Python Exercises

This is a collection of programming examples and problems in Python. These exercises are intended to develop basic algorithmic programming skills.

ciao!

how could we possibly talk about programming without the ubuquitous “Hello world!” program?

Solution: here

minmax

Write a program that reads a sequence of numbers from the standard input, with one or more numbers per line, and prints the minimum and the maximum values in the list. For example, with this input:

```
10
15
7
9
1
3
```

The output must be as follows:

```
min=1
max=15
```

Solution: here

flipline

Write a program that reads a line of text from the standard input, and prints that line on the standard output in reverse, that is, flipped right-to-left. For example, with this input:

```
1 2 3 ciao
```

The output must be as follows:

```
oaic 3 2 1
```

Solution: here

---

pi

Write a program that reads a positive integer N as a command-line argument, and prints the value of the function $\pi(i)$ for every positive $i < N$, where $\pi(i)$ is the number of prime numbers that are less than or equal to i.

For example, with the following invocation

```
./pi.py 10
```

The output must be as follows:

```
1 0
2 1
3 2
4 2
5 3
6 3
7 4
8 4
9 4
10 4
```

Solution: here

---

histogram

Write a program that reads a list of non-negative numbers from the standard input, with one number per line, and prints a histogram corresponding to those numbers. In this histogram, each number x is represented by a line of x characters "#". For example, with this input:

```
10
```

```
15
7
9
1
3
```

The output must be as follows:

```
##########
###############
#######
#########
#
###
```

Solution: here

---

vertical histogram

Write a program that reads a list of numbers n from the standard input, with one or more numbers per line, and prints a vertical histogram corresponding to those numbers. In this histogram, each number x is represented by a column of x characters `#` starting from a base-line of n dash characters `-` representing value 0. Positive numbers are represented by a column above zero while negative numbers are represented with a column below zero.

For example, with this input:

```
7 3 -2 10 5 -3 3 5 8
```

The output must be as follows:

```
   #
   #
   #    #
#  #    #
#  #    #
#  ##  ##
#  ##  ##
## ## ###
## ## ###
## ## ###
---------
  #  #
```

```
  #  #
     #
```

Solution: here

---

compress

Write a program that reads a sequence of space-separated values from the input and outputs a "compressed"; version of the input sequence. The compression is obtained by transforming a sequence of three or more identical elements as X `*` N, where X is the element and N is the number of consecutive copies of X.

For example, with this input:

```
-1 1 1 1 7 7 7 7 5 5 1 1 4 1
```

The output should be

```
-1
1 * 3
7 * 4
5
5
1
1
4
1
```

Solution: here

---

longest identical sequence

Given a sequence of integers $A = a_1, a_2, \ldots, a_n$, a *maximal identical sequence* is a sequence of adjacent identical values $v = a_i = a_{i+1} = a_{i+2} = \ldots = a_{i+k}$ of maximal length k. Write a program that reads an input sequence A, with one or more numbers per line, and outputs the *value* of the first (leftmost) maximal identical sequence in A.

For example, with this input:

```
8 -2 3 3 4 4 4 2 2 2 3 -2 3 3 3 3 4 4 4 8 2 2 8 2 2 22 2 22 22
```

the output must be

```
3
```

Solution: here

---

commonword

Write a program that reads a text file from the standard input and outputs the most common word in the input. The *words* in the input are the maximal-length sequences of alphabetic characters. So, a word does not include spaces or punctuation characters. If two or more words appear more often, the program may print any one of them.

For example, if the input is this:

```
What is free software?

"Free software" is a matter of liberty, not price. To understan
concept, you should think of "free" as in "free speech", not as
"free beer".

Free software is a matter of the users' freedom to run, copy,
distribute, study, change and improve the software.
```

then the output should be this:

```
free
```

Solution: here

---

character frequencies

Write a program that reads a text file from the standard input, and prints the list of all characters present in the input followed by the number of times the character appears in the input.

Solution: here

---

character frequencies 2

Write a variant of the “character frequencies” program described above that outputs all the characters in reverse order of frequency (i.e., from the highest to the lowest frequency count).

Solution: here

---

word frequencies

Write a program that reads a text file from the standard input and prints the list of all the words in the text file, each followed by the number of times the word occurs in the text. Words in the input are the maximal-length sequences of alphabetic characters (e.g., a-z). So, a word does not include spaces or punctuation characters.

Solution: here

---

shortest least common word

Write a program that reads a list of words from the standard input, with one word per line, and prints any one of the shortest words among those that appear least often.

Solution: here

---

largest cluster

Write a program that reads a list of numbers from the standard input, with one or more numbers per line separated by spaces on each line, and prints the largest cluster of extension D. The parameter D is a number that may be given as the first command-line argument. If the parameter is not specified, the default value is $D = 10$. A cluster of extension D is a list of all the elements taken from the input list that are within a distance of at most D from each other. For example, if the input sequence is $12, 2, 34, 7, 21, 24, 50, 45, -9, 7, 45$, and if $D = 10$, then the output should contain the numbers $12, 2, 7, 7$ because those numbers form a cluster of extension 10, and no other cluster of extension 10 contains more than 4 elements.

Solution: here

---

count inversions

Write a program that reads a sequence of numbers from the standard input, with one or more numbers per line separated by spaces on each line, and prints the number of *inversions* in the sequence. In a sequence $A = [a_1, a_2, \ldots, a_n]$, an inversion is a pair a_i, a_j such that $i < j$ and $a_i > a_j$.

Solution: here

roman numeral

Write a program that takes a number N (a positive integer) in decimal notation from the console, and prints its value as a roman numeral.

Solution: here

prime factors

Write a program that reads a positive integer n from the console or from the command line, and outputs its prime factorization. For example, for $n = 2312$, the program should output `2^3 17^2`, and for $n = 10242311$ the output should be `19 701 769`.

Solution: here

maximal regular sequence

Write a program that, given n numbers read from the standard input, outputs the maximal length of a *regular* sequence built with those n numbers. A regular sequence $x_1, x_2, \ldots x_k$ is such that $x_1 \leq x_2 \leq \cdots \leq x_k$ and $x_2 - x_1 = x_3 - x_2 = \ldots x_k - x_{k-1}$. For example, with the input:

```
14 22 5 16 10 34 35 23 51 28 0
```

The program should output

```
5
```

Because there is a regular sequence of 5 elements $(10, 16, 22, 28, 34)$ but there isn't one of 6 or more elements.

check sudoku

Write a program that reads nine lines from the input each containing nine numbers between 1 and 9, representing a sudoku puzzle. The program should output `Correct` if the input represents a valid sudoku.

For example, with the following input, the output should be `Correct`:

```
1 2 3 4 5 6 7 8 9
4 5 6 7 8 9 1 2 3
7 8 9 1 2 3 4 5 6
```

```
2 3 4 5 6 7 8 9 1
5 6 7 8 9 1 2 3 4
8 9 1 2 3 4 5 6 7
3 4 5 6 7 8 9 1 2
6 7 8 9 1 2 3 4 5
9 1 2 3 4 5 6 7 8
```

Conversely, while with the this input, the output shoudl be `Incorrect`:

```
1 2 3 4 5 6 7 8 9
4 5 6 7 8 9 1 2 3
7 8 9 1 2 3 4 5 6
2 3 4 5 9 5 8 7 1
9 6 7 8 6 1 2 3 4
8 9 1 2 3 4 5 6 7
3 4 5 6 7 8 9 1 2
6 7 8 9 1 2 3 4 5
5 1 2 3 4 7 6 9 8
```

here, and whenever the given input is incorrect, the program should also indicate at least one row, or column, or 3x3 sub-matrix that violates the rules of sudoku.

Solution: here

---

maze

Write a program that prints a maze generated at random like this one:

```
/\//\///\\\\/\\\\/\\\\//\/////\/\//\//\/\\/\/\\\////\\//\\
/\\/\/\\\\\\\//\\//\\\/////\//\////\////\////\//\\////\/\/
/\//\\///\//\//\/\\\\\\\\\////\/\\/\\\//\\/////\\\/\\\\\\///
\\\\/\\/////\\//\//\\\//\//\//\\\//\\\\//\/\/\/\////\\\\/\/\/
\\///\\\\\/////\\\\\/\/\//\\\//\/\\//\/\\//\/\\\\//////\\\\/\/\
/\//\//\/\//////\\/\//////////\\/\///////////\\\\/\/\\//\/\\///
/\\\\\//\\/\/\///\/\///\/\////\\//\\\\\/\///\\\//////\\\\\\\//\\
//\//\\///\////\/\///\\/\/\\\\\////\////\/////\////\\//\///
///\\/\//\\/\\/\/\\\\\\//\/\\/\//\/\\\\\//\///\/\\//////\/\/
\//\//\\\/\//////\///\\\\/////\///\\//\/\\\//\\/\//\/\\\\/\/
/\\/////\//\/\\/\\\/\\/\\//\\/\///\\//\\\/////\\/\//\\\\\\/\\
\\/\\\\\///\////\\\\\\\\\\\/\\\\//\//\\\\\\/\\\//\\\\\/\////\/
/\\//\\////\/\///\//\///\///\////\/\///\/\\///\\//////\/\//\\\
\//\\\//\/\\\\////\////\////\///\/\//\/\\//\\\\//\/\/\\\
\/\/\//\\/\/\\/\\/\/\////\\/\\/\/////\///\///\///\/\///\\\
//\\//\\\\\\\///\//\\/\/\\\\\/\/\//\/\\\\\/\/\\//\\////\\\\/
//\//\/\//\\/\//\/\///\//\/\\\\\/\/\/\\\\\///\\\\//////\////\/
```

```
/\////\/\/\///\/\/\\//\\\\\/\\/\\\\\/\//\/\\//\
\//\/////\/\\//\\\/\\\\//\\\\\///\\/\\//\\/\\///\/
//\/////\\\/\/\//\//\\/\////\/\//\\/\\\//\\/\\\/\
```

Solution: here

Also, write a program that, given an input maze like the one above (read from the standard input) finds and prints a complete path that enters from the top and exists from the bottom of the maze. For example, one solution for the maze above is is this one:

```
          \\
          /\\
        /\//\/
       //\\//
       \\///
     /\////
     \\/\/
     //
    //
    \\/\
     \//
     //
    //
   //
  //
 //
/\//
\\/
//
//
```

Solution: here

partition

Write a program that prints all the partitions of a given integer. A partition of n is a bag of integers $a_1, a_2, \ldots$, such that $a_1 + a_2 + \ldots = n$. A "bag" is like a set, in the sense that the order doesn't matter, but it is also like a sequence in the sense that it may contain repeated elements. In this case, to make the output unambiguous, your program must print each partition in decreasing order.

For example, the result for $n = 7$ is:

```
7
6 1
```

```
5 2
5 1 1
4 3
4 2 1
4 1 1 1
3 3 1
3 2 2
3 2 1 1
3 1 1 1 1
2 2 2 1
2 2 1 1 1
2 1 1 1 1 1
1 1 1 1 1 1 1
```

Solution: here

---

fifteen

Write a program to play the game “fifteen”. The game shows a 4-by-4 table containing 15 numbers (or letters), plus an empty cell. The objective of the game is to arrange the numbers in order. A move amounts to moving a cell into the adjacent empty cell. Or, in other words, a move exchanges the empty cell with an adjacent one. For example, starting from this initial state:

```
+--+--+--+--+
|11| 3|12|15|
+--+--+--+--+
| 7| 4| 6|  |
+--+--+--+--+
| 8| 2|10|14|
+--+--+--+--+
| 9| 5| 1|13|
+--+--+--+--+
```

The player may move number 6 into the empty cell, obtaining this configuration:

```
+--+--+--+--+
|11| 3|12|15|
+--+--+--+--+
| 7| 4|  | 6|
+--+--+--+--+
| 8| 2|10|14|
+--+--+--+--+
| 9| 5| 1|13|
+--+--+--+--+
```

Solution: here

tree

Write a program that reads a *tree* and prints it. The program reads trees in two different formats:

1. a text file in which each line contains two strings: *node parent-node*.

   For example:

   ```
   ciao root
   miao root
   mamma ciao
   gatto miao
   cane gatto
   topo gatto
   bimbo mamma
   bambina mamma
   bambini mamma
   ```

2. a text file in which each line contains a sequence of strings separated by the character "/". For example:

   ```
   /usr/share/texmf-texlive
   /usr/share/texmf-texlive/dvips
   /usr/share/texmf-texlive/dvips/pst-text
   /usr/share/texmf-texlive/dvips/hfbright
   /usr/share/texmf-texlive/dvips/tetex
   /usr/share/texmf-texlive/metapost
   /usr/share/texmf-texlive/metapost/venn
   /usr/share/texmf-texlive/metapost/makecirc
   /usr/share/texmf-texlive/metapost/frcursive
   /usr/share/texmf-texlive/metapost/nkarta
   /usr/share/texmf-texlive/fonts
   /usr/share/texmf-texlive/fonts/map
   /usr/share/texmf-texlive/fonts/map/dvips
   /usr/share/texmf-texlive/fonts/map/dvips/eurosym
   /usr/share/texmf-texlive/fonts/map/dvips/hfbright
   /usr/share/texmf-texlive/fonts/map/dvips/updmap
   /usr/share/texmf-texlive/fonts/map/dvips/tetex
   /usr/share/texmf-texlive/fonts/map/dvips/iwona
   /usr/share/texmf-texlive/fonts/map/dvips/phaistos
   /usr/share/texmf-texlive/fonts/ofm
   /usr/share/texmf-texlive/fonts/ofm/public
   ```

```
/usr/share/texmf-texlive/fonts/ofm/public/oinuit
/usr/share/texmf-texlive/fonts/ofm/public/ocherokee
/usr/share/texmf-texlive/fonts/ofm/public/cm-lgc
/usr/share/texmf-texlive/fonts/tfm
/usr/share/texmf-texlive/fonts/tfm/cg
/usr/share/texmf-texlive/fonts/tfm/cg/albertus
/usr/share/texmf-texlive/fonts/tfm/cg/courier
/usr/share/texmf-texlive/fonts/tfm/cg/lettrgth
/usr/share/texmf-texlive/fonts/tfm/cg/coronet
/usr/share/texmf-texlive/fonts/tfm/cg/atqolive
/usr/share/texmf-texlive/fonts/tfm/cg/times
/usr/share/texmf-texlive/fonts/opentype
/usr/share/texmf-texlive/fonts/opentype/public
/usr/share/texmf-texlive/fonts/opentype/public/iwona
/usr/share/texmf-texlive/fonts/opentype/public/antt
/usr/share/texmf-texlive/fonts/truetype
/usr/share/texmf-texlive/fonts/truetype/public
/usr/share/texmf-texlive/fonts/truetype/public/belleek
/usr/share/texmf-texlive/fonts/enc
/usr/share/texmf-texlive/fonts/enc/dvips
/usr/share/texmf-texlive/fonts/enc/dvips/hfbright
/usr/share/texmf-texlive/bibtex
/usr/share/texmf-texlive/bibtex/bst
/usr/share/texmf-texlive/bibtex/bst/directory
/usr/share/texmf-texlive/bibtex/bst/adrconv
/usr/share/texmf-texlive/bibtex/bst/hc
/usr/share/texmf-texlive/xdvi
/usr/share/texmf-texlive/xdvi/pixmaps
/usr/share/texmf-texlive/omega
/usr/share/texmf-texlive/omega/ocp
/usr/share/texmf-texlive/omega/ocp/oinuit
/usr/share/texmf-texlive/omega/ocp/misc
/usr/share/texmf-texlive/omega/ocp/ocherokee
/usr/share/texmf-texlive/omega/otp
/usr/share/texmf-texlive/omega/otp/misc
/usr/share/texmf-texlive/omega/otp/ocherokee
/usr/share/texmf-texlive/source
```

The program must print the resulting tree in one of three formats selected through a command-line argument. If no command-line argument is given, then the program should print the output in all three output format. The formats are defined as follows:

indentation

represents a tree by indenting each element under its parent element. For example, in the case of the first example, the risult should be:

```
root
```

```
ciao
  mamma
    bimbo
    bambina
    bambini
miao
  gatto
    cane
    topo
```

simple

represents a tree with the textual connectors “+” “-” and “|” in a simple way. For example, the output for the first example tree should be this:

```
+-root
 +-ciao
 | +-mamma
 |   +-bimbo
 |   +-bambina
 |   +-bambini
 +-miao
   +-gatto
     +-cane
     +-topo
```

pretty

represents the tree with textual connectors in a richer and more expressive format. For example, the second example should produce this output:

```
•┬share
 └┬texmf-texlive
  ├┬metapost
  ├┬venn
  ├┬makecirc
  ├┬nkarta
  └┬frcursive
  ├┬fonts
  ├┬ofm
  └┬public
   ├┬oinuit
   ├┬ocherokee
   └┬cm-lgc
  ├┬enc
  └┬dvips
   └┬hfbright
  ├┬tfm
  └┬cg
```

```
    │   ├─courier
    │   ├─lettrgth
    │   ├─atqolive
    │   ├─coronet
    │   ├─albertus
    │   └─times
    ├─map
    │ └─dvips
    │   ├─hfbright
    │   ├─updmap
    │   ├─eurosym
    │   ├─tetex
    │   ├─iwona
    │   └─phaistos
    ├─opentype
    │ └─public
    │   ├─iwona
    │   └─antt
    └─truetype
      └─public
        └─belleek
  ├─bibtex
  │ └─bst
  │   ├─directory
  │   ├─adrconv
  │   └─hc
  ├─source
  ├─dvips
  │ ├─pst-text
  │ ├─hfbright
  │ └─tetex
  ├─xdvi
  │ └─pixmaps
  └─omega
    ├─ocp
    │ ├─oinuit
    │ ├─misc
    │ └─ocherokee
    └─otp
      ├─misc
      └─ocherokee
```

Solution: here

---

maxlines

Write a program that reads a text file from standard input and outputs all the lines in the input that have maximal length. For example, if the input is

```
italy
switzerland
germany
sweden
austria
netherlands
belgium
france
```

then the output should be

```
switzerland
netherlands
```

Solution: here

---

program-x

Examine **this python program**. Write another Python program that does the same thing, but with a better complexity. Test your program with the following input files (passed through standard input).

- **points_{10}**
- **points_{100}**
- **points_{1000}**
- **points_{10000}**
- **points_{20}**
- **points_{200}**
- **points_{2000}**
- **points_{50}**
- **points_{500}**
- **points_{5000}**

Solution: here

---

tst

Write a program that reads a list of words from the standard input, constructs a ternary search trie (TST) by inserting each word in the TST in the given order, and then prints the TST on the standard output, which is a textual output stream.

The output of your program must conform exactly to a specific format defined as follows. This format can be best described through a series of examples.

- **input 1** should produce **output 1**
- **input 2** should produce **output 2**
- **input 3** should produce **output 3**

Solution: **here**

---

plusminus

Write a program that reads a string of digits S as a command-line argument, and inserts plus (`+`) and minus signs (`-`) in the sequence of digits so as to create an expression that evaluates to zero. The program should output a resulting expression, if one such expression exists, or `None` if no such expressions exist.

For example, if S is `201015900001`, then the program could output `+2+0+10+1-5-9+0+0+0+0+1`. Instead, if S is `200015800001`, then the program should output `None`.

Optimization: Write a variant of this program that produces expressions with the least number of plus or minus signs. For example, an input value S `2391222918889` could produce an output expression `+2+3+9+1+2-2-29-1+8+8+8-9`. However, a better expression would be `+239-122-29+1-88+8-9`, since the first one has 12 elements while the second one has only 7, and the minimum size is 6 (e.g., `+239-12-229+1-88+89`).

As another example, consider $S =$ `382124861290230228328992 2`. There exists an expression of only 7 elements, which is minimal: `24-8612-9023+022832-8992-2`

Use your program to find the *minimal* expressions for

- $S =$ `74926578305522048456443 93`
- $S =$ `92374381610287463001735516`
- $S =$ `38210486125023022832899225 3`
- $S =$ `46928365011828245521312837 82`
- $S =$ `46928365011828245521312837821`

Solution: **here**