STEM Society Meeting, February 13, 2018

James Emery

Last Edit: 2/17/2018

Contents

1 About the STEM Society and the STEM Society Website

STEM is an abbreviation for Science, Technology, Engineering and Mathematics. The acronym STEM is commonly associated with K-12 education, but our use of the term is only slightly bound to this meaning. There are over one hundred people on the mailing list, although a much smaller group attends any one meeting. We meet on the second Tuesday of each month at the Trailside Center at 99th and Holmes in Kansas City, Missouri. The meetings are open to all. The start time is 6PM. We make presentations, have discussions, and have demonstration experiments. These relate to Science, the History of Science, Mathematics, Engineering, Philosophy and Technology at all levels. The topics have ranged from a technical discussion of the Mathematics of General Relativity to scientific experiments for young students.

These meeting notes contain links to many other documents, which may be viewed or downloaded by clicking the link. A partial list of documents can be reached by clicking the heading **Documents**. The meeting notes may also be viewed in an archive file (archive.pdf), which is in the list of documents. Many of the documents are PDF files. They may be viewed or downloaded to the computer by clicking, provided Adobe Reader, or another program capable of reading PDF files, is present. There are many more documents available at the site than are listed under **Documents** because the documents.htm file is not at all up to date. The last time I checked, about March 2014, there were about 350 document files on the site. We are in the process of creating better techniques for finding documents and authors. The first meeting of the STEM Society was in November of 2006.

The web site is:

http://www.stem2.org/

Direct to the documents list:

http://www.stem2.org/je/documents.htm

Direct to the archive file:

http://www.stem2.org/je/archive.pdf

2 The February 13, 2018 Meeting Announcement

The February meeting of the STEM Society will take place on the second Tuesday of the month, February 13, 2018, at the Trailside Center at 99th and Holmes in Kansas City, Missouri. The starting time is 6PM. Also look at our website for past meeting notes:

The website is:

http://www.stem2.org/

Topics and Discussions:

(a) Bob Kessler will talk about local Kansas City activity in STEM education (Science, Technology, Engineering, and Mathematics). We may also talk about opposition to this kind of education, (which recalls C. P. Snow's old book "The Two Cultures").

(b) Jim Emery will talk about something in the areas of: Computer Programming, Linear Algebra, accessing material on the STEM Society website, Physics: the Mechanical Universe from Caltech (which has been made available on the internet within the last couple of years through YouTube), the Millikan oil drop experiment, the difference between little girls and little boys, punishment fitting the crime, advice from Carl Sagan about the decay of knowledge in the United States, and the local use of stemdocs software.

(c) If there is leftover time we can talk about other topics. So if you have something of interest you want to talk about, by all means bring it in.

3 About Frankenstein in Science Magazine

science 12 JANUARY 2018 VOLUME 359 ISSUE 6372

http://www.sciencemag.org/news/2018/01/long-shadow-frankenstein

http://science.sciencemag.org/content/sci/359/6372.toc.pdf

CREDITS: (LEFT TO RIGHT) UNIVERSAL
PICTURES/SCREENPROD/PHOTONONSTOP/ALAMY STOCK PHOTO
NASA; H. FORD (JHU); AUTZEN
ET AL.

12 JANUARY 2018 VOL 359 ISSUE 6372 131

12 JANUARY 2018 VOLUME 359 ISSUE 6372

CONTENTS 160, 228, & 237

How a channel senses calcium

ON THE COVER

The cover illustration reimagines the monster first
conjured in Mary Shelleys 1818 novel.
Two centuries later, the story of how the fictional
Dr. Frankenstein built a
creature from scavenged body parts and unleashed it
on the world has
lost none of its power to unsettle, and Frankenstein
has become a byword
for scientific overreach. See pages 137 and 146.

Illustration: Craig and Karl

INTRODUCTION 146
After 200 years, Frankenstein
is still essential reading for scientists
By K. Kupferschmidt FEATURES 148

4 The Mechanical Universe

Here is a link to finding the Mechanical Universe lectures:

`http://bit.ly/2gvNAA3 on youtube.`

There is a wikipedia article about the Mechanical Universe titled

`https://en.wikipedia.org/wiki/The_Mechanical_Universe`

You can also reach this using Google. I often download and store wikipedia articles as pdf files. My copy of this one, which is on my Toshiba laptop is the file:

`c:\je\pdf\TheMechanicalUniverse.pdf`

This pdf contains a listing of all of the 52 episodes, each about 30 minutes in duration.

5 The Oil Drop Experiment Episode

Robert Millikan (March 22, 1868 - December 19, 1953) was a physicist who performed the oil drop experiment to measure the very tiny charge on the very tiny electron, for which he won the Nobel prize.

This episode, number 12 of the Mechanical Universe, is quite interesting because it is somewhat incredible how Millikan was able to measure the extremely small charge of the electron, with rather crude apparatus. He did this work in the period from 1908 -1910 at the University of Chicago. He received the Nobel Prize in Physics for this in 1923. He moved to Cal-Tech in 1921. Some text books on the Mechanical Universe were also published. I have one:

[1] Frautschi Steven C, Olenick Richard P., Apostol Tom M., and Goodstein David L., **The Mechanical Universe: Mechanics and Heat, Advanced Edition**, Cambridge University Press, 1986.

The **Oil-Drop Experiment** occurs in Chapter 8, in sections 8.8 through 8.9, pages 192 - 202.

6 Carl Sagan Quote from his Book: *The Demon Haunted World*

"I have a foreboding of an America in my children's or grandchildren's time ... when awesome technological powers are in the hands of a very few, and no one representing the public interest can even grasp the issues; when the people have lost the ability to set their own agendas or knowledgeably question those in authority; when clutching our crystals and nervously consulting our horoscopes, our critical faculties in decline, unable to distinguish between what feels good and what's true, we slide, almost without noticing, back into superstitions and darkness."

Carl Sagan

[1] Sagan Carl, *The Demon Haunted World.*

7 Little Girls and Little Boys

In the current questioning atmosphere about the behavior of girls and boys let us consider the old nursery rhyme:

```
Here's a long version of "What Are Little Girls Made of?"
as found in The Baby's Opera by Walter Crane (circa 1877):

1. What are little boys made of?
What are little boys made of?
Frogs and snails and puppy-dog's tails,
And that are little boys made of.

2. What are little girls made of?
What are little girls made of?
Sugar and spice and all that's nice,
And that are little girls made of.

3. What are young men made of?
What are young men made of?
Sighs and leers, and crocodile tears,
And that are young men made of.

4. What are young women made of?
What are young women made of?
Ribbons and laces, and sweet pretty faces,
And that are young women made of.
```

From Wickipedia, *"Walter Crane (15 August 1845 14 March 1915) was an English artist and book illustrator. He is considered to be the most influential, and among the most prolific, childrens book creators of his generation and, along with Randolph Caldecott and Kate Greenaway (See the Randolph Caldecott Medal and the Kate Greenaway Medal), one of the strongest contributors to the child's nursery motif that the genre of English children's illustrated literature would exhibit in its developmental stages in the latter 19th century."*

He is likely not the original creator of the original nursery rhyme, of which I have not found the original source.

8 The Punishment Must Fit The Crime

For justice to be served, "The punishment must fit the crime." This is either a principle of English common law, or something written by Gilbert and Sullivan in an operetta, at any rate it is true. Perhaps this idea was argued in a Platonic dialogue?

A blatant examples of the failure of this principle of justice, includes

1. Hanging young children pickpockets in London in the 19th century, in order completely to do away with the irritant of children pickpockets.

2. Executing male dogs (and some females), for the crime of humping human legs.

3. Sentencing those found with possessing a given amount of Marijuana to a mandatory twenty year sentence in prison, in order to wipe out Marijuana use in the United States. These people were being sentenced for being guilty of all Marijuana use, which is not the crime they have committed.

4. The killing of hundreds of drug users in the Philippines, directed by president Rodrigo Duterte of the Philippines, for being individually guilty for all drug use, not just for their own drug use.

Other blatant examples of lack of justice might be the Blacklisting of writers in the movie industry in the 1950's, who were accused of communistic sympathies, not to mention the Holocaust where millions of people were accused and executed for being Jewish, and then there is the murdering of millions of people in Russia by Stalin for being counterrevolutionary, whatever that happened to be in his opinion. Also there is the current craze for the summary execution of some people for driving while "black."

In order to help guarantee that "The Punishment Fits the Crime" many civil societies have instituted an institution known as "Trial by Jury," but often ignored.

The problem of bosses in general is a big one in many ways, it is a dictatorship, in companies and corporations as well as in countries. The queen of hearts was not only humorous, but a serious pest. Firings and beheadings are frequently not nice at all, whether by a dictator or by misguided popular opinion. Remember the old advice of Bertrand Russell's grandmother, "Do not follow a crowd to do evil!"

9 A Python Program for Those Who Drink and Then Want to go Home

Consider a file **f.txt** containing the following lines:

```
abcdefghijklmnopqrstuvwxyz
314159265358979
2718281828459045
mississippi
amanaplanacanalpanama
```

We have written a python program that is insane, because **reversefile.py** is its name.

We can run this program typing just like this:

```
c:\python reversefile.py f.txt
```

And it creates this:

```
 number of arguments =  2
 command line arguments  ['reversefile.py', 'g.txt']
zyxwvutsrqponmlkjihgfedcba
979853562951413
5409548281828172
ippississim
amanaplanacanalpanama
```

If you wonder why it creates this, here is why:

```
#reversefile.py read a file, each line as a string 1/20/2018
#data file f.txt
import sys
#print sys.argv
#print sys.argv[0]
#
#function to reverse characters in string
def reverse(s):
 result=""
 for c in s:
  result = c + result
 return result
#
na=len(sys.argv)
print ' number of arguments = ', na
print ' command line arguments ',sys.argv
if( na < 2):
 print "reversefile.py, read all file lines"
 print "and reverse each string line read"
 print "Sample data file: f.txt"
 print "Usage: python reversefile.py file "
 sys.exit(0)
f1=open(sys.argv[1],'r')
n=0;
lns=0
for a in f1:
 lns=lns+1
 # s.strip(None) stips white space from the end of string s
 #  including the extra newline read from the string
 #  prevents an extra blank line between prints
 #  See Lee python book, page 81, and Appendix C
 print reverse(a).strip(None)
#print lns," number of lines read"
f1.close
```

After some practice memorizing such reversals, you will pass the tests, and return home to feed your children ice cream, rather than to force them to in-

terrupt their play and visit you in San Quentin each month where you never give them ice cream at all.

The program for reversing a string shows several manipulations of strings. A string can be defined by assigning a variable s to text, which is delimited by a pair of either double quotes or single quotes. Two strings can be concatenated with a plus sign. A string s can be indexed with an integer as in $s(5)$, which is the sixth character of s because indexing starts with zero. The function len gives the number of characters in a string. A character can be added or appended to a string in the same way that two strings are concatenated as in adding a character of string (s), to the string t, as in $t = t + s(j)$. A range is a set of integers from a starting integer to an ending integer. Thus range $(0, n)$ is the set of n integers $\{0, 1, 2, 3, ..., n-1\}$. Notice that the command **for a in f1**, a loop uses the word "in," so we shall continue to increment the loop as long as the string "a" is found in the file "f1." So a "for" marches through a sequence of some kind. Here each line in the file is judged to be a string, and thus a file is read as list of strings. The loop is terminated when the last line is found in the file, that is the last string in the list.

10 James Emery: Matrix 101

stem2.org/je/matrix.pdf

11 James Emery: Running the StemDocs Software

I have constructed a directory that you can install on your computer that contains programs to search the file called **stemdocs.txt** for topics, and the meeting notes. The directory will contain the following with perhaps some additions:

```
Directory of c:\stemcalc

02/13/2018  11:38 AM                 0 out1
02/13/2018  11:38 AM                 0 outxx
02/11/2018  10:10 AM         5,292,980 stemsocnotesfiles.zip
02/11/2018  09:55 AM    <DIR>          .
02/11/2018  09:55 AM    <DIR>          ..
02/11/2018  09:42 AM               396 tmp2.txt
02/11/2018  09:42 AM               392 tmp1.txt
01/09/2018  04:45 PM             3,718 out2
01/09/2018  04:08 PM               843 stemdocsx.bat
01/08/2018  10:49 PM            44,577 stemdocs.txt
01/07/2018  10:31 PM               119 out
01/03/2018  05:10 PM            29,255 stemsoc010918.pdf
09/02/2011  11:38 PM            56,320 semi2nlx.exe
11/18/2010  09:27 PM           587,776 7za.exe
02/15/2010  10:18 PM               105 fl.bat
09/28/1999  08:50 AM            49,152 grepx.exe
              14 File(s)      6,065,633 bytes
               2 Dir(s)  84,190,281,728 bytes free
```

I call the directory stemcalc because if you were to put this directory on a flash drive, you could insert and run programs on the flash drive without disturbing the hard drive of the computer. Perhaps you could install python

on the same flash drive for the same reason. So perhaps this is a way we could run some examples on the Trailside computer.

Program **stemdocsx.bat** contains the following commands:

```
@echo off
rem stemdocsx.bat, Version 1/7/2018, by Jim Emery
rem lists those lines in file stemdocs.txt containing specified words
if "%1"=="" goto help
grepx -i %1 stemdocs.txt > tmp1.txt
copy tmp1.txt tmp2.txt
:search
shift
if "%1"=="" goto list
grepx -i %1 tmp2.txt > tmp1.txt
copy tmp1.txt tmp2.txt
goto search
:list
semi2nlx tmp1.txt tmp2.txt
type tmp2.txt | more
goto end
:help
echo stemdocsx.bat, by jim emery, Version 1/7/2018
echo Lists file names, and titles, for documents on the stem society website: stem2.org
echo Prints lines in the file stemdocs.txt that contain all strings entered on the command line
echo a semicolon in a printed line causes a conversion to a newline character and so
echo converts a single found line to printed multiple lines
echo Usage: stemdocsx string1 string2 string3 ...
:end
```

The software will be written to your flash drive from which the software is run, or copied to a directory from the flash drive, or downloaded with a zip file found on the our website. I will have to see how it all works on various computer setups, using different versions of Windows.

This is intended to be educational, not just useful. I want people to do things. Don't be lazy!

Note: Capitalisation is ignored in searching.

Example 1. *Find the document for the meeting of January 2018*

```
c:\stemcalc\stemdocs january 2018

stemsoc010918.pdf January 2018
 Marc Robinson: Language Death and Documentation,
 Carl Sagan Quote
 Litle Girls and Little Boys
 James Emery: The Punishment Must Fit The Crime
 James Emery: A Python Program for Those Who Sometimes Find
```

```
 Themselves Driving While Slightly Drunk
 James Emery: Matrix 101
 James Emery: Running the StemDocs Software
 James Emery: Topics in Linear Algebra and Its Applications
```

Example 2. *Find the meeting notes when Tom Grant talked, or is referenced.*

```
 c:\stemcalc\stemdocsx tom grant

stemsoc011315.pdf Title: Stem Society Meeting, January 13, 2015
 Tom Grant Solar Panel Leasing
stemsoc061014.pdf Title: Stem Society Meeting, June 10, 2014
 Tom Grant: the Keystone XL Pipeline oil, Emery Cryptography RSA Algorithm
 books: Stuff Matters
 Miodownnik Mark, {\bf Stuff Matters}, 2013, Houghton-Mifflin-Harcourt.,
 article make mag. 36 Nuclear Fuser
stemsoc081313.pdf Title: Stem Society Meeting, August 13, 2013
 Tom Grant Nuclear Energy Fukushima, Bob Kessler books Cerner
 Rich Kaufman Natural Gas Fracking, Chris Sanderson Immanual Kant
 Steve Cummings Peltier Cooling, Quantum Mechanics and Semiconductors
stemsoc120914.pdf TiTle: STEM Society Meeting, December 9, 2014
 Ryan Rozzelli: 3D Laser Scanner Demonstration
 Tom Grant youtube Which falls first? The Ball Or The Feather? Human Universe, gra
stemsoc060915.pdf
 June 2015
 Tom Grant: Paleontology
 Jim Emery: Some Differential Equations of MathematicalPhysics
 Tom Grant: Power Point Document on Solar Leasing Presented January 2015
 Steve Cummins: Locally Collected Rocks With Embedded Fossils
stemsoc090815.pdf
 September 2015
 Rick Hines: The Autopsy of a Disk Drive
 Rick Hines: 3.5 Million Year Old Coral Fossel from Truman Lake
 Discussion of Energy and Nuclear Reactors, Led by Tom Grant
 Rich Kaufman: Discussion of an Article on the Proposed Iranian Nuclear Arms Deal
 Bob Kessler: K.C., Me, Wen Ho Lee, and Ye
```

James Emery Review: The Volterra Chronicles:
The Life and Times of an Extraordinary Mathematician 1860-1940
James Emery: The Mystery of Matter: the Search for the Elements
stemsoc120815.pdf
December 2015
Physics Experiment Demonstrations
Tom Grant: An Introduction to Nuclear Reactor Engineering
Dirac Biography
Discussion: Why Does Everyone Hate Mathematics?
Historical Adventures in Arithmetic: Casting Out Nines
Lehrer on Lobachevski
Bibliography
stemsoc011216.pdf
January 2016
Tom Grant: A History of Nuclear Weapons
Review by Rich Kaufman: The Manga Guide to Relativity
Jim Emery Review of A Biography of Paul Dirac
Jim Emery: Historical Adventures in Arithmetic: Casting Out Nines
Bibliograph
stemsoc020916.pdf
February 2016
Tom Grant: Isotope Tales
Polonium 210
Poisoning of Alexander Lituinenko
Bob Kessler, Discussion of the Book: The Virus of the Mind
stemsoc071216.pdf
July 2016
James Emery: Some Mathematics
James Emery: Book Review, "Visions of Technology," by Richard Rhodes Editor, 199
James Emery: Book Review, "The Bourbaki Gambit," by Carl Djerassi, 1994
Future Talk on Thermodynamics
James Emery: Kansas City Maker Faire 2016
James Emery: The Johnson County Maker Space
Rich Kaufman: CRISPR
Tom Grant: The Vanishing of the Anasazi Indians
John Gamble, Theta Functions
References
stemsoc091316.pdf

```
 September 2016
 2016 Spencer Award, American Chemical Society
 Rich Kaufman: Suggestions for Meeting Topics
 Jim Emery: Introduction to the Concept of the Potential Function
 Tom Grant: The Gallinagenocide
stemsoc031417.pdf
 March 2017
 Tom Grant: Riding or Pushing, Did Sisyphus Need a Bicycle?
 Rich Kaufman Book Discussion "The Vaccine Race," The Hayflick Limit
stemsoc091217.pdf
 September 2017
 Tom Grant: Extraordinary Popular Equations and the Madness of Crowds (tatoo)
 Cecile Lagandre: The Formation of the Driftless Area
 Jennett Conant Lecture Linda Hall Library
 Dava Sobel Lecture Linda Hall Library
stemsoc121217.pdf
 December 2017
 James Emery: Elementary Algorithms in Python,
  Division of arbitrarily Long Integers, School Square Root Algorithm
 Tom Grant: Dark Zone Archeology (in caves), Power Point Slides
```

12 James Emery: Topics in Linear Algebra and Its Applications

stem2.org/je/lineara.pdf

13 Python for Scientific Computing

Here is a link to my document called **Scientific Programming in Python**

stem2.org/je/python.pdf

13.1 Integers

There are traditional integers in Python as in most languages, but there are also long integers, which can be of any length (within memory limits). Now

ordinary short integers, that is traditional integers, which used to be 32 bits, will not overflow in Python but rather will be converted to the unlimited length integers automatically. Such integers make doing number theory calculations in python very possible.

modulo operator The modulo binary operator mod, used as in $k \bmod n$, is the remainder of k/n. This is the operator % in Python, so that $k \bmod n$ is $k \,\%\, n$ in python. So if $k \,\%\, n = 0$, then n divides k.

13.2 Example: Greatest Common Divisor

The gcd.py program uses command line parameters. These are parameters that are passed to a program from the command line. For example if you run the program gcd.py with the command

```
python gcd.py 7142858 999999
```

then 7142858 and 999999 are the command line parameters and gcd.py computes the gcd as

```
142857
```

Many programs are written so that if you run the program with no parameters, then you are presented with the help information about what the program does and what the parameters are.

The gcd of two integers is the largest integer that divides both of them.

```
#gcd.py greatest common divisor, getting m and n from the command line
# latest version Aug 28,2017
# May 30, 2014
#see gcdexample, m and n defined in the program
import sys
# python gcd9.py 2010 560

#function greatest common divisor
def gcd(b1,b2):
 b=[]
 k=[]
 rm=[]
 v=[]
 v=[1,2,3]
 b.append(0)
 b.append(b1)
```

```
 b.append(b2)
 k.append(0)
 rm.append(0)
 n=0
 while (b2 > 0):
  n=n+1
  #print " b1= ",b1
  #print " b2= ",b2
  g=b2
  q=b1/b2
  k.append(q)
  #note, if b2>b1 then q=0, then r=b1 so b1 and b2 are switched
  #print " q= ",q
  r=b1-b2*q;
  b.append(r)
  print " ",b1,"=",q,"*",b2,"+",r
  rm.append(r)
  #print " r= ",r
  b1=b2
  b2=r
 v[0]=g
 #print
 #print " number of divisions n=",n
 #print " b="
 #for i in range(1,len(b)):
  # print " b[",i,"]=",b[i]
 #print " length of k=",len(k)
 #print " quotients=",k
 #for i in range(1,len(k)):
  # print " k[",i,"]=",k[i]
 #print " remainders=",rm
 #for i in range(1,len(rm)):
  # print " rm[",i,"]=",rm[i]
 alpha=1
 beta=-k[n-1]
 #print " g=alpha*b[n-1]+beta*b[n]=",alpha*b[n-1]+beta*b[n]
 p=n-2
 while (p>0):
  tmp=alpha
  alpha=beta
  beta=tmp-beta*k[p]
  #print "p=",p
  #print " g=alpha*b[p]+beta*b[p+1]=",alpha*b[p]+beta*b[p+1]
  p=p-1
 #print
 #print " alpha=",alpha," beta=",beta
 #print " g=alpha*b[1]+beta*b[2]=",alpha*b[1]+beta*b[2]
 v[1]=alpha
 v[2]=beta
 return v

#function least common multiple
def lcm(m,n):
 v=m*n/gcd(m,n)
 return v

#main
```

```
#print sys.argv
#print sys.argv[0]
na=len(sys.argv)
#print 'number of arguments = ', na
if( na < 3):
 print "gcd.py, greatest common divisor of two integers"
 print "Also prints multipliers alpha and beta of m and n"
 print "that produce the gcd."
 print "as well as m/gcd and n/gcd."
 print "Usage: python gcd.py m n "
 sys.exit(0)
m=int(sys.argv[1])
print " m= ",m
n=int(sys.argv[2])
print " n= ",n
v=gcd(m,n)
#print " length v = ",len(v)
print " v = [gcd,alpha,beta]=",v
print " gcd(m,n)= ",v[0]
alpha=v[1]
#print " alpha=",alpha
beta=v[2]
#print " beta=",beta
print " alpha*m+beta*n= ",alpha*m+beta*n
print " m/gcd=", m/v[0]
print " n/gcd=", n/v[0]
```

Running the gcd.py program without parameters generates some help information:

```
gcd.py, greatest common divisor of two integers
Also prints multipliers alpha and beta of m and n"
that produce the gcd.
as well as m/gcd and n/gcd.
Usage: python gcd.py m n
```

Now we run with two integers specified:

```
$ python gcd.py 200560490130 3451796661527379
m=  200560490130
n=  3451796661527379
gcd(m,n)=  9582441
lcm(m,n)=  72246104125768042470
m= m/gcd = 20930
n= n/gcd = 360221019
gcd(m,n) =  1
```

13.2.1 Example For Explaining the Algorithm

Here is a **gcd.py** printout of a simple example for explaining how the algorithm works.

```
m=  1638
n=  660
 1638 = 2 * 660 + 318
 660 = 2 * 318 + 24
 318 = 13 * 24 + 6
 24 = 4 * 6 + 0
v = [gcd,alpha,beta]= [6, 27, -67]
gcd(m,n)=  6
alpha*m+beta*n=  6
m/gcd= 273   (3)(7)(13)
n/gcd= 110 = (2)(5)(11)
```

13.2.2 My Modulo Program mod.py

The **greatest integer function**, given a real value x, is written $[x]$ and equals the largest integer less than or equal to x. For example $[5.3] = 5$, and we have the repeating decimal $-17.0/7 = -2.42857142857142857...$, so $[-17.0/7] = -3$. Now computer languages return the greatest integer for division of two integers. Thus you will find in Python, and computer languages in general, one hopes, that they return the value -3 for $-17/7$.

Here is my program **mod.py**:

```
# mod.py m mod(n) program using function mod(m,n) 6-3-14
# 2-9-2018 added comparison with built in python mod function
import sys
#+ function m mod n
def mod(m,n):
 q=m/n
 r=m-q*n
 print m,"=",q,"*",n,"+",r
 return r

#program main mod
#print sys.argv
#print sys.argv[0]
na=len(sys.argv)
#print 'number of arguments = ', na
if( na < 3):
 print "mod.py, compute:  m mod n"
 print "Usage: python mod.py m n "
 sys.exit(0)
m=int(sys.argv[1])
print " m= ",m
n=int(sys.argv[2])
print " n= ",n
v=mod(m,n)
print m," mod ",n ," = ",v
print " using python function m % n =", m % n
```

Here are two examples of running the program, computing 17 mod 7, and computing -17 mod 7:

```
c:\je\py\python mod.py 17 7
 m=  17
 n=  7
17 = 2 * 7 + 3
17  mod  7  =  3
 using python function m % n = 3
```

and

```
c:\je\py\python mod.py -17 7
 m=  -17
 n=  7
-17 = -3 * 7 + 4
-17  mod  7  =  4
 using python function m % n = 4
```

14 Index