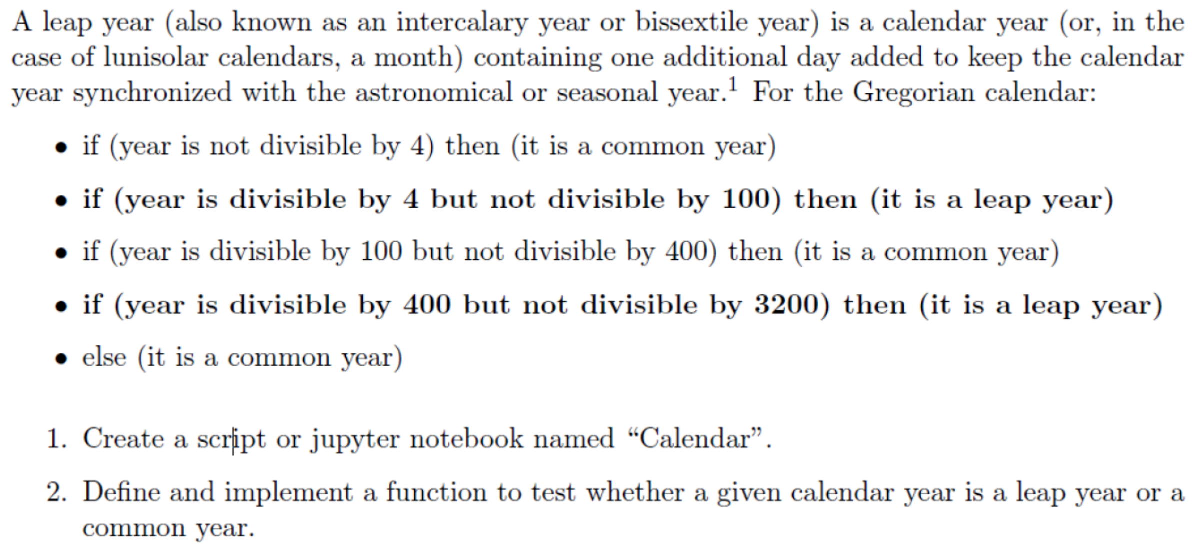
CSCM35/CSLM35 Big Data and Data Mining
Lab assignment 1

The assignment consists of multiple tasks that are designed to be completed during the lab sessions and signed off by either module instructor or teaching assistant. If you are not able to complete the tasks during the lab sessions, then you should do them at home and have them ready to be marked off by the deadline. All lab tasks must be uploaded to Canvas before the deadline stated on each assignment sheet. Source codes must be written in Jupyter Notebook or via Google Colab and formatted neatly with sufficient and clear comments. Plagiarism will not be tolerated. Zip all your files with the following naming convention for submission:

- **[Student Number]-[Last Name][First Initial]-[Assignment][Number].zip**
- **For example: 123456-Scott.Y-Assignment1.zip**

Assignment 1 to be completed by Wed 6 Mar 2024
This assignment is about getting familiar with Python syntax, basic mathematics and external packages commonly used in data mining and machine learning applications.

Task 1.1 Leap Year (3 marks)



A leap year (also known as an intercalary year or bissextile year) is a calendar year (or, in the case of lunisolar calendars, a month) containing one additional day added to keep the calendar year synchronized with the astronomical or seasonal year.[1] For the Gregorian calendar:

- if (year is not divisible by 4) then (it is a common year)
- **if (year is divisible by 4 but not divisible by 100) then (it is a leap year)**
- if (year is divisible by 100 but not divisible by 400) then (it is a common year)
- **if (year is divisible by 400 but not divisible by 3200) then (it is a leap year)**
- else (it is a common year)

1. Create a script or jupyter notebook named "Calendar".
2. Define and implement a function to test whether a given calendar year is a leap year or a common year.

Task 1.2 The Day of the Week (3 marks)

Zeller's congruence is an algorithm devised by Christian Zeller to calculate the day of the week for any Julian or Gregorian calendar date.[2] For the Gregorian calendar, Zeller's congruence is as follows:

$$h = (q + \lfloor\frac{13(m+1)}{5}\rfloor + K + \lfloor\frac{K}{4}\rfloor + \lfloor\frac{J}{4}\rfloor - 2J)\ mod\ 7 \tag{1}$$

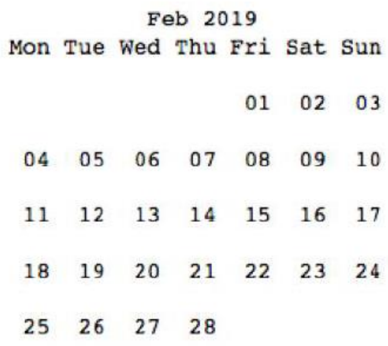
where

- **h** is the day of the week (0 = Saturday, 1 = Sunday, 2 = Monday, ..., 6 = Friday).
- **q** is the day of the month.
- **m** is the month (3 = March, 4 = April, 5 = May, ..., 14 = February). *i.e.* If January or February is entered you must add 12 to the month and minus 1 from the year, which puts them into month 13 or 14 of previous year.
- **K** the year of the century (year mod 100).
- **J** is the zero-based century (actually $\lfloor year/100 \rfloor$). *i.e.* The zero-based centuries for 1995 and 2000 are 19 and 20 respectively (to not be confused with the common ordinal century enumeration which indicates 20th for both cases).
- $\lfloor ... \rfloor$ is the floor function or integer part, *i.e.* // in python.
- **mod** is the modulo operation or remainder after division, *i.e.* % in python.
- For an ISO week date Day-of-Week d (1 = Monday to 7 = Sunday), use $d = ((h + 5)\ mod\ 7) + 1$.

1. Within the same script or *jupyter* notebook created in Task 1.1, define and implement a function to calculate ISO Day-of-Week given a date including year, month and day of month, *i.e.* Year=2019 (calendar year), Month=2 (February), Day of Month=5 (the 5th day of the month).

Task 1.3 Print Calendar of the Year (3 marks)

Print out the monthly calendar of a given year within the terminal or *jupyter* notebook, a



```
          Feb 2019
Mon Tue Wed Thu Fri Sat Sun

                 01  02  03

 04  05  06  07  08  09  10

 11  12  13  14  15  16  17

 18  19  20  21  22  23  24

 25  26  27  28
```

Figure 1: An example of monthly calendar.

partial example is shown in Fig. 1.

- You should include a heading to show the month and the year, *i.e.* Feb 2019.
- You should include a sub-heading to show the day of the week, *i.e.* starting from Mon to Sun.
- Each day in the month should be arranged correctly to match with the heading of the day of the week vertically.
- You can use the functions that have been implemented for Task 1.1 and Task 1.2, but any internal or external data-time related library is prohibited.

Task 1.4 Linear Algebra (3 marks)

Given the following linear equation $\mathbf{AX} = \mathbf{B}$, where $A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$. Create a script or jupyter notebook named "Algebra" which:

- Solve this linear equation using NumPy solver (Check the online documentation for the function *numpy.linalg.solve(...)*).
- Solve this linear equation using NumPy matrix inverse (Check the online documentation for the function *numpy.linalg.inv(...)*).
- **B** can be calculated as a linear combination of the column vectors of **A** that are scaled by corresponding components of column vector **X**. Given **X**, the solution of above linear system obtained from either step 1 or step 2, draw out the linear combination process on a 2-D plane using *matplotlib* package. An example of plot is shown in Fig 2.

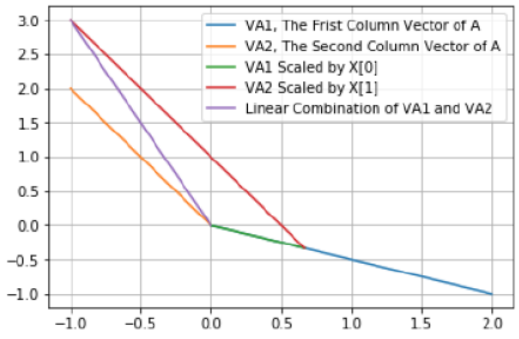


Figure 2: An example of linear combination process. (Note: The value of **B** used in this example is different to the question.)

Task 1.5 Probability and Statistics (3 marks)

Binomial distribution with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent experiment, each asking a yes-no question.

- Binomial distribution can be simulated by using a ball drawing game, *i.e.* there are total of 10 balls in the bag, where 3 are in red and 7 are in black. You are drawing a ball from the bag, but only when the ball you draw from the bag is black, you count the draw as a success. After the draw, the ball needs to be placed back into the bag again. You repeat such draws for n times, the number of successes within n draws is a random variable which subject to Binomial distribution.
- Create a script or jupyter notebook namely "Binomial" which implements n times the ball drawing game. The parameter p corresponds to the probability of drawing black ball out of the bag, *i.e.* 0.7 in the example above.
- One trial of Binomial experiment consists of n replicates as we described above. To calculate the distribution of the Binomial experiment, you need to repeat such trial many times (#Trial > 10000) and record the number of successes in each trial. The histogram of the number of successes for all trials is a good discrete approximation to Binomial distribution when #Trial is sufficiently large.
- Within the script, implement #Trail of Binomial trials, record and plot the histogram of the number of successes all trials. An example of plot is shown in Fig 3.

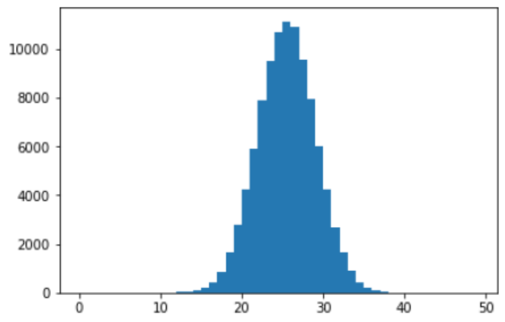


Figure 3: An example of Binomial distribution approximated by the ball drawing game.