

```

#####
#
# A simple example of a Maple 'source' file. Such a file can
# be easily created and maintained using your favorite text
# editor and can contain arbitrary Maple commands.
# I find this mechanism particularly useful for developing
# and maintaining Maple procedures.
#
# The file can most easily be read into a Maple session by
# first 'cd'-ing to the directory which contains this file:
#
# % cd /Public/Members/matt/Src/maple/examples
#
# starting up maple or xmaple
#
# % xmaple &
#
# then typing
#
# > read example;
#
# If maple (or xmaple) isn't running in the directory
# containing this file, then you must use an absolute
# pathname and be sure to enclose the name in backquotes.
# (This also applies to filenames containing a '.', which
# is why I tend to use simple names (no extensions) for
# files containing Maple source.
#
# > read '/Public/Members/matt/Src/maple/examples/example';
#
# Recall that use of the colon (:) as terminator rather
# than semi-colon (;) inhibits echoing of results.
#
#####

```

```
aa := 23 / 155;
```

```
myadd := proc(x::numeric, y::numeric)  
    x + y;  
end;
```

```

#####
# ladd: Adds all elements of a list.
#
# For illustrative purposes only---procedure could more
# compactly defined as
#
# ladd := proc(l::list)
#   local i;
#   add(l[i],i=1..nops(l))
# end;
#####
ladd := proc(l::list)

#-----
# Define local variables.
#-----
    local lsum, i;
#-----
# Check for valid argument, exit with error message
# if not valid.
#-----
    if nops(l) = 0 then
        ERROR('argument is the NULL list');
    fi;
#-----
# Initialize sum to first element of list.
#-----
    lsum := l[1];
#-----
# Loop over rest of elements accumulating the sum.
#-----
    for i from 2 to nops(l) do
        lsum := lsum + l[i];
    od;

```

```
#-----  
# Return the sum.  
#-----  
    lsum;  
end:
```

```

#####
# Manipulations with power series
#
# Following "Handbook of Mathematical Functions"
# Abramowitz and Stegun (A & S)
# Section 3.6, Infinite Series
#####

s[1] := 1 + a[1]*x + a[2]*x^2 + a[3]*x^3 + a[4]*x^4:
s[2] := 1 + b[1]*x + b[2]*x^2 + b[3]*x^3 + b[4]*x^4:
s[3] := 1 + c[1]*x + c[2]*x^2 + c[3]*x^3 + c[4]*x^4:

unknowns := {c[1],c[2],c[3],c[4]}:

#####
# Define a 'shorthand' procedure for converting an
# expression (in the current case a series) to a polynomial
#####
P := proc(x::anything) convert(x,polynom) end:

#####
# Solves for coefficients c_i in terms of a_i and b_i.
# 'series_in' should be an expression in s[1] and s[2].
# Note the use of global variables (s[3],unknowns).
#####
series_op := proc(series_in::anything)
    solve({coeffs(P(s[3])) -
           P(series(series_in,x=0,5)),x},unknowns);
end:

#####
# Use 'series_op' procedure to reproduce various A & S
# formulae.
#

```

```

# Note: Due to an apparent bug in 'series' and 'taylor'
# in Maple V.5, the computations of AS_3_6_18 and AS_3_6_19
# must be repeated---the first time through, 'series_op'
# fails, the second time it works.
#####
AS_3_6_16 := series_op( 1 / s[1] ):
AS_3_6_17 := series_op( 1 / s[1]^2 ):
AS_3_6_18 := series_op( sqrt(s[1]) ):
AS_3_6_18 := series_op( sqrt(s[1]) ):
AS_3_6_19 := series_op( 1 / sqrt(s[1]) ):
AS_3_6_19 := series_op( 1 / sqrt(s[1]) ):
AS_3_6_20 := series_op( s[1]^n ):
AS_3_6_21 := series_op( s[1] * s[2] ):
AS_3_6_22 := series_op( s[1] / s[2] ):
AS_3_6_23 := series_op( exp(s[1] - 1) ):
AS_3_6_24 := series_op( 1 + ln(s[1]) ):

```