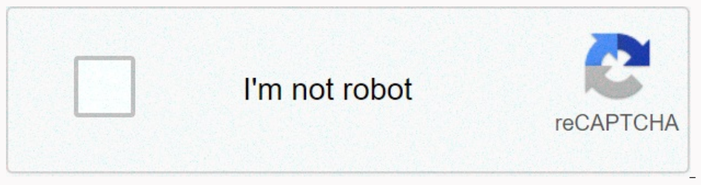

Continue

Python write float array to file

The most basic way to write a file in Python is to open a file with write access: f=open('file.txt', 'wb'), and then call the writing method to write to the file: Returning the line must be clearly included using: f.write(line 1) f.write(line 2), and the file should be closed with: The best way to write a variable with this technique is to use the string, which is described in more detail here. The basic command to format a variable as a string is: where the string contains a string with a tuple of the value, for example: >>> %s %5.2f %10.4e % (name, 3.4, 1.e-10). Name 3.40 1.0000e-10, we can use this when writing a file, so if we have two lists or an array of values a and b you can do: =[1,2,3,4,5] b = [2,6,4,3,2] f = open(file.txt', 'wb') for i in the range (len(a)): f.write(%i %5.5 2f % (a[i], b[i])) f.close(), which produces files with: 1 2.00 2 6.00 3 4.00 4 3.00 5 2.00 Numpy has a function called savetxt that makes it easy to write an array to a file. One can do: import numpy into np a = [1,2,3,4,5] b = [2,6,4,3,2] np.savetxt ('file_numpy.txt', zip(a, b), fmt=i%5.2f), which produces the same output on the top and avoids loops. v_0t This section summarizes the basic file management in Python, including the file utility in the file readout numpy package{1}{2}. 2] We want to read the parameters in the file into the Python variable and create a table with columns \(t\) and \(s)\) for \(t\in [0,2] \) with the steps of \(\Delta t=0.1 \) (as specified by dt and range.dat in the file. 'r') for the line in file: # value = line.split('=') variable =variable.strip() # removes the leading/traling space. Exit if variable =='v0': v0 = floating elif variable (value) =='a': a =float(value). Elif variable == 'dt': dt = floating elif variable (value) == 'range': Interval = eval(value) infile.close() The name of the file is here.dat. To split a line into a word separated by the character '=', we use line.split('='), which results in a list of words. For example, >>> = variable 'v0 = 5.3' >>> Value = line.split('=') >>> Variable 'v0' values >>> ' 5.3'. It is important to note that the value (the text to the right of = on each line) is a string variable. We need to convert it into a floating object to get a variable that we can calculate. Assignment interval = eval(value) performs some magic and deserves an explanation: evals) to interpret the text in the string as python code in the current example. The value is [0, 2] and is interpreted as python code, for example, as a list, and the interval becomes the name for the entry object with content [0, 2.dat]. Now there is no need to close the file because it is done automatically after blocking the file writing, assuming we create \(t\) and \(s)\) values in two lists t_values and s_values and want to write these are beautifully formatted tables to file. The following code works: outfile = open('table1.dat', 'w') outfile.write('#t(t)'). Write table headers for t, s in zip(t_values, s_values): outfile.write('%.2f %.4f' % (t, s)) The file intended for writing must be opened with the parameter 'w'. The numpy package contains a convenient savetxt function for saving tabular data. The savetxt function allows to control the format of numbers in each column (fmt) through the printf syntax. s(t)] in each row data = np.array([t_values, s_values]).dat. The tabular data in the file can be read back into the numpy array by the loadtxt function: data =np.loadtxt('table2.dat', comments='#') the line that starts with the comment character is skipped to read. The resulting object data is a two-dimensional array: data[i,j]. The row values i and the j column number in the table, such as data[i,0], contain \(t \) and data[i,1]. The value \(s(t) \) in the row i numpy.savetxt(fname, X, fmt='%.18e', delimiter=' ', new line=', ='header, fnamefilename parameter or file manager if the file name .gz file is automatically saved in gzip format that compresses loadtxt understand gzipped files transparently X1D array_likeData 2D is saved to fmtstr text file or strs, single format option A (%10.5f), a sequence of formats, or multiple string formats, eg. For X complex, the legal option for fmt is: single identifier fmt='%.4e'. a full string that identifies all parts of the real and imaginary sections, such as ' %.4e %+.4ej %.4ej %+.4ej %.4ej %.4ej '.4ej %+.4ej' for 3 specified columns, one section per column - in this case, the actual and imaginary sections must have a separate identifier, such as ['%.3e + %.3ej', '(%.15e%+.15ej)'] for 2 columns Delimiter New lines, auxiliary strings, or header dividers, additional strings to be written at the beginning of the footer file, additional strings to be written at the end of the Commentsstr file, additional strings that are predefined to header and footer strings. to mark as a comment. Default value: '#' as expected, such as numpy.loadtxt encoding{none, str}, ไม่สามารถใช้ได้กับกระแสข้อมูลเอาต์พุต If the encoding is something other than 'bytes' or 'latin1' you will not be able to load the file in NumPy versions < 1.14. Default is 'latin1'. See also saveSave an array to a binary file in NumPy .npy format savezSave several arrays into an uncompressed .npz archive savez_compressedSave several arrays into a compressed .npz archive Notes Further explanation of the fmt parameter (%[flag]width[.precision]specifier): flags:- : left justify + : Forces to precede result with + or -. 0 : Left pad the number with zeros instead of space (see width). width:Minimum number of characters to be printed. The value is not truncated if it has more characters. precision: For integer specifiers (eg. d,i,o,x), the minimum number of digits. For, and f specifiers, the number of digits to print after the decimal point. For g and G, the maximum number of significant digits. For s, the maximum number of characters. specifiers:c : character d or i : signed decimal integer or : scientific notation with or. f : decimal floating point g,G : use the shorter of e,E or f o : signed octal s : string of characters u : unsigned decimal integer x,X : unsigned hexadecimal integer This explanation of fmt is not complete, for an exhaustive specification see [1]. References 1 Format Specification Mini-Language, Python Documentation. Examples >>> x = y = z = np.arange(0.0,5.0,1.0) >>> np.savetxt('test.out', x, delimiter=',') # X is an array >>> np.savetxt('test.out', (x,y,z)) # x,y,z equal sized 1D arrays >>> np.savetxt('test.out', x, fmt='%1.4e') # use exponential notation Task Write two equal-sized numerical arrays 'x' and 'y' to a two-column text file named ' ชื่อแฟ้ม' The first column of the file contains a value from the specified array 'x' For example, consider: x = {1, 2, y = {1, 1.4142135623730951, 1.732050807568772, /* sqrt(x) */ xprecision = 3; precipitation = 5; This work is intended as a sub-task for measuring the relative efficiency of sorting ada usage algorithms[edit] Ada.Text_IO; use Ada.Text_IO;with Ada.Float_Text_IO; use Ada.Float_Text_IO;with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions; Ada.Numerics.Elementary_Functions; Step Write_Float_Array is Float_Array type (1..4) array of Float; procedure Write_Columns (file: File_Type; X: Float_Array; Y: Float_Array; X_Pres: Nature := 3; Y_Pres: Natural := 5) start for me in Float_Array'range loop Put file (=>, Entry => X(I), Fore => 1, Aft => X_Pres - 1 New_Line Y_Pres); Write_Columns end loops; File: File_Type; X: Float_Array:= (1.0, 2.0, 3.0, 1.0e11 Float_Array); Get_Line Out_File For me in Float_Array'range loop Y(I) := Sqrt (X (I)); end loop; Write_columns (file, X, Y); CLOSE (file);Write_Float_Array; ถ้า errno NE 0 แล้วหยุด FI; สำหรับ i TO UPB x DO # รูปแบบ := ถ้าเลขชี้กำลังสัมบูรณ์มีขนาดเล็กพอให้ใช้ดงที่ ELSE ใช้ลอย FI; # FORMAT repr x := (ABS log(x[i]) <x width= |= $g(-x= width,x= width-2)$= |= $g(-x= width,x= width-4,-1)$=),= repr= y= :=(abs=></x> <y width= |= $g(-y= width,y= width-2)$= |= $g(-y= width,y= width-4,-1)$=);= putf(f,= (repr= x,= x[i,= $= $,= repr= y,= y[i,= l))= od;= close(f));#= example= usage:= #test:(= []real= x=(1, 2,= 3,= 1e11);= [upb= x]real= y;= for= i= to= upb= x= do= y[i]:=sqrt(x[i]) od;= printf(($x= before:$,= xg,= x,= l));= printf(($y= before:$,= xg,= y,= l));= writedat(sqrt.dat,= x,= y,= 3+2,= 5+2); = printf($after:l$);= file= sqrt= dat;= int= errno=open(sqrt dat,= sqrt.dat,= stand= in= channel);= if= errno= ne= 0= then= stop= fi;= on= logical= file= end(sqrt= dat,= (ref= file= sqrt= dat)bool:= stop);= to= upb= x= do= string= line;= get(sqrt= dat,= (line,= new= line));= print((line,new= line))= od)= x= before:= +1.00000000000000e= +0= +2.00000000000000e= +0= +3.00000000000000e= +0= +1.00000000000000e= +11= y= before:= +1.00000000000000e= +0= +1.41421356237310e= +0= +1.73205080756888e= +0= +3.16227766016838e= +5= after:= 1.000= 1.00000= 2.000= 1.41421= 3.000= 1.73205= 1e11= 316228= awk[edit]= as= usual,= the= order= of= array= traversal= in= awk= is= not= necessarily= the= same= as= the= input= had:= $= awk= 'begin{split(1= 2= 3= 1e11,x);=>แยก(1 1.4142135623730951 1.7320508075688772 316227.76601683791,y);> สำหรับ(i in x)printf(%6g 3.1623e+05 1 2 1.4142 3 1.7321 for the text file section of the job, just redirect the stdout to that task. For i% = 0 to 3 y(i%) = SQR(x(i%)) XPRECISION Next = 3 Depreciation = 5 outfile% = OPENOUT(file name .txt) IF outfile%=0 ข้อผิดพลาด 100, ไม่สามารถสร้างไฟล์ได้ สำหรับ i% = 0 ถึง 3 @% = & 1000000 + (xprecision < 8)= a$=STR$(x(i%)) += chr$(9)= @%=&1000000 += (yprecision=> < 8)= a$= +=STR$(y(i%)) print= #outfile%,= a$:= bput= #outfile%,= 10= next = close= #outfile%= 1= 1= 2= 1.4142= 3= 1.7321= 1e11= 3.1623e5= c[edit]= #include=> <stdio.h>#include <math.h>int main(int argc, char **argv) { float x[4] = {1,2,3,1e11}, y[4]; int i = 0; ไฟล์ *filePtr; filePtr = fopen(floatArray,w); สำหรับ (i = 0; i < 4;= i++)= {= y[i]=sqrt(x[i]); fprintf(fileptr,= %.3g\t%.5g,= x[i],= y[i]);= } = return= 0;}= the= file= floatarray= then= contains= the= following:= 1= 12= 1.41423= 1.73211e+11= 3.1623e+05= c#[edit]= using= system.io; class= program{= static= void= main(string[]= args)= {= var= x=new double[]= {= 1,= 2,= 3,= 1e11= };= var= y=new double[]= {= 1,= 1.4142135623730951,= 1.7320508075688772,= 316227.76601683791= }; = int= xprecision=3; int= yprecision=5; string= formatstring={0:G += xprecision= += }\t{1:g= += yprecision= += }; = using= (var= outf=new streamwriter(floatarraycolumns.txt))= for= (int= i=0; i=> < x.length;= i++)= outf.writeline(formatstring,= x[i],= y[i]);= }}= 1= 1= 2= 1.4142= 3= 1.7321= 1e+11= 3.1623e+05= c++[edit]= function= writedat():=> <class inputiterator,= class= inputiterator2=>void writedat (const char* ชื่อไฟล์, InputIterator xbegin, InputIterator xend, InputIterator2 ybegin, InputIterator2 เยน, int xprecision=3, int yprecision=5){ std::ofstream f.exceptions(std::ofstream::failbit | std::ofstream::badbit); f.open(ชื่อแฟ้ม); สำหรับ (; xbegin != xend และ ybegin !) ++xbegin, ++ybegin) f < std::setprecision(xprecision)=> < *xbegin=> < '\t'=> < std::setprecision(yprecision)=> < *ybegin=> < '';}= example:= #include=> <algorithm>#include <cmath>// ::sqrt()#include <fstream>#include <iomanip>// setprecision()#include <iostream>#include #include <string> <vector>int main(){ try { // เตรียมข้อมูลการทดสอบคู่ x[] = {1, 2, 3, 1e11}; const size_t xsize = sizeof(x) / sizeof(*x); std::เวก<double>เตอร์ y(xsize); std::transform(&x[0], x[xsize], y.begin(), ::sqrt); เขียนแฟ้มโดยใช้ความแม่นยำเริ่มต้น writedat(sqrt.dat, &x[0], &x[xsize], y.begin(), y.end()); พิมพ์แฟ้มผลลัพธ์::ifstream f(sqrt.dat); สำหรับ (std::สายอักขระ; std::getline(f, line);) std::cout < line=> < std::endl;= }= catch(std::exception&= e)= {= std::cerr=> < writedat:= exception:= '=> < e.what()=> < '; return 1; } return 0;} 1 1 2 1.4142 3 1.7321 1e+11 3.1623e+05 COBOL[edit] identification division. program-id. wr-float. environment division. input-output section. file-control. select report-file float.txt organization sequential. data division. file section. fd report-file report is floats. working-storage section. 1 i binary pic 9(4). 1 x-values comp-2. 2 value 1.0. 2 value 2.0. 2 value 3.0. 2 value 1.0e11. 1 redefines ';= return= 1;= }= return= 0;} = 1= 1= 2= 1.4142= 3= 1.7321= 1e+11= 3.1623e+05= cobol[edit]= = identification= division.= program-id.= wr-float.= environment= division.= input-output= section.= file-control.= select= report-file= assign= float.txt= organization= sequential.= data= division.= file= section.= fd= report-file= report= is= floats.= working-storage= section.= 1= i= binary= pic= 9(4).= 1= x-values= comp-2.= 2= value= 1.0.= 2= value= 2.0.= 2= value= 3.0.= 2= value= 1.0e11.= 1= redefines=></ '; return 1; } return 0;} 1 1 2 1.4142 3 1.7321 1e+11 3.1623e+05 COBOL[edit] identification division. program-id. wr-float. environment division. input-output section. file-control. select report-file assign float.txt organization sequential. data division. file section. fd report-file report is floats. working-storage section. 1 i binary pic 9(4). 1 x-values comp-2. 2 value 1.0. 2 value 2.0. 2 value 3.0. 2 value 1.0e11. 1 redefines > </double> </vector> </string> </iostream> </iomanip> </fstream> </cmath> </algorithm> </class> </math.h> </stdio.h> </stdio.h> Comp-2. 4. 1 Comp-2. For more information, see Floating Road 1 Floating- Line Type D of 2 Lines Plus 1. 3 Columns 1 Figure -9.99e+99 source x(i) 2 columns 12 pic -9.9999e+99 source y(i) Step section Start Open default file output float operation differently i from 1 by 1 until i > 4 y(i) processing = sqrt function (x(i)) floating build- สายสิ้นสุดดำเนินการสิ้นสุดลอยปิดรายงาน- ไฟล์หยุดทำงานของ สิ้นสุดโปรแกรม wr-float 1.00E+00 1.0000E+00 2.00E+00 1.4142E+00 3.00E+00 1.7321E+0 0 1.00E+11 3.1623E+05 Common Lisp[edit] (พร้อมแฟ้มเปิด (กระแสข้อมูล (ชื่อเส้นทาง:ชื่อเส้นทาง:d) :dทิศทาง :output) (ให้* ((x (อาร์เรย์ 4 :เนื้อหาเริ่มต้น '(1 2 3 1e11)) (y (แผนที่ 'เวกเตอร์ 'sqrt x)) (xprecision 3) (yprecision 5) (fmt (รูปแบบ nil ~~,1,~~ d,,G~~~12t~~,~dG~~%xprecision yprecision))) (แผนที่ nil (lambda (b) (รูปแบบสตรีม fmt a b)) x y)) โดยใช้ CLISP ฉันได้รับ 1. 1.0000 2. 1.4142 3. 1.7321 1.0E+011 3.16228E+5 D[edit] นำเข้า std.file, std.conv, std.string; โมฆะหลัก() { Auto x = [1.0, 2, 3, 1e11]; Auto y = [1.0, 1.4142135623730951, 1.73205080756872, 316227.76601683791]; fx; x) tmp ~= format (% ~ text (xPrecision) ~ g %. ~ Text (yPrecision) ~ g\r, fx, y[i]float_array.txt); 11 3.1623e+05 Elixir[edit] defmodule Write_float_arrays run def(xs, ys, fname, precision\\[]) do xprecision = Keyword.get(precision: x, 2) yprecision = Keyword.(get, accuracy :y, 3) รูปแบบ = ~.#{xprecision}g\t~.#{yprecision}g~n File.open! (fname, [:write], fn file -> Enum.zip(xs, ys) |> Enum.each(fn {x, y} -> :io.fwrite file, รูปแบบ[x, y] endend x = [1.0, 2.0, 3.0, 1.0e11]y = สำหรับ n <- x, do: :math.sqrt(n)fname = ชื่อแฟ้ม.txt Write_float_arrays.task(x, y, fname)IO.puts File.read! (fname) ความแม่นยำ = [x: 3, y: 5]Write_float_arrays.task(x, y, fname, ความแม่นยำ)IO.puts File.read! (fname) 1.0 1.00 2.0 1.41 3.0 1.73 1.0e+11 3.16e+5 1.00 1.0000 2.00 1.4142 3.00 1.7321 1.00e+11 3.1623e+5 Erlang[edit] Erlang คิดว่า 1 เป็นจำนวนเต็ม เพื่อโน้มน้าวมันมิฉะนั้นฉันต้องใช้ 1.0 -โมดูล(write_float_arrays) ของ -ส่งออก([งาน/0, to_a_text_file/3, to_a_text_file/4]) Jobs() -> File = afile, Xs = [1.0, 2.0, 3.0, 1.0e11], Ys = [1.0, 1.4142135623730951, 1.73205080768772, 316227.76601683791], Option = [{xcicision, 3}, ycipreions, 5], file to_a_text_file Xs, Ys, Options), {ok, Content} = file:read_file (file), io:fwrite (file content: ~p~n, [content]) to_a_text_file (file, Xs, Ys) -> to_a_text_file (file, Xs, Ys, [] to_a_text_file (file, Xs, Ys, file) Options) -> Xprecision = List of Proponents:get_value (xprecision, Options, 2), Yprecision = proplists:get_value (prediction, options, 2), Format = List: Flat (io_lib:format(~~.~pg~pg~n, [Xprecision, Yprecision]), {ok, IO} = File:Open (File, [Burn]), {ok= io:fw(IO, Format, [X, Y]) | <- lists:zip(= xs,= ys)],= file:close(= io=). = 3=>write_float_arrays:task() เนื้อหาของไฟล์: <1.00 1.00002.00= 1.41423.00= 1.73211.00e+11= 3.1623e+5=>> Euphoria[edit] ค่าคงที่ x = {1, 2, 3, 1e11}, y = {1, 1.4142135623730951, 1.7320508075688772, 316227.76601683791 จำนวนเต็ม fn fn = เปิด (ชื่อแฟ้ม,w)สำหรับ n = 1 ถึงความยาว (x) ทำ printf(fn,%.3g\t%.5g,{x[n],y[n]})สิ้นสุด forclose(fn) F#[edit] [<EntryPoint>]ให้ argv หลัก = x = [1.; 2.; 3.; 1e11] ให้ y = List.map System.Math.Sqrt x ให้ xprecision = 3 ให้ yprecision = 5 ใช้ไฟล์ = System.IO.File.CreateText(ลอย.dat) ให้บรรทัด = sprintf %*g\t%*g List.iter2 (สนุก x y -> ไฟล์. WriteLine (บรรทัด xprecision x การคิดค่าเสื่อมราคา y)) x y 0 เนื้อหาของแฟ้ม, แสดงภาพด้วย TAB=8 1 1 2 1.4142 3 1.7321 1e+11 3.1623e+05 Forth[edit] สร้าง x 1e f, 2e f, 3e f, 1e11 f, create y 1e f, 2e fsqrt f, 3e fsqrt f, 1e11 fsqrt f, : หลัก s sqrt.txt w /o เปิดไฟล์โยนเพื่อ outfile-id 4 ทำ 4 ชุดความแม่นยำ x i ลอย + f. 6 ชุดความแม่นยำ y i ลอย + f. cr ห่วง outfile-id stdout เพื่อ outfile-id ปิดไฟล์; โยน Fortran[แก้ไข] Fortran 90[แก้ไข] โปรแกรม writefloats โดยนัยไม่มีจริง, มิติ(10) :: เป็นจำนวนเต็ม sqrta :: i จำนวนเต็มพารามิเตอร์ :: หน่วย = 40 a = (/ (i, i = 1,10) /) sqrta = sqrt (a) เป็น (หน่วย, ไฟล์ = xydata.txt, สถานะ =ใหม่, การกระทำ =เขียน) ปิด (หน่วย) ประกอบด้วย subroutine writexy (u, x, y) จริงมิติ (:), เจตนา (ใน) :: x, y integer, intent(in) :: u integer :: i write(u, (2F10.10.14)) (x(i), y(i), i=lbound(x,1), ubound(x,1)) สิ้นสุดโปรแกรมการเขียน subroutine เขียนอาร์เรย์ x และ y ควรมีขอบเขตเดียวกัน (และขนาด); Fortran 77[edit] โปรแกรม writefloats จำนวนเต็ม i ความแม่นยำคู่ x(4), y(4) ข้อมูล x /1d0, 2d0, 4d0, 1d11/ ทำ 10 i = 1, 4 y = sqrt(x) 10 เปิดต่อไป (หน่วย = 15, ไฟล์ ='two_cols.txt', สถานะ='new') เขียน(15, '(f20.3,f21.4)') (x(i), y(i), i = 1, 4) สิ้นสุด FreeBASIC[แก้ไข] ' FB 1.05.0 Win64 Dim x(0 ถึง 3) เป็น Double = {1, 4) สิ้นสุด FreeBASIC[แก้ไข] ' FB 1.05.0 Win64 Dim x(0 To 3) เป็น Double = {1, 4) 2, 3, 1e11}ติมซ้ำ y(0 ถึง 3) เป็นคู่ = {1, 1.4142135623730951, 1.7320508075688772, 316227.76601683791} เปิด เอาต์พุต.txt สำหรับเอาต์พุตเป็น #1For i เป็นจำนวนเต็ม = 0 ถึง 2 พิมพ์ #1 โดยใช้ #; x(i); พิมพ์#1, Spc(7); Use #.###;y(i)Type Next #1, Use #^^^^; x(3); Type #1, Spc(2); Using ##.####^^^^;y(3)Close #1 Output Content .txt : 1 1.0000 2 1.4142 3 1.7321 1E+11 3.1623E+05 Go[edit] Main import package (fmt os) var (x = []float64{1, 2, 3, 1e11} y = []float64{1, 1.4142135623730951, 1.7320508075688772, 316227.76601683791} xprecision = 3 yprecision = 5) func main() { if len(x) !len= (y) { Println(x, y length, y length Back } f, err:= os create (file name) if err != nil { fmt Println(err) returns } for i:= range x { fmt Fprintf(f, %.*e, %.*e, xprecision-1, x[i], yprecision-1, y[i]) } f.Close()} File contents: 1.00e+00, 1.0000e+00 2.00e+00, </1.00></-> </1.00></-> 1.7321e+00 1.00e+11, 3.1623e+05 Haskell[edit] Probably not very idiomatic but oh well import System.IOimport Text.Printfimport Control.Monad writeDat filename x y xprec yprec = withFile filename WriteMode $ \h -> -- Haskell's printf doesn't support a precision given as an argument for some reason, so we insert it into the format manually: let writeLine = hPrintf h $ %. ++ show xprec ++ g\t%. ++ show yprec ++ g in zipWithM_ writeLine x y Example usage Prelude> let x = [1, 2, 3, 1e11] Prelude> let y = map sqrt x Prelude> y [1.0,1.4142135623730951,1.7320508075688772,316227.7660168379] Prelude> writeDat sqrt.dat x y 3 5 Prelude> readFile sqrt.dat >>= putStr 1.00 1.00000 2.000 1.41421 3.000 1.73205 1.000e11 316227.76602 Alternative solution without Printf import System.IOimport Control.Monadimport Numeric writeDat filename x y xprec yprec = withFile filename WriteMode $ \h -> let writeLine a b = hPutStrLn h $ showGFloat (Just xprec) a ++ \t ++ showGFloat (Just yprec) b in zipWithM_ writeLine x y HicEst[edit] REAL :: n=4, x(n), y(n)CHARACTER :: outP = ทดสอบ.txt OPEN(FIle = outP)x = (1, 2, 3, 1E11)y = x ^ 0.5DO i = 1, n WRITE(FIle=outP, Format='F5, F10.3') x(i), y(i)ENDDO Alternative: Show or edit an array formatted in a dialog box like a spreadsheet with a common scroll bar. More - Export - Write array files formatted to file: DLG(text =x, format ='i12', edit =y, format ='F10.2', Y =0) icons and Unicon[edit] The following works in both languages. Main Link Printing Steps() All enter (x := [], (1 to 3) | x[i := 1 to *], y[i]) ends the file name content after running: ->File name 1.00e0 1.0000e0 2.00e0 1.4142e0 3.00e0 1.732 1e0 1.0 0e+11 3.1623e+5 -> IDL[edit] Info: x = [1,2,3,1e11] y =sqrt(x) xprecision=3 Depreciation =5; I /O file: openw,unit,datafile.txt,/get for i = 1L, n_elements(x) make printf, x[i-1],y[i-1], format=form=form free_lun, datafile file unit.txt 1 12 1.41423 1.73211E+011 3.1623E+005 This is quite ugly and un-IDLish, for example, should not rely on x and y of the same size, and if the data is exported in a human-readable format, it should be lined up more beautifully. And if it must be in two-column format with x and y side by side, we may consider running ASCII_Template or partly instead of formatting ugly hands J[edit] like 'file' NB for fwrites x = 1 2 3 1e11y =%: x NB y is sqrt(x) xprecision = 3ycision = 5 = file name 'whatever.txt= (0jprecision, Depreciation) : x,,y data fwrites File name or more compact: ((0 j. 3 5): (,.%:) 1 2 3 1e11) fwrites This loses all inline comments and names, and instead relies on the reader's understanding of the purpose of each name (for example: 3 is the precision of the first column, and 5 is the precision of the second column). Please note that the idea of j with precision here is the position behind the decimal point: (0 j. 3 5): (,.%:) 1 2 3 1e11 1.000 1.00000 2.000 1.41421 3.000 1.7320510000000000.000 316227.76602[edit] นำเข้า java.io*; คลาสสาธารณะ FloatArray { สาธารณะคงที่โมฆะเขียนDat (ชื่อไฟล์สตริง, คู่ [] x, คู่ [] y, int xprecision, int yprecision) โยน IOException { assert x.length == y.length; PrintWriter ออก = PrintWriter ใหม่ (ชื่อแฟ้ม); สำหรับ (int i = 0; i < x.length;= i++)= out.printf(%.+xprecision+g\t%.+yprecision+g,= x[i],= y[i]);= out.close();= } = public= static= void= main(string[]= args)= {= double[]= x={1, 2,= 3,= 1e11};= double[]= y=new double[x.length];= for= (int= i=0; i=> < x.length;= i++)= y[i]=Math.sqrt(x[i]); try= {= writedat(sqrt.dat,= x,= y,= 3,= 5);= }= catch= (ioexception= e)= {= system.err.println(writedat:= exception:= +e);= } = try= {= bufferedreader= br=new bufferedreader(new= filereader(sqrt.dat));= string= line;= while= ((line=br.readLine()) !=null) system.out.println(line);= }= catch= (ioexception= e)= {= }= }= joy[edit]= define= write-floats==['g 0]= [formatf]= enconcat= map= rollup['g= 0]= [formatf]= enconcat= map= swap= zipfilename= w= fopen= swap[[fputchars]= 9= fputch]= step= 10= fputch]= stepfclose.= using= it:= [1.0= 2.0= 3.0= 1e11]= 3= [1.0= 1.41421356= 1.73205080= 316227.7660168]= 5= write-floats.= jq[edit]= program:= [1,= 2,= 3,= 1e11]= as= $x|= $x= |= map(sqrt)= as= $y|= range(0;= $x|length)= as= $i|= \($x[$i]) = \($y[$i])= execution:= to= write= the= output= to= filename:= $= jq= -n= -r= -f= write_float_arrays_to_a_text_file.jq=> ชื่อแฟ้ม Julia[edit] xprecision = 3yprecision = 5x = รอบ(.) [1, 2, 3, 1e11],xprecision)y = รอบ ([1, 1.4142135623730951, 1.7320508075688772, 316227.76601683791],yprecision)writedlm(filename, [x y], '\t') Kotlin[edit] // version 1.1.2 import java.io.File fun main(args: Array<String>) { val x = doubleArrayOf(1.0, 2.0, 3.0, 1e11) val y = doubleArrayOf(1.0, 1.4142135623730951, 1.7320508075688772, 316227.76601683791) val xp = 3 val yp = 5 val f = %.${xp}g\t%.${yp}g val writer = File(output.txt).writer() writer.use { for (i in 0 until x.size) { val s = f.format(x[i], y[i]) writer.write(s) } }} Contents of 'output.txt': 1.00 1.0000 2.00 1.4142 3.00 1.7321 1.00e+11 3.1623e+05 Lingo[edit] on saveFloatLists (filename, x, y, xprecision, yprecision) eol = numtochar(10) -- LF fp = xtra(fileIO).new() fp.openFile(tFile, 2) cnt = x.count repeat with i = 1 to cnt the floatPrecision = xprecision fp.writeString(string(x[i]) fp.writeString(TAB) the floatPrecision = yprecision fp.writeString(string(y[i]) fp.writeString(eol) end repeat fp.closeFile()end x = [1.0 , PI, sqrt(2)]y = [2.0, log(10), x, y, 3,</String> 3,</String> File name Lua[edit] = file .txt x = { 1, 2, 3, 1e11 }y = { 1, 1.4142135623730951, 1.7320508075688772, 316227.76601683791 };xprecision = 3;yprecision = 5; fstr = %.tostring(xprecision) f....tostring(guessing) ffp = io.open (file name, w+) For i = 1, #x do fp:write (string.format(fstr, x[i], y[i])) ends io.close (fp) Mathematica[edit] exportPrec[path_, data1_, data2_, prec1_, prec2_]:=Export[path,transpose[{Map[ToString[numberForm[#, prec2]] & data2],Map[ToString[NumberForm[#, prec1]],&data1]}, TABLE] MATLAB / Octave[edit] x = [1, 2, 3, 11]] y = [1, 1.4142135623730951, 1.732050807568772, 316227.7601683791]; fid = fopen ('file name', 'w') fprintf(fid,'%.3g\t%.5g', [x;y]); fclose(fid); Module write_float_arrays.:-Interface :- import_module Io.-PRED Core(IO: :d i, IO:uo) is det.:- Operation :- import_module float, entry, math, string of the main(! io.open_output IO), (OpenFileResult = ok(File), X = [1.0, 2.0, 3.0, 1e11], list.foldl_corresponding(write_dat(File, 3, 5), X, Map (sqrt, X), IO), io.close_output (file, !IO) ; Errors (IO_Error), io.stderr_stream (Stderr, !IO), io.format(Stderr, Errors: %s, [s(io.error_message(IO_Error)., !IO), io.set_exit_status(1,! (write_dat text_output_stream::in, int::in, int::in, float::in, float:in, float:in, io::d i, io:. write_dat (file, XPrec, XPrec, X, Y, Y, IO) :- io.format(file, %.*g\t%.*g, [i(XPrec), f(X), i(YPrec), f(Y)], IO) File

contents: 1 1 2 1.4142 3 1.7321 1e+11 3.1623e+05 NetRexx[edit] /* NetRexx */ ตัวเลือกแทนที่รูปแบบความคิดเห็น java crossref บันทึกสัญลักษณ์ nobinary -- คิดค้นชื่อไฟล์ข้อความเป้าหมายตามแหล่งที่มาของไฟล์ต้นฉบับของโปรแกรมนี้ . pgmName '.nrx' .outFile = pgmName || '.txt' do formatArrays(outFile, [1, 2, 3, 1e+11], [1, 1.4142135623730951, 1.7320508075688772, 316227.76601683791])catch ex = Exception ex.printStackTraceend return -- ~~~