Computational Astrophysics I

Assignment 1: Binaries and RNG

The assignment will not be graded, this is for your learning interest.

1 Binary files

This question teaches the concept that all files are actually binary files. It's just that some files has additional metadata, called *file signatures* or *magic numbers/headers*, which designates the file content type.

You need to install the package `bitarray` *to do the assignment. Install it using* `conda install bitarray -c conda-forge` *or* `pip install conda-forge` *based on your system's conda/python installation.*

With the assignment, you're given a file `secret.png`. Obviously, this is a "secret" file and you can't double click and open it.

1. Using the given function, can you read its binary content? You are encouraged to play with the `bitsize` parameter.

```
def read_raw_binary_bit(filepath, bitsize):
    """
    Reads a file given `filepath` and
    outputs binary numbers in raw format with given `bitsize`
    """
    to_return = []
    bits = bitarray()
    with open(filepath, 'rb') as f:
        bits.fromfile(f)
    for i in range(0, len(bits), bitsize):
        to_return.append(bits[i:i+bitsize].to01())
    return to_return
```

2. Did you know that computers store and process in bytes (1 byte = 8 bits)? Try setting bitsize=8, and convert each binary number to their decimal equivalents. (Hint: You can use the `int()` function with proper base.) Make a simple line plot of decimal values, do you notice any repeating shapes or cycles?

3. Do you remember the ASCII encoding that was discussed in the class? Can you decode those binaries or decimals to see whether there is a text hidden? (Hint: You can use the `chr()` function.) Do you see what the data was and does this correlate to what you saw in the plot earlier?

4. You can now open the file using relevant software, and see that it actually opens and has the content that you found out. Change the extension of the file from `.jpg` to something else, what happens if you try to open it?

BONUS: Try to read some other filetypes (e.g., .png, .pdf, .mp3, .mp4, ...) in binary and see the first few bytes. If you decode those first few bytes (first 4-8 bytes suffices) using ASCII, do you notice anything? Since you'll be reading bytewise, you can simply use the function below to read first 6 bytes,

```
with open(file, 'rb') as f:
    data = f.read(6)
```

Note that, these binaries are often easier to read as hexadecimal numbers. Do you see why?

You can refer to the URL for a list of magic numbers for different file formats: https://en.wikipedia.org/wiki/List_of_file_signatures.

Once you do the bonus, I have a treat! You'll be introduced to a tool that you can use to inspect a binary pretty easily.

2 Random number generators (RNG)

Consider the linear congruential generators defined, for $n \geq 0$, by

$$\begin{aligned} \text{PIE}: &\quad Y_{n+1} = (3141592653 Y_n + 2718281829) \bmod 2^{35} \\ \text{RANDU}: &\quad Y_{n+1} = (65539 Y_n) \bmod 2^{31} \end{aligned}$$

where the seed Y_0 can be an arbitrary integer for PIE while it must be an odd integer for RandU. The generator PIE is due to Knuth while RandU was a popular RNG in the 1960's and early 1970's.
(Caution!): *The numbers appearing above are exact integers. Do not modify their values!*

In the following, use the seed value $Y_0 = 1$ in all calculations.

1. Set up a numerical code for generating the integer sequence $\{Y_1, Y_2, \ldots, Y_N\}$ for arbitrary $N \geq 1$, for each of the generators. Having computed these integers, the code should output the floating point numbers

$$U_n \equiv \frac{Y_n}{m}, \quad 1 \leq n \leq N,$$

where the modulus $m = 2^{35}$ for PIE and $m = 2^{31}$ for RANDU. Demonstrate that your code reproduces the values (up to 5 significant digits)

$$U_{10} = 5.5224 \times 10^{-1} \quad \text{for PIE}, \qquad U_{10} = 6.8024 \times 10^{-3} \quad \text{for RANDU}.$$

(Make sure you get these values, before proceeding.)

2. For each of PiE and RandU, generate the sequence $\{U_n\}_{n=1}^N$ with $N = 10^4$ and construct a histogram $p(u)$ of these values using 10 equally spaced bins in the range $0 \leq u < 1$. The histograms should be normalised so that they cover unit area underneath themselves:

$$\int_0^1 du\, p(u) = 1.$$

Plot and compare the resulting histograms with the expected distribution

$$p(u) = \begin{cases} 1, & 0 \leq u < 1, \\ 0, & \text{otherwise.} \end{cases}$$

3. For each of PiE and RandU, generate the sequence of doublets

$$\{(U_{2n-1}, U_{2n})\}_{n=1}^N \quad \text{with } N = 10^4.$$

Plot each of these sequences (separately) as scatter plots in 2 dimensions (i.e., interpret these sequences as $\{(x_n, y_n)\}_{n=1}^N$). Do the 2-d distributions appear uniform in each case?

4. For each of PiE and RandU, generate the sequence of triplets

$$\{(U_{3n-2}, U_{3n-1}, U_{3n})\}_{n=1}^N \quad \text{with } N = 10^4.$$

Plot each of these sequences (separately) as scatter plots in 3 dimensions (i.e., interpret these sequences as $\{(x_n, y_n, z_n)\}_{n=1}^N$). Do the 3-d distributions appear uniform in each case?

Hint: To answer this question properly you must explore a wide range of viewing angles ("camera positions") for the plot. You only need to report the most interesting of these.

5. Based on the results above, which RNG is better, PiE or RandU? Why?