of single digit integers. Return the list.

(c) Write a function that carries out the multiplication on the list of integers.

(d) Write a function that sums the digits of the list of integers and returns the sum.

(e) Write a function that determines if the card is valid or not. The function returns a boolean.

(f) In `main()` print whether the credit card is valid or not.

The card given is valid. If you change the last digit of the card to 2 it will become invalid. If you want to test with other credit cards, google "test credit cards".

Submission

Create a tarball of your *.py files.

```
tar czvf cse107_firstname_lastname_lab4.tar.gz *.py
```

To check the contents of your tarball, run the following command:

```
tar tf cse107_firstname_lastname_lab4.tar.gz *.py
```

You should see a list of your Python source code files.

Upload your tarball in Moodle before the start of you next lab.