Lab 5

COMP9021, Session 2, 2018

1 Highest value of indicator

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’]
```

2 Longest sequence of consecutive letters

Write a program longest_sequence.py that prompts the user for a string w of lowercase letters and outputs the longest sequence of consecutive letters that occur in w, but with possibly other letters in between, starting as close as possible to the beginning of w.

Here is a possible interaction:

```
$ python3 longest_sequence.py
Please input a string of lowercase letters: a
The solution is: a
$ python3 longest_sequence.py
Please input a string of lowercase letters: abcefgh
The solution is: efgh
$ python3 longest_sequence.py
Please input a string of lowercase letters: abcefg
The solution is: abc
$ python3 longest_sequence.py
Please input a string of lowercase letters: ablccmdnneofffpg
The solution is: abcdefg
$ python3 longest_sequence.py
Please input a string of lowercase letters: abcdiivjwkaalbmmbz
The solution is: ijklm
$ python3 longest_sequence.py
Please input a string of lowercase letters: abcpqrstuvwxbcbcddddeffghijklrst
The solution is: abcdefghijkl
```

3 The 9 puzzle (finding a solution is optional)

Dispatch the integers from `0` to `8`, with 0 possibly changed to `None`, as a list of 3 lists of size 3, to represent a 9 puzzle. For instance, let

```
[[4, 0, 8], [1, 3, 7], [5, 2, 6]]
```

or

```
[[4, None ,8], [1, 3, 7], [5, 2, 6]]
```

represent the 9 puzzle

4		8
1	3	7
5	2	6

with the 8 integers being printed on 8 tiles that are placed in a frame with one location being tile free. The aim is to slide tiles horizontally or vertically so as to eventually reach the configuration

1	2	3
4	5	6
7	8	

It can be shown that the puzzle is solvable iff the permutation of the integers 1, . . . , 8, determined by reading those integers off the puzzle from top to bottom and from left to right, is even. This is clearly a necessary condition since:

- sliding a tile horizontally does not change the number of inversions;
- sliding a tile vertically changes the number of inversions by -2, 0 or 2;
- the parity of the identity is even.

Write a program `nine_puzzle.py` with two functions:

- `validate_9_puzzle(grid)` that prints out whether or not `grid` is a valid representation of a solvable 9 puzzle;

- solve_9_puzzle(grid) that, assuming that grid is a valid representation of a solvable 9 puzzle, outputs a solution to the puzzle, with a minimal number of moves.

Here is a possible interaction:

```
$ python3
Python 3.6.3 ...
>>> from nine_puzzle import *
>>> validate_9_puzzle([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
This is an invalid or unsolvable 9 puzzle
>>> validate_9_puzzle([[1, 2, 3], [4, 1, 6], [7, 8, 0]])
This is an invalid or unsolvable 9 puzzle
>>> validate_9_puzzle([[1, 2, 3], [4, 5, 6], [7, 8]])
This is an invalid or unsolvable 9 puzzle
>>> validate_9_puzzle([[1, 2, 3], [4, 5, 6], [7, 8, 0]])
This is a valid 9 puzzle, and it is solvable
>>> solve_9_puzzle([[1, 2, 3], [4, 5, 6], [7, 8, 0]])
Here is a minimal solution:

  1  2  3
  4  5  6
  7  8
>>> validate_9_puzzle([[1, 2, 3], [4, 5, 6], [None, 8, 7]])
This is an invalid or unsolvable 9 puzzle
>>> validate_9_puzzle([[1, 2, 3], [4, 5, 6], [None, 7, 8]])
This is a valid 9 puzzle, and it is solvable
>>> solve_9_puzzle([[1, 2, 3], [4, 5, 6], [None, 7, 8]])
Here is a minimal solution:

  1  2  3
  4  5  6
     7  8

  1  2  3
  4  5  6
  7     8

  1  2  3
  4  5  6
  7  8
>>> validate_9_puzzle([[4, None, 8], [3, 1, 7], [5, 2, 6]])
This is an invalid or unsolvable 9 puzzle
>>> validate_9_puzzle([[4, None, 8], [1, 3, 7], [5, 2, 6]])
This is a valid 9 puzzle, and it is solvable
```

```
>>> solve_9_puzzle([[4, None, 8], [1, 3, 7], [5, 2, 6]])
Here is a minimal solution:

  4     8
  1  3  7
  5  2  6

  4  3  8
  1     7
  5  2  6

  4  3  8
     1  7
  5  2  6

  4  3  8
  5  1  7
     2  6

  4  3  8
  5  1  7
  2     6

  4  3  8
  5  1  7
  2  6

  4  3  8
  5  1
  2  6  7

  4  3
  5  1  8
  2  6  7

  4     3
  5  1  8
  2  6  7

  4  1  3
  5     8
  2  6  7

  4  1  3
  5  6  8
  2     7

  4  1  3
  5  6  8
  2  7

  4  1  3
  5  6
  2  7  8
```

```
4  1  3
5     6
2  7  8

4  1  3
   5  6
2  7  8

4  1  3
2  5  6
   7  8

4  1  3
2  5  6
7     8

4  1  3
2     6
7  5  8

4  1  3
   2  6
7  5  8

   1  3
4  2  6
7  5  8

1     3
4  2  6
7  5  8

1  2  3
4     6
7  5  8

1  2  3
4  5  6
7     8

1  2  3
4  5  6
7  8
```

4 Sierpinski triangle (optional)

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:

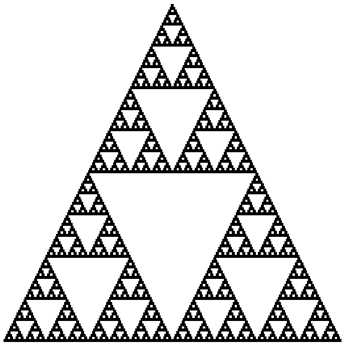
- $\binom{5}{3} = 10$, which corresponds to a white square as 10 is even; indeed, 5 is 101 in binary, 3 is 11 in binary, and there is at least one bit set to 1 in 11 (namely, the leftmost one), which is not set to 1 in 101;
- $\binom{6}{2} = 15$, which corresponds to a black square as 15 is odd; indeed,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

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…