ENGG1003 - Lab Week 5

Brenton Schulz

1 Pre-Reading

Read through the following textbook sections:

- Section 2.4 Random Numbers: `https://link.springer.com/chapter/10.1007/978-3-030-16877-3_2#Sec23`
- Section 3.3.4 Example: Random Walk in Two Dimensions: `https://link.springer.com/chapter/10.1007/978-3-030-16877-3_3#Sec10`

2 Revision - Summations and `for` Loops

The Week 3 lab included Exercise 3.11 from the textbook. This exercise requires an "infinite sum" to be evaluated of the form:

$$\pi = 8\sum_{k=0}^{\infty}\frac{1}{(4k+1)(4k+3)} \tag{1}$$

Breaking down this notation:

- Σ means "sum"
- $k = 0$ below the Σ means that k is a *variable* which takes integer values starting at zero
- ∞ above the Σ means that there is no limit to how large k can become
- Each value of k gets substituted into the $\frac{1}{(4k+1)(4k+3)}$ expression and all results added together
- "$\pi =$" implies that the sum will approach π as k approaches ∞

Putting it all together, the first few terms of Equation 1 can be written out as:

$$\frac{1}{(4\times 0+1)(4\times 0+5)}+\frac{1}{(4\times 1+1)(4\times 1+5)}+\frac{1}{(4\times 2+1)(4\times 2+5)}+... \tag{2}$$

It turns out that expressions of this form map quite cleanly to `for` loops when programming:

- An expression is evaluated multiple times for different values of one variable
- A `sum` variable can "accumulate" calculations inside the loop in the form `sum = sum + <something>`
- k as the "summation variable" becomes the "loop variable" `k`
  - Pay close attention to the starting value of `k=0`
  - Computer's can't count to ∞, so choose a "reasonable" value of, say, 100 000 for the final value of `k`
- The $\frac{1}{(4k+1)(4k+3)}$ expression becomes the Python expression `1/((4*k+1)*(4*k+3))` (note double level nesting of `( ()() )` in the denomenator - without the outer `( )` the expression becomes $\frac{4k+3}{4k+1}$.

Task 1: Python Implementation of Equation 1

Given all the above, write a Python script which implements Equation 1 for `k=0` to `k=99` (ie: with `range(0,100)`.

Remember to multiply the result of the summation by 8 to get an approximation of π!

The final value should be `3.13659`.

If you get `0.39207` the multiply by 8 was omitted.

Once your code is correct repeat the calculation with `range(0,100000)` to get the more accurate result of `3.141587`.

3 Random Numbers

Task 2: Simplified Random Walk

Section 3.3.4 of the textbook shows a “random walk” program which randomly moves a point N, S, E, or W 1000 times.

In this task you will repeat this problem with floating point steps. ie: the x and y movements will be drawn from a uniform distribution using `np.random.uniform()`. This will make the simulation roughly analogous to brownian motion - the movement of particles in a liquid or gas.

The program is to fill in two `numpy` arrays: one of x locations and one of y. The array index will then become a “time” variable. This will allow the simulation results to be plotted with `plot(x,y)`.

Write a Python script which implements the follow pseudocode to move a point around randomly and plot the result:

```
BEGIN
  N = 1000
  x = an array of N zeros
  y = an array of N zeros
  FOR each element, n, in x and y from 1 to N:
    x[n] = x[n-1] + a random float from [-1,1)
    y[n] = y[n-1] + a random float from [-1,1)
  ENDFOR
  plot the set of x-y points, drawing lines between points
END
```

You may call `np.random.uniform()` within the loop or create a pair of additional arrays containing random numbers before entering the loop.

4 The CSV Format

As seen in lectures, we will be studying the use of the `pandas` library for reading CSV (comma separated value/variable) files.

The CSV format allows for "spreadsheet" data to be stored as "human-readable" ASCII text. Column headings and numerical values are written as plain text and columns are separated by commas[1]. Rows are separated by new lines.

Task 3: Opening CSV Files

This lab will make use of a real scientific dataset from the Bureau of Meteorology. The Bureau of Meteorology makes a vast amount of climate data available online here: `http://www.bom.gov.au/climate/data/index.shtml`

You are welcome to browse data as it suits your interest but this lab will make use of the monthly rainfall data for the Nobbys Signal Station weather station on the coast of Newcastle.

Download the `zip` containing the above `csv` data from here:

```
http://www.bom.gov.au/jsp/ncc/cdio/weatherData/av?p_display_type=
monthlyZippedDataFile&p_stn_num=061055&p_c=-745551072&p_nccObsCode=139&
p_startYear=
```

Unzip the file and open the three included files in a **text editor**. The lab computers have Notepad++ installed but you could also use Notepad or, at worst, open them in PyCharm.

Observe the "human readable" nature of the `csv` format. You may wish to keep this file open for debugging purposes.

Observe that the same rainfall data is provided in two layouts:

- `IDCJAC0001_061055_Data1.csv` contains one row per month with the `Year` and `Month` columns indicating which month (from Janurary 1862 to Feburary 2021) the rainfall figure is from
- `IDCJAC0001_061055_Data12.csv` contains a 2D grid where the row indicates a year and column indicates a month.

Note that there are several `null` values in the file ending `_Data12.csv`. These are missing data points. Missing data in `_Data1.csv` is indicated with missing rows where adjacent year/month combinations are discontinuous (eg: It jumps from `1864,08` to `1867,01`).

In this course we will use the `pandas` library to read `csv` files. Before it can be used you must run `pip install pandas` to install it.

[1] Some variations allow for space or tab separation while still being classed as "CSV" files.

Task 4: Reading and Analysing Rainfall Data

This task is presented in multiple parts and is presented in a more "real" style compared to previous lab tasks. You are welcome to discuss problem solving strategies with your lab peers as well as demonstrators.

1. Write a Python script which uses `pandas` to open either of the provided `csv` files and produces a single `numpy` array of monthly rainfall data.

   Your array should be 1910 elements. Data is provided from Janurary 1862 to Feburary 2021 which is $159 \times 12 = 1908$ months plus Janurary and Feburary 2021 = 1910 data points.

   See Slide 18f of the Thursday Week 4 lecture slides for a `pandas` example. The recording timestamp is: `https://www.youtube.com/watch?v=GZmJTGqh5pw#t=50m45s`

   Note that `null`, or missing, data should, be stored in your array as a value of -1. Since negative rainfall is physically impossible this value unambiguously indicates missing data as opposed to, say, zero rainfall.

   Depending on which file you choose to read from different challenges will arise. For example, `_Data1.csv` contains missing rows so you can't simply copy the rainfall data column to an array - the index of the array must be calculated from the year and month columns. If you read from the "2D grid" format you will need to make special provisions for the `null` values.
2. Plot the timeseries data, leaving missing values as -1. A simple `plot(rainfall)` will suffice here as we are visulising the data for development purposes - not producing a plot for publication.
3. Plot a *histogram* of the monthly rainfall data. To do this you can use the `plt.hist()` function from `matplotlib.pyplot`. If the data is stored in the `numpy` array `rainfall` then the histogram can be plotted with:

   ```
   plt.hist(rainfall)
   plt.show()
   ```
4. Summary statistics for Nobbys Signal Station are provided by the BOM at the bottom of this page: `http://www.bom.gov.au/jsp/ncc/cdio/weatherData/av?p_nccObsCode=139&p_display_type=dataFile&p_startYear=&p_c=&p_stn_num=061055`
   From your data, calculate the mean, lowest, and highest rainfall figures for each month and compare them with BOM's calculations.

   Note that the `%` (modulus) operator may be useful when converting an array index to a month. ie: the index modulo 12 will provide a month number where 0 is Janurary and 11 is December.
5. Because the data is missing values we will make some effort to take a "best guess" and fill them in. The timeseries plot would have shown that rainfall data is very random so a technique such as linear interpolation is not appropriate - we need a method based in statistics.

   Fill in the missing data with one twelth of the annual mean figure as reported by the BOM. This is 1118 / 12 = 93mm.
6. Plot the histogram of the "filled in" data. Does it look like the technique above works or have you, say, created false lumps in the histogram?
7. Fill in all missing figures with the mean of that month for all years. ie: if a `null` occurs in Janurary fill it in with 88.7 mm.

   Re-plot the histogram. Is this method any better? Does the histogram look "smoother"?