Lab 5

COMP9021, Session 2, 2016

1 ☞ Most popular name *Easy*

As in the fifth lecture, we use the data of the National Data on the relative frequency of given names in the population of U.S. births, stored in a subdirectory named `names` of the working directory, in files named `yob`*`xxxx`*`.txt` with *`xxxx`* (the year of birth) ranging from 1880 to 2013. Write a program `most_popular_name.py` that prompts the user for a first name, and finds out the first year when this name was most popular in terms of frequency of names being given, as a female name and as a male name.

Here is a possible interaction:

```
$ python3 most_popular_name.py
Enter a first name: notgivenyet
In all years, notgivenyet was never given as a female name.
In all years, notgivenyet was never given as a male name.
$ python3 most_popular_name.py
Enter a first name: Zed
In all years, Zed was never given as a female name.
In terms of frequency, Zed was the most popular as a male name first in the year 1894.
  It then accounted for 0.01% of all male names
$ python3 most_popular_name.py
Enter a first name: Zilpha
In terms of frequency, Zilpha was the most popular as a female name first in the year 1888.
  It then accounted for 0.01% of all female names
In all years, Zilpha was never given as a male name.
$ python3 most_popular_name.py
Enter a first name: John
In terms of frequency, John was the most popular as a female name first in the year 1880.
  It then accounted for 0.05% of all female names.
In terms of frequency, John was the most popular as a male name first in the year 1880.
  It then accounted for 8.74% of all male names.
$ python3 most_popular_name.py
Enter a first name: Rebecca
In terms of frequency, Rebecca was the most popular as a female name first in the year 1974.
  It then accounted for 1.03% of all female names.
In terms of frequency, Rebecca was the most popular as a male name first in the year 1975.
  It then accounted for 0.00% of all male names.
```

```
$ python3 most_popular_name.py
Enter a first name: Charlotte
In terms of frequency, Charlotte was the most popular as a female name first in the year 2013.
  It then accounted for 0.53% of all female names.
In terms of frequency, Charlotte was the most popular as a male name first in the year 1907.
  It then accounted for 0.00% of all male names.
$ python3 most_popular_name.py
Enter a first name: Madison
In terms of frequency, Madison was the most popular as a female name first in the year 2001.
  It then accounted for 1.23% of all female names.
In terms of frequency, Madison was the most popular as a male name first in the year 1881.
  It then accounted for 0.03% of all male names.
$ python3 most_popular_name.py
Enter a first name: Peter
In terms of frequency, Peter was the most popular as a female name first in the year 1887.
  It then accounted for 0.00% of all female names.
In terms of frequency, Peter was the most popular as a male name first in the year 1957.
  It then accounted for 0.54% of all male names.
```

2 ☞ Highest value of indicator *Easy*

不过你读的*file*都有*14M*，有意思吗老师？

Here we use data available at http://datacatalog.worldbank.org on Health Nutrition and Population statistics, stored in the file HNP_Data.csv, assumed to be saved in the working directory. Write a program highest_value_for_indicator.py that prompts the user for an Indicator Name. If the indicator exists and is associated with a numerical value for some countries or categories, for some the years 1960-2015, then the program finds out the maximum value, and outputs:

- that value;
- the years when that value was reached, from oldest to more recents years;
- for each such year, the countries or categories for which that value was reached, listed in lexicographic order.

Here is a possible interaction:

```
$ python3 highest_value_for_indicator.py
Enter an Indicator Name: Belly explosion by excessive Coca Cola consumption
Sorry, either the indicator of interest does not exist or it has no data.
$ python3 highest_value_for_indicator.py
Enter an Indicator Name: Literacy rate, youth total (% of people ages 15-24)
The maximum value is: 100
It was reached in these years, for these countries or categories:
    2007: ['Azerbaijan']
    2013: ['Moldova']
$ python3 highest_value_for_indicator.py
Enter an Indicator Name: Age population, age 12, female, interpolated
The maximum value is: 13193254
It was reached in these years, for these countries or categories:
    2000: ['China']
$ python3 highest_value_for_indicator.py
Enter an Indicator Name: Newborns protected against tetanus (%)
The maximum value is: 99
It was reached in these years, for these countries or categories:
    2006: ['Bahamas, The']
    2007: ['Bahamas, The']
    2008: ['Bahamas, The', 'Bahrain']
    2009: ['Bahamas, The']
    2010: ['Bahamas, The']
    2011: ['Bahamas, The']
    2012: ['Bahamas, The']
    2013: ['Bahamas, The', 'Guyana']
    2014: ['Bahamas, The', 'Guyana']
```

```
$ python3 highest_value_for_indicator.py
Enter an Indicator Name: Female headed households (% of households with a female head)
The maximum value is: 49.4
It was reached in these years, for these countries or categories:
    2007: ['Ukraine']
$ python3 highest_value_for_indicator.py
Enter an Indicator Name: Number of neonatal deaths
The maximum value is: 5106312
It was reached in these years, for these countries or categories:
    1990: ['World']
$ python3 highest_value_for_indicator.py
Enter an Indicator Name: Age at first marriage, female
The maximum value is: 33.7
It was reached in these years, for these countries or categories:
    1991: ['St. Lucia']
```

3 Extracting information from a web page (optional) *Interesting*

Write a program `SMS_titles.py` that extracts titles from a front page `SMH.html` of the Sydney Morning Herald, also provided under the name `SMH.txt`, meant to be saved in the working directory. You are provided with the expected output, saved in the file `SMS_titles_outputs.txt`, though you might do a better job and remove some of the titles (for instance, *The Lady who lives on the Moon* could go...). Make sure that the output does not include any unwanted HTML entity.

For this question, you can either use the `beautifulsoup` package (see the program `worldbank.py` from the first set of notes) or regular expressions (there is a jupyter notebook sheet on regular expressions).

4 Sierpinski triangle (optional)

Wat?!

请你告诉我，这个题的意义在哪里？

Write a program `sierpinski_triangle.py` that generates Latex code, a `.tex` file, that can be processed with pdflatex to create a `.pdf` file that depicts Sierpinski triangle, obtained from Pascal triangle by drawing a black square when the corresponding number is odd. A simple method is to use a particular case of Luca's theorem, which states that the number of ways of choosing k objects out of n is odd iff all digits in the binary representation of k are digits in the binary representation of n. For instance:

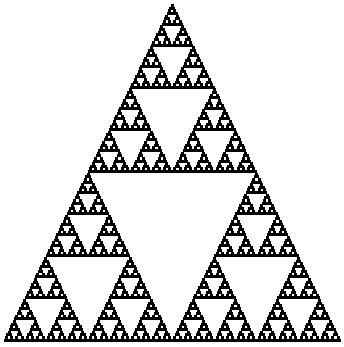
- $\binom{5}{3} = 10$, which corresponds to a white square as 10 is even; indeed, 5 is 101 in binary, 3 is 11 in binary, and there is at least one bit set to 1 in 11 (namely, the leftmost one), which is not set to 1 in 101;
- $\binom{6}{2} = 15$, which corresponds to a black square as 15 is odd; indeed, 6 is 110 in binary, 2 is 10 in binary, and all bits (actually, the only bit) set to 1 in 10 are set to 1 in 110.

So your program has to generate a file named `Sierpinski_triangle.tex`, similar to the one provided; examine the contents of this file to see which text needs to be output.

The file `Sierpinski_triangle.pdf` is also provided, but if you want to generate it yourself from `Sierpinski_triangle.tex`, you need to have Tex installed on your computer (install it if that is not the case, see http://www.tug.org/texlive/), and then execute

```
pdflatex Sierpinski_triangle.tex
```

from the command line, or open `Sierpinski_triangle.tex` in the Latex editor that comes with your distribution of Tex, and it will just be a matter of clicking a button...



5 A calendar program (optional, advanced)

Ah, easy but has a lot of work to do

写个万年历是什么鬼，麻烦死来

Write a program `calendar.py` that provides a variant on the Unix `cal` utility (in particular because it lets the weeks start on Monday, not Sunday), following this kind of interaction:

```
$ python3 calendar.py
I will display a calendar, either for a year or for a month in a year.
The earliest year should be 1753.
For the month, input at least the first three letters of the month's name.
Input year, or year and month, or month and year: 3194 Sept
    September 3194
 Mo Tu We Th Fr Sa Su
           1  2  3  4
  5  6  7  8  9 10 11
 12 13 14 15 16 17 18
 19 20 21 22 23 24 25
 26 27 28 29 30
$ python3 calendar.py
I will display a calendar, either for a year or for a month in a year.
The earliest year should be 1753.
For the month, input at least the first three letters of the month's name.
Input year, or year and month, or month and year: dEcEm 3194
    December 3194
 Mo Tu We Th Fr Sa Su
           1  2  3  4
  5  6  7  8  9 10 11
 12 13 14 15 16 17 18
 19 20 21 22 23 24 25
 26 27 28 29 30 31
```

```
$ python3 calendar.py
I will display a calendar, either for a year or for a month in a year.
The earliest year should be 1753.
For the month, input at least the first three letters of the month's name.
Input year, or year and month, or month and year: 3194
                                3194

       January                   February                  March
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
                1  2      1  2  3  4  5  6      1  2  3  4  5  6
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   7  8  9 10 11 12 13
10 11 12 13 14 15 16  14 15 16 17 18 19 20  14 15 16 17 18 19 20
17 18 19 20 21 22 23  21 22 23 24 25 26 27  21 22 23 24 25 26 27
24 25 26 27 28 29 30  28                    28 29 30 31
31

       April                     May                     June
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
             1  2  3                     1         1  2  3  4  5
 4  5  6  7  8  9 10   2  3  4  5  6  7  8   6  7  8  9 10 11 12
11 12 13 14 15 16 17   9 10 11 12 13 14 15  13 14 15 16 17 18 19
18 19 20 21 22 23 24  16 17 18 19 20 21 22  20 21 22 23 24 25 26
25 26 27 28 29 30     23 24 25 26 27 28 29  27 28 29 30
                      30 31

        July                    August                  September
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
             1  2  3   1  2  3  4  5  6  7            1  2  3  4
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   5  6  7  8  9 10 11
11 12 13 14 15 16 17  15 16 17 18 19 20 21  12 13 14 15 16 17 18
18 19 20 21 22 23 24  22 23 24 25 26 27 28  19 20 21 22 23 24 25
25 26 27 28 29 30 31  29 30 31              26 27 28 29 30
       October                 November                December
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
                1  2      1  2  3  4  5  6            1  2  3  4
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   5  6  7  8  9 10 11
10 11 12 13 14 15 16  14 15 16 17 18 19 20  12 13 14 15 16 17 18
17 18 19 20 21 22 23  21 22 23 24 25 26 27  19 20 21 22 23 24 25
24 25 26 27 28 29 30  28 29 30              26 27 28 29 30 31
31
```

In doing this exercise, you will have to find out (or just remember...) how leap years are determined, and what is so special about the year 1753...