Assignment 1

Here and in all the following assignments:

(i) **Submit a notebook/notebooks having the python algorithms and the plots generated by them.**

(ii) **Remember NOT to include your name or student number in your report.**

w = 2 means that in grading the assignment this part has weight 2. In FeedbackFruits grades are given in percentages, so w = 2 ≙ 2 x 0.80 x (2/total points) x 100 %, where 0.80 corresponds to 80 % of the grade coming from the exercise solutions. Straightforward, right? 😉

Set up the programming environment that you are going to use in the course (no need to report anything on this).

1. Implement a GGL random number generator and run it to test that it works as it should. Then implement a RAN3 generator so that the initial set of elements (seeds) are generated by the GGL algorithm you wrote. Finally, simulate $X \sim U(0,1)$ and $Y \sim U(0,1)$ using your GGL, your RAN3 and the built-in random number generator in Python (Mersenne-Twister). For each RNG, place the consecutive random numbers as points in (x, y)-coordinates. (More precisely, take consecutively generated random numbers in the order $x, y, x, y, x, y, \dots$) Inspect a sufficiently small interval (slice) $x \in [x_A, x_B]$ $(x_B - x_A < 10^{-3})$ and $y \in (0,\ 1]$ to see if there are any differences. (**Note**: you must generate *a lot* of points, well over 20000 - even up to 10^6 - random number pairs for the visual tests in order to see differences.) Name what you find.
   In your report that you submit you should have:
   (i) (w = 1) Notebook showing clearly the **implementation of ggl.**
   (ii) (w = 3) The required **plot for ggl.**
   (iii) (w = 1) Notebook showing clearly the **implementation of RAN3.**
   (iv) (w = 2) The required **plot for RAN3.**
   (v) (w = 1) **Notebook** with which you ran **Mersenne-Twister** and the required **plot.**
   (vi) (w = 2) **Explanation of the findings** -at least for ggl - **using correct terminology**.

2. In a (ridiculously) simplistic model a person's optimism/pessimism is characterised by a number that changes through interactions with others. A large number means strong optimism and a small number strong pessimism. You want to find out the probability density resulting from a large number of interactions. You take 500 000 agents (persons) in the model and set the number describing their initial state at 50. Each of these agents has 100 interactions in a) and b). In c), due to the nature of the resulting distribution, take only 10 interactions for each agent. The interactions change an agent's state according to the following models:
   a) Independently of previous interactions, each new interaction shifts the agent's state by equal probability towards pessimism by -1, or towards optimism by +1.
   (i) (w = 2) **Obtain and plot the probability mass function** (PMF). **(ii)** (w = 1) **Name the probability mass function (PMF)** (= the discrete probability density function) **and the probability density function (pdf)** it approaches in the limit of infinite number of agents (and also in the limit of continuous shifts of states).

b) Independently of previous transitions, each new interaction shifts the agent's state by equal probability towards pessimism by -0.5, or towards optimism by +10.
(i) (w = 2) **Obtain and plot the probability mass function** (PMF). **(ii)** (w = 1) **Name the probability mass function (PMF)** (= the discrete probability density function) **and the probability density function (pdf)** it approaches in the limit of infinite number of agents (and also in the limit of continuous shifts of states).

c) The agent's response is dependent on previous interactions such that the new state is determined by multiplying the present state by c if the interaction strengthens pessimism and by $1/c$ if the interaction strengthens optimism. Here, $c = 0.7$. Probabilities to shift in either direction are equal. **(i)** (w = 2) **obtain and display the PMF using linear binning (standard histogram).** **(ii)** (w = 2) **Do logarithmic binning** by using python command plt.hist(state, bins=np.logspace(start=np.log10(1), stop=np.log10(2000), num=50), density='true'). Try varying scaling of the axes. **Display only the plot with the scaling from which you can identify the PMF.** Can you roughly recognize the form of the PMF for the appropriately chosen scaling? **Name this PMF form and/or the corresponding asymptotic pdf.** **(iii)** (w = 1) **Name the stochastic process**.

Note 1: You can use ready random generators of python. You get PMF's by plotting histograms (hist-command in matplotlib, with density='true').
Note 2: In some case(s) it may help to have one axis or both axes logarithmic in order to determine the pdf.
Note 3: All PMFs and pdfs can be taken as unnormalized: no need to determine prefactors.