A4: A Bug's Life

- Assignment A4 should be complete individually.

Learning Objectives

- Learn the impact a bug can have in the real world
- Explore life path numbers
- Practice creating your own fruitful functions
- Learn to use modulus to capture remainders

Start by watching this video.

Then read following from "A Bug's Life: Sometimes a Bug is More than a Nuisance"
by James Gleick
from URL: http://www.around.com/ariane.html

It took the European Space Agency 10 years and $7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space.

One bug, one crash. Of all the careless lines of code recorded in the annals of computer science, this one may stand as the most devastatingly efficient. From interviews with rocketry experts and an analysis prepared for the space agency, a clear path from an arithmetic error to total destruction emerges.

To play the tape backward: At 39 seconds after launch, as the rocket reached an altitude of two and a half miles, a self-destruct mechanism finished off Ariane 5, along with its payload of four expensive and uninsured scientific satellites. Self-destruction was triggered automatically because aerodynamic forces were ripping the boosters from the rocket.

This disintegration had begun an instant before, when the spacecraft swerved off course under the pressure of the three powerful nozzles in its boosters and main engine. The rocket was making an abrupt course correction that was not needed, compensating for a wrong turn that had not taken place.

Steering was controlled by the on-board computer, which mistakenly thought the rocket needed a course change because of numbers coming from the inertial guidance system. That device uses gyroscopes and accelerometers to track motion. The numbers looked like flight data -- bizarre and impossible flight data -- but were actually a diagnostic error message. The guidance system had in fact shut down.

This shutdown occurred 36.7 seconds after launch, when the guidance system's own computer tried to convert one piece of data -- the sideways velocity of the rocket -- from a 64-bit format to a 16-bit format. The number was too big, and an overflow error resulted. When the guidance system shut down, it passed control to an identical, redundant unit, which was there to provide backup in case of just such a failure. But the second unit had failed in the identical manner a few milliseconds before. And why not? It was running the same software.

This bug belongs to a species that has existed since the first computer programmers realized they could store numbers as sequences of bits, atoms of data, ones and zeroes: 1001010001101001. . . . A bug like this might crash a spreadsheet or word processor on a bad day. Ordinarily, though, when a program converts data from one form to another, the conversions are protected by extra lines of code that watch for errors and recover gracefully. Indeed, many of the data conversions in the guidance system's programming included such protection.

But in this case, the programmers had decided that this particular velocity figure would never be large enough to cause trouble. After all, it never had been before. Unluckily, Ariane 5 was a faster rocket than Ariane 4. One extra absurdity: the calculation containing the bug, which shut down the guidance system, which confused the on-board computer, which forced the rocket off course, actually served no purpose once the rocket was in the air. Its only function was to align the system before launch. So it should have been turned off. But engineers chose long ago, in an earlier version of the Ariane, to leave this function running for the first 40 seconds of flight -- a "special feature" meant to make it easy to restart the system in the event of a brief hold in the countdown.

The Europeans hope to launch a new Ariane 5 next spring, this time with a newly designated "software architect" who will oversee a process of more intensive and, they hope, realistic ground simulation. Simulation is the great hope of software debuggers everywhere, though it can never anticipate every feature of real life. "Very tiny details can have terrible consequences," says Jacques Durand, head of the project, in Paris. "That's not surprising, especially in a complex software system such as this is."

These days, we have complex software systems everywhere. We have them in our dishwashers and in our wristwatches, though they're not quite so mission-critical. We have computers in our cars -- from 15 to 50 microprocessors, depending how you count: in the engine, the transmission, the suspensions, the steering, the brakes and every other major subsystem. Each runs its own software, thoroughly tested, simulated and debugged, no doubt.

Bill Powers, vice president for research at Ford, says that cars' computing power is increasingly devoted not just to actual control but to diagnostics and contingency planning -- "Should I abort the mission, and if I abort, where would I go?" he says. "We also have what's called a limp-home strategy." That is, in the worst case, the car is supposed to behave more or less normally, like a car of the pre-computer era, instead of, say, taking it upon itself to swerve into the nearest tree.

The European investigators chose not to single out any particular contractor or department for blame. "A decision was taken," they wrote. "It was not analyzed or fully understood." And "the possible implications of allowing it to continue to function during flight were not realized." They did not attempt to calculate how much time or money was saved by omitting the standard error-protection code.

"The board wishes to point out," they added, with the magnificent blandness of many official accident reports, "that software is an expression of a highly detailed design and does not fail in the same sense as a mechanical system." No. It fails in a different sense. Software built up over years from millions of lines of code, branching and unfolding and intertwining, comes to behave more like an organism than a machine.

"There is no life today without software," says Frank Lanza, an executive vice president of the American rocket maker Lockheed Martin. "The world would probably just collapse." Fortunately, he points out, really important software has a reliability of 99.9999999 percent. At least, until it doesn't.

The instructions

After you have read the above excerpt, respond to the following prompts. You only need write a paragraph of 3-5 sentences or more for each. The goal is that your response shows you have thought about the article and added your own experience or insights to its ideas.

In a paragraph or more, describe which of the many contributing factors within the bug's origins strikes you as the "most preventable" and why? Of course, this will be in hindsight. Explain.	1. To me the most preventable would be to just turn the system off. The article mentioned that once in flight, the system was no longer needed. While the systems could have probably been changed around to handle the numbers, the simplest fix would be to just turn it off.

The investigative report that uncovered the bug stated that software "does not fail in the same sense as a mechanical system." Do you agree or disagree with this statement? Explain in a paragraph.	2. I believe they are correct by saying this. Generally with mechanical systems it is easy to see the failure that is going to happen and when it happens. Sometimes with software you don't know the errors until you run the system (e.g. runtime and semantic errors).

In the remainder of this assignment, you will gain practice in creating and using a fruitful function which employs integer division and modulus.

A key idea in this activity is to use fruitful functions for returning values. Note that when a function returns a value, it is typical to put the value into a variable or use it directly. The following examples illustrate both of these ideas:

- peel_digits.py
- is_even.py

Life Path Numbers

Numerologists consider the **Life Path Number** to be the most important number in one's numerology chart.

The **Life Path Number** is a number which is derived from the sum total of the digits that make up your birth date (with a couple of exceptions).

The exceptions occur with **master numbers**, such as the month of November, which is the 11th month, or birthdays on the 11th or 22nd, or sum totals of 11, 22, or 33. These master numbers are not converted to single digits in the process of computing a Life Path number. Master numbers are all multiples of 11, such as 11, 22, and 33.

- **Step 1**: month: Convert the number of the birth month to digits and sum them, unless the month is November, in which case it is left as an 11.
- **Step 2**: day: Convert the birthday day to digits and sum these unless the day is the 11th or the 22nd, in which case the 11 and 22 are left as an 11 or a 22.
- **Step 3**: year: Convert the year to 4 digits, and sum the four digits.
- **Step 4**: Sum: Add all these digits together from Steps 1, 2, and 3. Reduce this total sum further by adding the remaining digits together until a single digit or a Master Number is obtained.

An example birthday of October 11, 1975 produces a Life Path Number of 7

- **Step 1**: month: Convert the number of the birth month to digits and sum them unless the month is November, in which case it is left as an 11: October = 10 => 1 + 0 = 1.

- **Step 2**: day: Convert the birthday day to digits and sum these unless the day is the 11th or the 22nd, in which case the 11 and 22 is left as an 11 or a 22. 11 is a master number, so it is not reduced.
- **Step 3**: Convert the year to 4 digits, and sum the four digits. 1975 => 1+9+7+5 = 22.
- **Step 4**: Sum all these digits together from Steps 1, 2, and 3. Reduce this total sum further by adding the remaining digits together until a single digit is obtained. 1+ 11 + 22 = 34. Then 3 + 4 = **7**

Another example birthday of December 31, 1981 producing a Life Path Number of 8

- **Step 1**: month: Convert the number of the birth month to digits and sum them unless the month is November, in which case it is left as an 11: December = 12 => 1 + 2 = 3.
- **Step 2**: day: Convert the birthday day to digits and sum these unless the day is the 11th or the 22nd, in which case the 11 and 22 is left as an 11 or a 22. 31 => 3 + 1 = 4
- **Step 3**: Convert the year to 4 digits, and sum the four digits. 1981 => 1+ 9 + 8 + 1 = 19
- **Step 4**: Sum all these digits together from Steps 1, 2, and 3. If the sum of these digits is a master number (11 or 22 or 33), return that number. Unless you get a master number, reduce this total sum further by adding the remaining digits together until a single digit is obtained. 3 + 4 + 19 = 26 => 2 + 6 = 8 is the life path number.

A last example birthday of January 3, 2005 producing a Life Path Number of 11

- **Step 1**: month: Convert the number of the birth month to digits and sum them unless the month is November, in which case it is left as an 11: January = 1 => 1
- **Step 2**: day: Convert the birthday day to digits and sum these unless the day is the 11th or the 22nd, in which case the 11 and 22 is left as an 11 or a 22. 3 => 3
- **Step 3**: Convert the year to 4 digits, and sum the four digits. 2005 => 2 + 0 + 0 +5 = 7
- **Step 4**: Sum all these digits together from Steps 1, 2, and 3. If the sum of these digits is a master number (11 or 22 or 33), return that number. Unless you get a master number, reduce this total sum further by adding the remaining digits together until a single digit is obtained. 1 + 3 + 7 = 11 => 11 is a Master Number so it is not reduced. It is the Life Path Number.

For an online Life Path Calculator see http://seventhlifepath.com/. You can use this to check your computations.

Modulus (Remainder) operator

The modulus operator, indicated by the percent symbol (%), calculates the remainder after dividing the number to the left with the number to the right. For example,

```
10 % 3 = 1
```

because 3 divides into 10 three times and leaves one as a remainder. If you were to take integer division and the modulus operators together, you can undo division to get the original number back:

```
10 % 3 = 1

10 // 3 = 3

(3 * 3) + 1 = 10
```

You may find the following programs useful to your understanding of the modulus operator:

- a4_is_even.py uses modulus to determine whether or not a number is even.
- a4_making_change.py is an example that demonstrates how to use modulus and integer division to make change for a certain amount of money (in cents).

The instructions

Create a new program for the life path number problem. In your program, create at least one fruitful function which does the following computation:

- Takes a single integer as an input parameter.
- Uses modulus (%) to determine if the input is a multiple of 11 (i.e. 11 or 22 or 33). If it is and is 33 or less, the input is a master number, so simply return the master number.
- If the input is not a master number, convert the input to digits and return the sum of the digits.
- Begin with a triple quoted docstring describing what arguments our function takes, what it does, and what the expected result is. It need not be long, but it should have enough info to be able to use the function without having to look at the code.

Make sure that your program:

- Uses functions for encapsulation with triple-quoted docstring for each of these functions. This docstring must include a description of the main purpose of the function, and descriptions of all input parameters as well as what is returned by the function.
- Asks the user to input their birthday. You probably want to use three separate prompts for month, day, and year.
- Uses a type conversion from string to integer for each of these.
- Calls the fruitful function described above with inputs of month, day, and year. Then calls it again to sum the digits.
- Makes use of the modulus function in determining if a number is a Master Number.
- Does something creative and interesting while informing the user of their Life Path Number. This might be a graphical display of the Life Path Number and/or a description of the numerological meaning of that particular number.
- Be sure to include our standard header at the top of your program with name, username, assignment number, purpose and acknowledgements.
- Includes a main() function.
- The highest level of your program (i.e., no indenting) should **only** contain the following:

    - the header
    - any import statements
    - function definitions
    - a call to the main() function
- Also, please be sure to include comments for the logical sections in your code.

Submission Instructions

1. Review the requirements above to ensure you completed everything that was expected of you.
2. Save your code as **A4_life_path_username.py**. Replace *username* with your Berea usernames. For example, the TA Bianca Marrero's file would be **a4_life_path_marrerob.py.**
   NOTE: Incorrect filenames will automatically reduce your grade by 1 point. Fortunately, the format is always the same no matter what the assignment.
3. Upload the Python file to Moodle by the due date listed on the course website: https://trello.com/b/w7bIrLoV/.