Lab 4

COMP9021, Session 1, 2017

1 ☞ A triangle of characters

Write a program characters_triangle.py that gets a strictly positive integer N as input and outputs a triangle of height N, following this kind of interaction:

```
$ python3 characters_triangle.py
Enter strictly positive number: 13
            A
           BCB
          DEFED
         GHIJIHG
        KLMNONMLK
       PQRSTUTSRQP
      VWXYZABAZYXWV
     CDEFGHIJIHGFEDC
    KLMNOPQRSRQPONMLK
   TUVWXYZABCBAZYXWVUT
  DEFGHIJKLMNMLKJIHGFED
 OPQRSTUVWXYZYXWVUTSRQPO
ABCDEFGHIJKLMLKJIHGFEDCBA
```

Two built-in functions are useful for this exercise:

- `ord()` returns the integer that encodes the character provided as argument;
- `chr()` returns the character encoded by the integer provided as argument.

For instance:

```
>>> ord('A')
65
>>> chr(65)
'A'
```

Consecutive uppercase letters are encoded by consecutive integers. For instance:

```
>>> ord('A'), ord('B'), ord('C')
(65, 66, 67)
```

2 ☞ Pascal triangle

Write a program pascal_triangle.py that prompts the user for a number N and prints out the first $N+1$ lines of Pascal triangle, making sure the numbers are nicely aligned, following this kind of interaction.

```
$ python3 pascal_triangle.py
Enter a nonnegative integer: 3
   1
  1 1
 1 2 1
1 3 3 1
$ python3 pascal_triangle.py
Enter a nonnegative integer: 7
               1
             1   1
           1   2   1
         1   3   3   1
       1   4   6   4   1
     1   5  10  10   5   1
   1   6  15  20  15   6   1
 1   7  21  35  35  21   7   1
$ python3 pascal_triangle.py
Enter a nonnegative integer: 11
                                    1
                                 1     1
                              1     2     1
                           1     3     3     1
                        1     4     6     4     1
                     1     5    10    10     5     1
                  1     6    15    20    15     6     1
               1     7    21    35    35    21     7     1
            1     8    28    56    70    56    28     8     1
         1     9    36    84   126   126    84    36     9     1
      1    10    45   120   210   252   210   120    45    10     1
   1    11    55   165   330   462   462   330   165    55    11     1
```

3 ☞ Computing statistics on the characters in a text

Write a program text_statistics.py that prompts the user for the name of a file and outputs how many times each digit occurs in this file, provided it does occur, following this kind of interaction:

```
$ cat test_1.txt
The Kiwis were the tournaments gallants, but this day were
overwhelmed, perhaps by the occasion,
certainly by Australia's brand of cricket forte.
Plan B - sans McCullum's salvo - had worked against
other attacks, but not the Australians' lair of limber lefties.

Beforehand, speculation centred on how the Kiwis,
playing away from their compact homelands
for the first time in the tournament, would deal
with the vastness of the MCG. Now, though, the problem
was not that the boundaries were too far away,
but the bowlers too close. Starc, and after him Mitch Johnson,
and must have looked like fishtailing trucks
coming towards them, with Josh Hazlewood swerving
from the other direction in the next lane.
"Our bowlers won us the World Cup," Clarke would aver later.
After six weeks of batting hit-and-giggle,
bowlers had the last laugh. They always do.

$ python3 text_statistics.py
Enter the name of a file: test_1.txt
There is no digit in this file.
$ cat test_2.txt
Chevron's decision to sell its 50 per cent stake in
Caltex Australia will make it easier for the local fuel
supplier to release franking credits to shareholders,
Caltex chief financial officer Simon Hepworth says.

Speaking after the $4.6 billion block sale, Caltex management
sought to assure investors that the company's
broader business strategy would be unchanged, despite the
departure of its US-domiciled major shareholder,
which has held its stake for 40 years.

But Mr Hepworth conceded the deployment of the company's
$1.1 billion franking credit balance could be
made easier by the transaction, given that as a US-based
shareholder, the return of franking credits was
not available to Chevron.
$ python3 text_statistics.py
Enter the name of a file: test_2.txt
Digits:    0   1   4   5   6
Count:     2   2   2   1   1
```

4 Map of CO_2 emissions (optional, needs a module not installed on CSE computers)

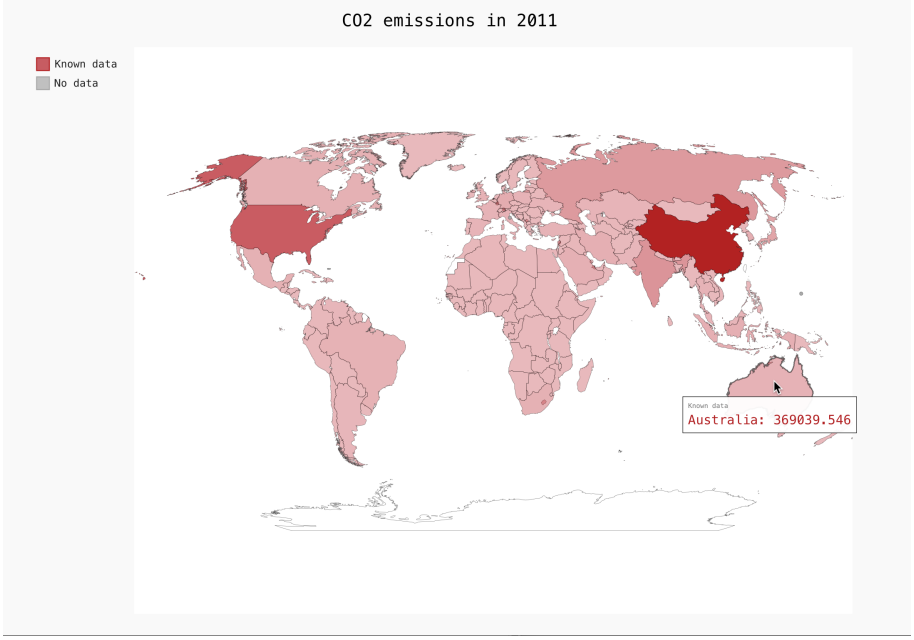
Write a program that extracts from the file `API_EN.ATM.CO2E.KT_DS2_en_csv_v2.csv`, stored in the subdirectory `API_EN` of the working directory, the country CO_2 emissions for the year 2011. Some data in this file are for entities different to countries, or for countries which are not values of the `COUNTRIES` dictionary of the `pygal.maps.world` module. The program will produce an output of the form

```
Leaving out Aruba
Leaving out Arab World
Leaving out American Samoa
Leaving out Antigua and Barbuda
Leaving out Bahamas, The
...
Leaving out Latin America & Caribbean (all income levels)
Leaving out Least developed countries: UN classification
Leaving out Low income
Leaving out Lower middle income
Leaving out Low & middle income
...
Leaving out Virgin Islands (U.S.)
Leaving out Vanuatu
Leaving out West Bank and Gaza
Leaving out World
Leaving out Samoa
```

to let the user know of all those entities and countries, which will be ignored. Some countries are described differently in the dictionary and in the file; these countries will not be ignored. The data will be shown interactively on a map, created as an object of class `World` of the `pygal.maps.world` module, that can be displayed in a browser by opening a file named `CO2_emissions.svg`—check out `render_to_file()`. To create the `World` object from a dictionary having as keys the keys of `COUNTRIES`, check out `add()`. The map should have—check out the `Style` class from the `pygal.style` module:

- as title for the map, `CO2 emissions in 2011`;
- one group of data with `Known data` as legend and with `#B22222` as colour, another group of data with `No data` as legend and with `#A9A9A9` as colour, both with a font size of 10pt;
- tooltips providing standard display for the first group, but with the amount of CO_2 emissions replaced by `?` for the second group, both with a font size of 8pt.

Here is the map with the cursor hovering over Australia, for which the CO_2 emissions are known.



Here is the map with the cursor hovering over Puerto Rico, for which the CO_2 emissions are not known.