

# Pari-GP reference card

(PARI-GP version 2.17.3)

Note: optional arguments are surrounded by braces {}.

To start the calculator, type its name in the terminal: **gp**

To exit **gp**, type **quit**, **\q**, or **<C-D>** at prompt.

## Help

describe function	?function
extended description	??keyword
list of relevant help topics	???pattern
name of GP-1.39 function <i>f</i> in GP-2.*	whatnow( <i>f</i> )

## Input/Output

previous result, the result before	%, %`, %`` , etc.
<i>n</i> -th result since startup	% <i>n</i>
separate multiple statements on line	;
extend statement on additional lines	\
extend statements on several lines	{ <i>seq</i> <sub>1</sub> ; <i>seq</i> <sub>2</sub> ;}
comment	/* ... */
one-line comment, rest of line ignored	\\ ...

## Metacommands & Defaults

set default <i>d</i> to <i>val</i>	default({ <i>d</i> },{ <i>val</i> })
toggle timer on/off	#
print time for last result	##
print defaults	\d
set debug level to <i>n</i>	\g <i>n</i>
set memory debug level to <i>n</i>	\gm <i>n</i>
set <i>n</i> significant digits / bits	\p <i>n</i> , \pb <i>n</i>
set <i>n</i> terms in series	\ps <i>n</i>
quit GP	\q
print the list of PARI types	\t
print the list of user-defined functions	\u
read file into GP	\r <i>filename</i>
set debuglevel for domain <i>D</i> to <i>n</i>	setdebug( <i>D</i> , <i>n</i> )

## Debugger / break loop

get out of break loop	break or <C-D>
go up/down <i>n</i> frames	dbg_up({ <i>n</i> }), dbg_down
set break point	breakpoint()
examine object <i>o</i>	dbg_x( <i>o</i> )
current error data	dbg_err()
number of objects on heap and their size	getheap()
total size of objects on PARI stack	getstack()

## PARI Types & Input Formats

t_INT. Integers; hex, binary	±31; ±0x1F, ±0b101
t_REAL. Reals	±3.14, 6.022 E23
t_INTMOD. Integers modulo <i>m</i>	Mod( <i>n</i> , <i>m</i> )
t_FRAC. Rational Numbers	<i>n</i> / <i>m</i>
t_FFELT. Elt in finite field <b>F</b> <sub><i>q</i></sub>	ffgen( <i>q</i> , 't)
t_COMPLEX. Complex Numbers	<i>x</i> + <i>y</i> * I
t_PADIC. <i>p</i> -adic Numbers	<i>x</i> + 0( <i>p</i> ^ <i>k</i> )
t_QUAD. Quadratic Numbers	<i>x</i> + <i>y</i> * quadgen( <i>D</i> ,{'w'})
t_POLMOD. Polynomials modulo <i>g</i>	Mod( <i>f</i> , <i>g</i> )
t_POL. Polynomials	<i>a</i> * <i>x</i> ^ <i>n</i> + ... + <i>b</i>
t_SER. Power Series	<i>f</i> + 0( <i>x</i> ^ <i>k</i> )
t_RFRAC. Rational Functions	<i>f</i> / <i>g</i>
t_QFB. Binary quadratic form	Qfb( <i>a</i> , <i>b</i> , <i>c</i> )
t_VEC/t_COL. Row/Column Vectors	[ <i>x</i> , <i>y</i> , <i>z</i> ], [ <i>x</i> , <i>y</i> , <i>z</i> ]~
t_VEC integer range	[1..10]

t_VECSMALL. Vector of small ints	Vecsmall([ <i>x</i> , <i>y</i> , <i>z</i> ])
t_MAT. Matrices	[ <i>a</i> , <i>b</i> ; <i>c</i> , <i>d</i> ]
t_LIST. Lists	List([ <i>x</i> , <i>y</i> , <i>z</i> ])
t_STR. Strings	"abc"
t_INFINITY. ±∞	+oo, -oo

## Reserved Variable Names

$\pi \approx 3.14$ , $\gamma \approx 0.57$ , $C \approx 0.91$ , $I = \sqrt{-1}$	Pi, Euler, Catalan, I
Landau's big-oh notation	O

## Information about an Object, Precision

PARI type of object <i>x</i>	type( <i>x</i> )
length of <i>x</i> / size of <i>x</i> in memory	# <i>x</i> , sizebyte( <i>x</i> )
real precision / bit precision of <i>x</i>	precision( <i>x</i> ), bitprecision( <i>x</i> )
<i>p</i> -adic, series prec. of <i>x</i>	padicprec( <i>x</i> , <i>p</i> ), serprec( <i>x</i> , <i>v</i> )
current dynamic precision	getlocalprec, getlocalbitprec

## Operators

basic operations	+, − , *, /, ^, sqr
i←i+1, i←i-1, i←i*j, ...	i++, i--, i*=j,...
Euclidean quotient, remainder	<i>x</i> \/ <i>y</i> , <i>x</i> % <i>y</i> , divrem( <i>x</i> , <i>y</i> )
shift <i>x</i> left or right <i>n</i> bits	<i>x</i> << <i>n</i> , <i>x</i> >> <i>n</i> or shift( <i>x</i> ,± <i>n</i> )
multiply by 2 <sup><i>n</i></sup>	shiftmul( <i>x</i> , <i>n</i> )
comparison operators	<=, <, >=, >, ==, !=, ==, lex, cmp
boolean operators (or, and, not)	, &&, !
bit operations	bitand, bitneg, bitor, bitxor, bitnegimply
maximum/minimum of <i>x</i> and <i>y</i>	max( <i>x</i> , <i>y</i> ), min( <i>x</i> , <i>y</i> )
sign of <i>x</i> (gives −1,0,1)	sign( <i>x</i> )
binary exponent of <i>x</i>	exponent( <i>x</i> )
derivative of <i>f</i> , 2nd derivative, etc.	<i>f</i> ' , <i>f</i> '' , ...
differential operator	diffop( <i>f</i> , <i>v</i> , <i>d</i> ,{ <i>n</i> =1})
quote operator (formal variable)	'x
assignment	x = <i>value</i>
simultaneous assignment <i>x</i> ← <i>v</i> [1], <i>y</i> ← <i>v</i> [2]	[x,y] = v

## Select Components

*Caveat*: components start at index *n* = 1.

<i>n</i> -th component of <i>x</i>	component( <i>x</i> , <i>n</i> )
<i>n</i> -th component of vector/list <i>x</i>	<i>x</i> [ <i>n</i> ]
components <i>a</i> , <i>a</i> + 1, ..., <i>b</i> of vector <i>x</i>	<i>x</i> [ <i>a</i> .. <i>b</i> ]
( <i>m</i> , <i>n</i> )-th component of matrix <i>x</i>	<i>x</i> [ <i>m</i> , <i>n</i> ]
row <i>m</i> or column <i>n</i> of matrix <i>x</i>	<i>x</i> [ <i>m</i> ,], <i>x</i> [, <i>n</i> ]
numerator/denominator of <i>x</i>	numerator( <i>x</i> ), denominator( <i>x</i> )

## Random Numbers

random integer/prime in [0, <i>N</i> [	random( <i>N</i> ), randomprime( <i>N</i> )
get/set random seed	getrand, setrand( <i>s</i> )

## Conversions

to vector, matrix, vec. of small ints	Col/Vec, Mat, Vecsmall
to list, set, map, string	List, Set, Map, Str
create ( <i>x</i> mod <i>y</i> )	Mod( <i>x</i> , <i>y</i> )
make <i>x</i> a polynomial of <i>v</i>	Pol( <i>x</i> ,{ <i>v</i> })
variants of Pol <i>et al.</i> , in reverse order	Polrev, Vecrev, Colrev
make <i>x</i> a power series of <i>v</i>	Ser( <i>x</i> ,{ <i>v</i> })
convert <i>x</i> to simplest possible type	simplify( <i>x</i> )
object <i>x</i> with real precision <i>n</i>	precision( <i>x</i> , <i>n</i> )
object <i>x</i> with bit precision <i>n</i>	bitprecision( <i>x</i> , <i>n</i> )
set precision to <i>p</i> digits in dynamic scope	localprec( <i>p</i> )
set precision to <i>p</i> bits in dynamic scope	localbitprec( <i>p</i> )

## Character strings

convert to TeX representation	strtex( <i>x</i> )
string from bytes / from format+args	strchr, sprintf
split string / join strings	strsplit, strjoin
convert time <i>t</i> ms. to h, m, s, ms format	strtime( <i>t</i> )
<b>Conjugates and Lifts</b>	
conjugate of a number <i>x</i>	conj( <i>x</i> )
norm of <i>x</i> , product with conjugate	norm( <i>x</i> )
<i>L</i> <sup><i>p</i></sup> norm of <i>x</i> ( <i>L</i> <sup>∞</sup> if no <i>p</i> )	normlp( <i>x</i> ,{ <i>p</i> })
square of <i>L</i> <sup>2</sup> norm of <i>x</i>	norml2( <i>x</i> )
lift of <i>x</i> from Mods and <i>p</i> -adics	lift, centerlift( <i>x</i> )
recursive lift	liftall
lift all t_INT and t_PADIC (→t_INT)	liftint
lift all t_POLMOD (→t_POL)	liftpol

## Lists, Sets & Maps

<b>Sets</b> (= row vector with strictly increasing entries w.r.t. cmp)	
intersection of sets <i>x</i> and <i>y</i>	setintersect( <i>x</i> , <i>y</i> )
set of elements in <i>x</i> not belonging to <i>y</i>	setminus( <i>x</i> , <i>y</i> )
symmetric difference <i>x</i> Δ <i>y</i>	setdelta( <i>x</i> , <i>y</i> )
union of sets <i>x</i> and <i>y</i>	setunion( <i>x</i> , <i>y</i> )
does <i>y</i> belong to the set <i>x</i>	setsearch( <i>x</i> , <i>y</i> ,{ <i>flag</i> })
set of all <i>f</i> ( <i>x</i> , <i>y</i> ), <i>x</i> ∈ <i>X</i> , <i>y</i> ∈ <i>Y</i>	setbinop( <i>f</i> , <i>X</i> , <i>Y</i> )
is <i>x</i> a set ?	setisset( <i>x</i> )

**Lists.** create empty list: *L* = List()

append <i>x</i> to list <i>L</i>	listput( <i>L</i> , <i>x</i> ,{ <i>i</i> })
remove <i>i</i> -th component from list <i>L</i>	listpop( <i>L</i> ,{ <i>i</i> })
insert <i>x</i> in list <i>L</i> at position <i>i</i>	listinsert( <i>L</i> , <i>x</i> , <i>i</i> )
sort the list <i>L</i> in place	listsort( <i>L</i> ,{ <i>flag</i> })

**Maps.** create empty dictionary: *M* = Map()

attach value <i>v</i> to key <i>k</i>	mapput( <i>M</i> , <i>k</i> , <i>v</i> )
recover value attach to key <i>k</i> or error	mapget( <i>M</i> , <i>k</i> )
is key <i>k</i> in the dict? (set <i>v</i> to <i>M</i> ( <i>k</i> ))	mapisdefined( <i>M</i> , <i>k</i> ,{& <i>v</i> })
evaluate <i>f</i> at <i>M</i> ( <i>k</i> )	mapapply( <i>M</i> , <i>k</i> , <i>f</i> )
remove <i>k</i> from map domain	mapdelete( <i>M</i> , <i>k</i> )

## GP Programming

### User functions and closures

*x*, *y* are formal parameters; *y* defaults to Pi if parameter omitted;  
*z*, *t* are local variables (lexical scope), *z* initialized to 1.

fun(x, y=Pi) = my(z=1, t); seq  
fun = (x, y=Pi) -> my(z=1, t); seq

attach help message <i>h</i> to <i>s</i>	addhelp( <i>s</i> , <i>h</i> )
undefine symbol <i>s</i> (also kills help)	kill( <i>s</i> )
<b>Control Statements</b> ( <i>X</i> : formal parameter in expression <i>seq</i> )	
if <i>a</i> ≠ 0, evaluate <i>seq</i> <sub>1</sub> , else <i>seq</i> <sub>2</sub>	if( <i>a</i> ,{ <i>seq</i> <sub>1</sub> },{ <i>seq</i> <sub>2</sub> })
eval. <i>seq</i> for <i>a</i> ≤ <i>X</i> ≤ <i>b</i>	for( <i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i> )
...for <i>X</i> ∈ <i>v</i>	foreach( <i>v</i> , <i>X</i> , <i>seq</i> )
...for primes <i>a</i> ≤ <i>X</i> ≤ <i>b</i>	forprime( <i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i> )
...for primes ≡ <i>a</i> (mod <i>q</i> )	forprimestep( <i>X</i> = <i>a</i> , <i>b</i> , <i>q</i> , <i>seq</i> )
...for composites <i>a</i> ≤ <i>X</i> ≤ <i>b</i>	forcomposite( <i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i> )
...for <i>a</i> ≤ <i>X</i> ≤ <i>b</i> stepping <i>s</i>	forstep( <i>X</i> = <i>a</i> , <i>b</i> , <i>s</i> , <i>seq</i> )
...for <i>X</i> dividing <i>n</i>	fordiv( <i>n</i> , <i>X</i> , <i>seq</i> )
... <i>X</i> = [ <i>n</i> , factor( <i>n</i> )], <i>a</i> ≤ <i>n</i> ≤ <i>b</i>	forfactored( <i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i> )
...as above, <i>n</i> squarefree	forsquarefree( <i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i> )
... <i>X</i> = [ <i>d</i> , factor( <i>d</i> )], <i>d</i>   <i>n</i>	fordivfactored( <i>n</i> , <i>X</i> , <i>seq</i> )
multivariable for, lex ordering	forvec( <i>X</i> = <i>v</i> , <i>seq</i> )

loop over partitions of  $n$   
... permutations of  $S$   
... subsets of  $\{1, \dots, n\}$   
...  $k$ -subsets of  $\{1, \dots, n\}$   
... vectors  $v, q(v) \leq B; q > 0$   
...  $H < G$  finite abelian group  
evaluate  $seq$  until  $a \neq 0$   
while  $a \neq 0$ , evaluate  $seq$   
exit  $n$  innermost enclosing loops  
start new iteration of  $n$ -th enclosing loop  
return  $x$  from current subroutine

**Exceptions, warnings**  
raise an exception / warning  
type of error message  $E$   
try  $seq_1$ , evaluate  $seq_2$  on error

**Functions with closure arguments / results**  
number of arguments of  $f$   
select from  $v$  according to  $f$   
apply  $f$  to all entries in  $v$   
evaluate  $f(a_1, \dots, a_n)$   
evaluate  $f(\dots f(f(a_1, a_2), a_3) \dots, a_n)$   
calling function as closure

**Sums & Products**  
sum  $X = a$  to  $X = b$ , initialized at  $x$   
sum entries of vector  $v$   
product of all vector entries  
sum  $expr$  over divisors of  $n$   
... assuming  $expr$  multiplicative  
product  $a \leq X \leq b$ , initialized at  $x$   
product over primes  $a \leq X \leq b$

**Sorting**  
sort  $x$  by  $k$ -th component  
min.  $m$  of  $x$  ( $m = x[i]$ ), max.  
does  $y$  belong to  $x$ , sorted wrt.  $f$   
 $\prod g^x \rightarrow$  factorization ( $\Rightarrow$  sorted, unique  $g$ )

**Input/Output**  
print with/without  $\backslash n$ , TeX format  
pretty print matrix  
print fields with separator  
formatted printing  
write  $args$  to file  
write  $x$  in binary format  
read file into GP  
... return as vector of lines  
... return as vector of strings  
read a string from keyboard

**Files and file descriptors**  
File descriptors allow efficient small consecutive reads or writes from or to a given file. The argument  $n$  below is always a descriptor, attached to a file in **r**(ead), **w**(rite) or **a**(ppend) mode.  
get descriptor  $n$  for file  $path$  in given  $mode$   
... from shell  $cmd$  output (pipe)

close descriptor  
commit pending write operations  
read logical line from file  
... raw line from file  
write  $s \backslash n$  to file  
... write  $s$  to file

forpart( $p = n, seq$ )  
forperm( $S, p, seq$ )  
forsubset( $n, p, seq$ )  
forsubset( $[n, k], p, seq$ )  
forqfvec( $v, q, b, seq$ )  
forsubgroup( $H = G$ )  
until( $a, seq$ )  
while( $a, seq$ )  
break( $\{n\}$ )  
next( $\{n\}$ )  
return( $\{x\}$ )

error(), warning()  
errname( $E$ )  
iferr( $seq_1, E, seq_2$ )

arity( $f$ )  
select( $f, v$ )  
apply( $f, v$ )  
call( $f, a$ )  
fold( $f, a$ )  
self()

sum( $X = a, b, expr, \{x\}$ )  
vecsum( $v$ )  
vecprod( $v$ )  
sumdiv( $n, X, expr$ )  
sumdivmult( $n, X, expr$ )  
prod( $X = a, b, expr, \{x\}$ )  
prodeuler( $X = a, b, expr$ )

vecsort( $x, \{k\}, \{fl = 0\}$ )  
vecmin( $x, \{\&i\}$ ), vecmax  
vecsearch( $x, y, \{f\}$ )  
matreduce( $m$ )

print, print1, printtex  
printp  
printsep( $sep, \dots$ ), printsep1  
printf()

write, write1, writetex( $file, args$ )  
writebin( $file, x$ )  
read( $\{file\}$ )  
readvec( $\{file\}$ )  
readstr( $\{file\}$ )  
input()

# Pari-GP reference card

(PARI-GP version 2.17.3)

**Timers**  
CPU time in  $ms$  and reset timer  
CPU time in  $ms$  since gp startup  
time in  $ms$  since UNIX Epoch  
timeout command after  $s$  seconds

**Interface with system**  
allocates a new stack of  $s$  bytes  
alias  $old$  to  $new$   
install function from library  
execute system command  $a$   
... and feed result to GP  
... returning GP string  
get \$VAR from environment  
expand env. variable in string

**Parallel evaluation**  
These functions evaluate their arguments in parallel (pthreads or MPI); args. must not access global variables (use export for this) and must be free of side effects. Enabled if threading engine is not *single* in gp header.  
evaluate  $f$  on  $x[1], \dots, x[n]$   
evaluate closures  $f[1], \dots, f[n]$   
as select  
as sum  
as vector  
eval  $f$  for  $i = a, \dots, b$   
... for each element  $x$  in  $v$   
... for  $p$  prime in  $[a, b]$   
... for  $p = a \bmod q$   
... for  $i = a, a + s, \dots, b$   
... multivariate  
export  $x$  to parallel world  
... all dynamic variables  
frees exported value  $x$   
... all exported values

gettime()  
getabstime()  
getwalltime()  
alarm( $s, expr$ )

allocatemem( $\{s\}$ )  
alias( $new, old$ )  
install( $f, code, \{gpf\}, \{lib\}$ )  
system( $a$ )  
extern( $a$ )  
externstr( $a$ )  
getenv("VAR")  
strexpend( $x$ )

parapply( $f, x$ )  
pareval( $f$ )  
parselect( $f, A, \{flag\}$ )  
parsum( $i = a, b, expr$ )  
parvector( $n, i, \{expr\}$ )  
parfor( $i = a, \{b\}, f, \{r\}, \{f_2\}$ )  
parforeach( $v, x, f, \{r\}, \{f_2\}$ )  
parforprime( $p = a, \{b\}, f, \{r\}, \{f_2\}$ )  
parforprimestep( $p = a, \{b\}, q, f, \{r\}, \{f_2\}$ )  
parforstep( $i = a, \{b\}, s, f, \{r\}, \{f_2\}$ )  
parforvec( $X = v, f, \{r\}, \{f_2\}, \{flag\}$ )

export( $x$ )  
exportall()  
unexport( $x$ )  
unexportall()

**Linear Algebra**  
dimensions of matrix  $x$   
multiply two matrices  
... assuming result is diagonal  
concatenation of  $x$  and  $y$   
extract components of  $x$   
transpose of vector or matrix  $x$   
adjoint of the matrix  $x$   
eigenvectors/values of matrix  $x$   
characteristic/minimal polynomial of  $x$   
trace/determinant of matrix  $x$   
permanent of matrix  $x$   
Frobenius form of  $x$   
QR decomposition  
apply matqr's transform to  $v$

**Constructors & Special Matrices**  
 $\{g(x): x \in v \text{ s.t. } f(x)\}$   
 $\{x: x \in v \text{ s.t. } f(x)\}$   
 $\{g(x): x \in v\}$   
row vec. of  $expr$  eval'd at  $1 \leq i \leq n$   
col. vec. of  $expr$  eval'd at  $1 \leq i \leq n$   
vector of small ints

matsize( $x$ )  
 $x * y$   
matmultodiagonal( $x, y$ )  
concat( $x, \{y\}$ )  
vecextract( $x, y, \{z\}$ )  
 $x \sim$ , mattranspose( $x$ )  
matadjoin( $x$ )  
mateigen( $x$ )  
charpoly( $x$ ), minpoly( $x$ )  
trace( $x$ ), matdet( $x$ )  
matpermanent( $x$ )  
matfrobenius( $x$ )  
matqr( $x$ )  
mathouseholder( $Q, v$ )

[g(x) | x <- v, f(x)]  
[x | x <- v, f(x)]  
[g(x) | x <- v]  
vector( $n, \{i\}, \{expr\}$ )  
vectorv( $n, \{i\}, \{expr\}$ )  
vectorsmall( $n, \{i\}, \{expr\}$ )

$[c, c \cdot x, \dots, c \cdot x^n]$   
 $[1, 2^x, \dots, n^x]$   
matrix  $1 \leq i \leq m, 1 \leq j \leq n$   
define matrix by blocks  
diagonal matrix with diagonal  $x$   
is  $x$  diagonal?  
 $x \cdot \text{matdiagonal}(d)$   
 $n \times n$  identity matrix  
Hessenberg form of square matrix  $x$   
 $n \times n$  Hilbert matrix  $H_{ij} = (i + j - 1)^{-1}$   
 $n \times n$  Pascal triangle  
companion matrix to polynomial  $x$   
Sylvester matrix of  $x$  and  $y$

**Gaussian elimination**  
kernel of matrix  $x$   
intersection of column spaces of  $x$  and  $y$   
solve  $MX = B$  ( $M$  invertible)  
one sol of  $M * X = B$   
basis for image of matrix  $x$   
columns of  $x$  *not* in **matimage**  
supplement columns of  $x$  to get basis  
rows, cols to extract invertible matrix  
rank of the matrix  $x$   
solve  $MX = B \bmod D$   
image mod  $D$   
kernel mod  $D$   
inverse mod  $D$   
determinant mod  $D$

**Lattices & Quadratic Forms**  
**Quadratic forms**  
evaluate  ${}^t x Q y$   
evaluate  ${}^t x Q x$   
signature of quad form  ${}^t y * x * y$   
decomp into squares of  ${}^t y * x * y$   
eigenvalues/vectors for real symmetric  $x$   
Cholesky decomposition of  $x$

**HNF and SNF**  
upper triangular Hermite Normal Form  
HNF of  $x$  where  $d$  is a multiple of  $\det(x)$   
multiple of  $\det(x)$   
HNF of  $(x \mid \text{diagonal}(D))$   
elementary divisors of  $x$   
 $q$ -rank from elementary divisors  
elementary divisors of  $\mathbf{Z}[a]/(f'(a))$   
integer kernel of  $x$   
**Z**-module  $\leftrightarrow$  **Q**-vector space

**Lattices**  
LLL-algorithm applied to columns of  $x$   
... for Gram matrix of lattice  
find up to  $m$  sols of  $\text{qfeval}(x, y) \leq b$   
... up to  $m$  closest vectors to  $t$   
 $v, v[i] :=$  number of  $y$  s.t.  $\text{qfeval}(x, y) = i$   
perfection rank of  $x$   
find isomorphism between  $q$  and  $Q$

powers( $x, n, \{c = 1\}$ )  
dirpowers( $n, x$ )  
matrix( $m, n, \{i\}, \{j\}, \{expr\}$ )  
matconcat( $B$ )  
matdiagonal( $x$ )  
matisdiagonal( $x$ )  
matmuldiagonal( $x, d$ )  
matid( $n$ )  
mathess( $x$ )  
mathilbert( $n$ )  
matpascal( $n - 1$ )  
matcompanion( $x$ )  
polsylvestermatrix( $x, y$ )

matker( $x, \{flag\}$ )  
matintersect( $x, y$ )  
matsolve( $M, B$ )  
matinverseimage( $M, B$ )  
matimage( $x$ )  
matimagecompl( $x$ )  
matsupplement( $x$ )  
matindexrank( $x$ )  
matrank( $x$ )  
matsolvemod( $M, D, B$ )  
matimagemod( $M, D$ )  
matkermod( $M, D$ )  
matinvmod( $M, D$ )  
matdetmod( $M, D$ )

qfeval( $\{Q = id\}, x, y$ )  
qfeval( $\{Q = id\}, x$ )  
qfsign( $x$ )  
qfgaussred( $x$ )  
qfjacobi( $x$ )  
qfcholesky( $x$ )

mathnf( $x$ )  
mathnfmod( $x, d$ )  
matdetint( $x$ )  
mathnfmodid( $x, D$ )  
matsnf( $x$ )  
snfrank( $v, q$ )  
poldiscreduced( $f$ )  
matkerint( $x$ )  
matrixqz( $x, p$ )

qflll( $x, \{flag\}$ )  
qflllgram( $x, \{flag\}$ )  
qfminim( $x, b, m$ )  
qfcvp( $x, t, m$ )  
qfrep( $x, B, \{flag\}$ )  
qfperfection( $x$ )  
qfisom( $q, Q$ )

precompute for isomorphism test with $q$	<code>qfisominit(<math>q</math>)</code>
automorphism group of $q$	<code>qfauto(<math>q</math>)</code>
convert <code>qfauto</code> for GAP/Magma	<code>qfautoexport(<math>G, \{flag\}</math>)</code>
orbits of $V$ under $G \subset \mathrm{GL}(V)$	<code>qforbits(<math>G, V</math>)</code>
<b>Polynomials &amp; Rational Functions</b>	
all defined polynomial variables	<code>variables()</code>
get var. of highest priority (higher than $v$ )	<code>varhigher(<math>name, \{v\}</math>)</code>
... of lowest priority (lower than $v$ )	<code>varlower(<math>name, \{v\}</math>)</code>
<b>Coefficients, variables and basic operators</b>	
degree of $f$	<code>poldegree(<math>f</math>)</code>
coef. of degree $n$ of $f$ , leading coef.	<code>polcoef(<math>f, n</math>), pollead</code>
main variable / all variables in $f$	<code>variable(<math>f</math>), variables(<math>f</math>)</code>
replace $x$ by $y$ in $f$	<code>subst(<math>f, x, y</math>)</code>
evaluate $f$ replacing vars by their value	<code>eval(<math>f</math>)</code>
replace polynomial expr. $T(x)$ by $y$ in $f$	<code>substpol(<math>f, T, y</math>)</code>
replace $x_1, \dots, x_n$ by $y_1, \dots, y_n$ in $f$	<code>substvec(<math>f, x, y</math>)</code>
$f \in A[x]$ ; reciprocal polynomial $x^{\deg f} f\left(\frac{1}{x}\right)$	<code>polrecip(<math>f</math>)</code>
gcd of coefficients of $f$	<code>content(<math>f</math>)</code>
derivative of $f$ w.r.t. $x$	<code>deriv(<math>f, \{x\}</math>)</code>
... $n$ -th derivative of $f$	<code>derivn(<math>f, n, \{x\}</math>)</code>
formal integral of $f$ w.r.t. $x$	<code>intformal(<math>f, \{x\}</math>)</code>
formal sum of $f$ w.r.t. $x$	<code>sumformal(<math>f, \{x\}</math>)</code>
<b>Constructors &amp; Special Polynomials</b>	
interpolation polynomial at $(x[1], y[1]), \dots, (x[n], y[n])$ , evaluated at $t$ , with error estimate $e$	<code>polinterpolate(<math>x, \{y\}, \{t\}, \{&amp;e\}</math>)</code>
monic polynomial from roots $r$	<code>polfromroots(<math>r</math>)</code>
$T_n/U_n, H_n$	<code>polchebyshev(<math>n</math>), polhermite(<math>n</math>)</code>
$P_n, L_n^{(\alpha)}$	<code>pollegendre(<math>n</math>), pollaguerre(<math>n, a</math>)</code>
$n$ -th cyclotomic polynomial $\Phi_n$	<code>polcyclo(<math>n</math>)</code>
return $n$ if $f = \Phi_n$ , else 0	<code>poliscyclo(<math>f</math>)</code>
is $f$ a product of cyclotomic polynomials?	<code>poliscycloprod(<math>f</math>)</code>
Zagier's polynomial of index $(n, m)$	<code>polzagier(<math>n, m</math>)</code>
<b>Resultant, elimination</b>	
discriminant of polynomial $f$	<code>poldisc(<math>f</math>)</code>
find factors of <code>poldisc(<math>f</math>)</code>	<code>poldiscfactors(<math>f</math>)</code>
resultant $R = \mathrm{Res}_v(f, g)$	<code>polresultant(<math>f, g, \{v\}</math>)</code>
$[u, v, R], xu + yv = \mathrm{Res}_v(f, g)$	<code>polresultantext(<math>x, y, \{v\}</math>)</code>
solve Thue equation $f(x, y) = a$	<code>thue(<math>t, a, \{sol\}</math>)</code>
initialize $t$ for Thue equation solver	<code>thueinit(<math>f</math>)</code>
<b>Roots and Factorization (Complex/Real)</b>	
complex roots of $f$	<code>polroots(<math>f</math>)</code>
bound complex roots of $f$	<code>polrootsbound(<math>f</math>)</code>
number of real roots of $f$ (in $[a, b]$ )	<code>polsturm(<math>f, \{[a, b]\}</math>)</code>
real roots of $f$ (in $[a, b]$ )	<code>polrootsreal(<math>f, \{[a, b]\}</math>)</code>
complex embeddings of <code>t.POLMOD <math>z</math></code>	<code>conjvec(<math>z</math>)</code>
<b>Roots and Factorization (Finite fields)</b>	
factor $f$ mod $p$ , roots	<code>factormod(<math>f, p</math>), polrootsmod</code>
factor $f$ over $\mathbf{F}_p[x]/(T)$ , roots	<code>factormod(<math>f, [T, p]</math>), polrootsmod</code>
squarefree factorization of $f$ in $\mathbf{F}_q[x]$	<code>factormodSQF(<math>f, \{D\}</math>)</code>
distinct degree factorization of $f$ in $\mathbf{F}_q[x]$	<code>factormodDDF(<math>f, \{D\}</math>)</code>
factor $n$ -th cyclotomic pol. $\Phi_n$ mod $p$	<code>factormodcyclo(<math>n, p</math>)</code>
<b>Roots and Factorization (<math>p</math>-adic fields)</b>	
factor $f$ over $\mathbf{Q}_p$ , roots	<code>factorpadic(<math>f, p, r</math>), polrootspadic</code>
$p$ -adic root of $f$ congruent to $a$ mod $p$	<code>padicappr(<math>f, a</math>)</code>
Newton polygon of $f$ for prime $p$	<code>newtonpoly(<math>f, p</math>)</code>
Hensel lift $A/\mathrm{lcl}(A) = \prod_i B[i] \bmod p^e$	<code>polhensellift(<math>A, B, p, e</math>)</code>

# Pari-GP reference card

(PARI-GP version 2.17.3)

$T = \prod (x - z_i) \mapsto \prod (x - \omega(z_i)) \in \mathbf{Z}_p[x]$	<code>polteichmuller(<math>T, p, e</math>)</code>
extensions of $\mathbf{Q}_p$ of degree $N$	<code>padicfields(<math>p, N</math>)</code>
<b>Roots and Factorization (Miscellaneous)</b>	
symmetric powers of roots of $f$ up to $n$	<code>polsym(<math>f, n</math>)</code>
Graeffe transform of $f, g(x^2) = f(x)f(-x)$	<code>polgraeffe(<math>f</math>)</code>
factor $f$ over coefficient field	<code>factor(<math>f</math>)</code>
cyclotomic factors of $f \in \mathbf{Q}[X]$	<code>polcyclofactors(<math>f</math>)</code>
<b>Finite Fields</b>	
A finite field is encoded by any element ( <code>t_FFELT</code> ).	
find irreducible $T \in \mathbf{F}_p[x]$ , $\deg T = n$	<code>ffinit(<math>p, n, \{x\}</math>)</code>
Create $t$ in $\mathbf{F}_q \simeq \mathbf{F}_p[t]/(T)$	<code>t = ffgen(<math>T, 't</math>)</code>
... indirectly, with implicit $T$	<code>t = ffgen(<math>q, 't</math>); T = t.mod</code>
map $m$ from $\mathbf{F}_q \ni a$ to $\mathbf{F}_{q^k} \ni b$	<code>m = ffebed(<math>a, b</math>)</code>
build $K = \mathbf{F}_q[x]/(P)$ extending $\mathbf{F}_q \ni a$ ,	<code>ffextend(<math>a, P</math>)</code>
evaluate map $m$ on $x$	<code>ffmap(<math>m, x</math>)</code>
inverse map of $m$	<code>ffinvmap(<math>m</math>)</code>
compose maps $m \circ n$	<code>ffcompomap(<math>m, n</math>)</code>
$x$ as polmod over codomain of map $m$	<code>ffmaprel(<math>m, x</math>)</code>
$F^n$ over $\mathbf{F}_q \ni a$	<code>fffrobenius(<math>a, n</math>)</code>
$\#\{\text{monic irred. } T \in \mathbf{F}_q[x], \deg T = n\}$	<code>ffnbirred(<math>q, n</math>)</code>
<b>Formal &amp; p-adic Series</b>	
truncate power series or $p$ -adic number	<code>truncate(<math>x</math>)</code>
valuation of $x$ at $p$	<code>valuation(<math>x, p</math>)</code>
<b>Dirichlet and Power Series</b>	
Taylor expansion around 0 of $f$ w.r.t. $x$	<code>taylor(<math>f, x</math>)</code>
Laurent series of closure $F$ up to $x^k$	<code>laurentseries(<math>f, k</math>)</code>
$\sum a_k b_k t^k$ from $\sum a_k t^k$ and $\sum b_k t^k$	<code>serconvol(<math>a, b</math>)</code>
$f = \sum a_k t^k$ from $\sum (a_k/k!) t^k$	<code>serlaplace(<math>f</math>)</code>
reverse power series $F$ so $F(f(x)) = x$	<code>serreverse(<math>f</math>)</code>
remove terms of degree $< n$ in $f$	<code>serchop(<math>f, n</math>)</code>
Dirichlet series multiplication / division	<code>dirmul, dirdiv(<math>x, y</math>)</code>
Dirichlet Euler product ( $b$ terms)	<code>direuler(<math>p = a, b, expr</math>)</code>
<b>Transcendental and <math>p</math>-adic Functions</b>	
real, imaginary part of $x$	<code>real(<math>x</math>), imag(<math>x</math>)</code>
absolute value, argument of $x$	<code>abs(<math>x</math>), arg(<math>x</math>)</code>
square/ $n$ th root of $x$	<code>sqrt(<math>x</math>), sqrtsn(<math>x, n, \{&amp;z\}</math>)</code>
all $n$ -th roots of 1	<code>rootsof1(<math>n</math>)</code>
FFT of $[f_0, \dots, f_{n-1}]$	<code>w = fftinit(<math>n</math>), fft/fftinw(<math>w, f</math>)</code>
trig functions	<code>sin, cos, tan, cotan, sinc</code>
inverse trig functions	<code>asin, acos, atan</code>
hyperbolic functions	<code>sinh, cosh, tanh, cotanh</code>
inverse hyperbolic functions	<code>asinh, acosh, atanh</code>
$\log(x)$ , $\log(1+x)$ , $e^x$ , $e^x - 1$	<code>log, loglp, exp, expm1</code>
Euler $\Gamma$ function, $\log \Gamma$ , $\Gamma'/\Gamma$	<code>gamma, lngamma, psi</code>
half-integer gamma function $\Gamma(n+1/2)$	<code>gammah(<math>n</math>)</code>
Riemann's zeta $\zeta(s) = \sum n^{-s}$	<code>zeta(<math>s</math>)</code>
$\sum_{1 \leq n \leq N} n^s$	<code>dirpowerssum(<math>N, s</math>)</code>
Hurwitz's $\zeta(s, x) = \sum (n+x)^{-s}$	<code>zetahurwitz(<math>s, x</math>)</code>
Lerch $\Phi(z, s, x) = \sum z^n (n+x)^{-s}$	<code>lerchphi(<math>z, s, x</math>)</code>
Lerch $L(s, x, t) = \Phi(e^{2i\pi t}, s, x)$	<code>lerchzeta(<math>s, x, t</math>)</code>
multiple zeta value (MZV), $\zeta(s_1, \dots, s_k)$	<code>zetamult(<math>s, \{T\}</math>)</code>
all MZVs for weight $\sum s_i = n$	<code>zetamultall(<math>n</math>)</code>
convert MZV id to $[s_1, \dots, s_k]$	<code>zetamultconvert(<math>f, \{flag\}</math>)</code>
MZV dual sequence	<code>zetamultdual(<math>s</math>)</code>
multiple polylog $Li_{s_1, \dots, s_k}(z_1, \dots, z_k)$	<code>polylogmult(<math>s, z</math>)</code>

incomplete $\Gamma$ function ( $y = \Gamma(s)$ )	<code>incgam(<math>s, x, \{y\}</math>)</code>
complementary incomplete $\Gamma$	<code>incgamc(<math>s, x</math>)</code>
$\int_x^\infty e^{-t} dt/t$ , $(2/\sqrt{\pi}) \int_x^\infty e^{-t^2} dt$	<code>eint1, erfc</code>
elliptic integral of 1st and 2nd kind	<code>ellK(<math>k</math>), ellE(<math>k</math>)</code>
dilogarithm of $x$	<code>dilog(<math>x</math>)</code>
$m$ -th polylogarithm of $x$	<code>polylog(<math>m, x, \{flag\}</math>)</code>
$U$ -confluent hypergeometric function	<code>hyperu(<math>a, b, u</math>)</code>
Hypergeometric ${}_pF_q(A, B; z)$	<code>hypergeom(<math>A, B, z</math>)</code>
Bessel $J_n(x)$ , $J_{n+1/2}(x)$	<code>besselj(<math>n, x</math>), besseljh(<math>n, x</math>)</code>
Bessel $I_\nu, K_\nu, H_\nu^1, H_\nu^2, Y_\nu$	<code>(bessel)i, k, h1, h2, y</code>
$k$ -th zero of $J_\nu(x)$	<code>besseljzero(<math>nu, \{k = 1\}</math>)</code>
$k$ -th zero of $Y_\nu(x)$	<code>besselyzero(<math>nu, \{k = 1\}</math>)</code>
Airy functions $A_i(x)$ , $B_i(x)$	<code>airy(<math>x</math>)</code>
Lambert $W$ : $x$ s.t. $xe^x = y$	<code>lambertw(<math>y</math>)</code>
Teichmuller character of $p$ -adic $x$	<code>teichmuller(<math>x</math>)</code>

## Iterations, Sums & Products

<b>Numerical integration for meromorphic functions</b>	
Behaviour at endpoint for Double Exponential (DE) methods: either a scalar ( $a \in \mathbf{C}$ , regular) or $\pm\infty$ (decreasing at least as $x^{-2}$ ) or	
$(x-a)^{-\alpha}$ singularity	<code>[<math>a, \alpha</math>]</code>
exponential decrease $e^{-\alpha x }$	<code>[<math>\pm\infty, \alpha</math>], <math>\alpha &gt; 0</math></code>
slow decrease $ x ^\alpha$	<code>... <math>\alpha &lt; -1</math></code>
oscillating as $\cos(kx)$	<code><math>\alpha = k\mathbf{I}</math>, <math>k &gt; 0</math></code>
oscillating as $\sin(kx)$	<code><math>\alpha = -k\mathbf{I}</math>, <math>k &gt; 0</math></code>
numerical integration	<code>intnum(<math>x = a, b, f, \{T\}</math>)</code>
weights $T$ for intnum	<code>intnuminit(<math>a, b, \{m\}</math>)</code>
weights $T$ incl. kernel $K$	<code>intfuncinit(<math>t = a, b, K, \{m\}</math>)</code>
integrate $(2i\pi)^{-1} f$ on circle $ z-a  = R$	<code>intcirc(<math>x = a, R, f, \{T\}</math>)</code>
<b>Other integration methods</b>	
$n$ -point Gauss-Legendre	<code>intnumgauss(<math>x = a, b, f, \{n\}</math>)</code>
weights for $n$ -point Gauss-Legendre	<code>intnumgaussinit(<math>\{n\}</math>)</code>
quasi-periodic function, period $2H$	<code>intnumosc(<math>x = a, f, H</math>)</code>
Romberg (low accuracy)	<code>intnumromb(<math>x = a, b, f, \{flag\}</math>)</code>
<b>Numerical summation</b>	
sum of series $f(n)$ , $n \geq a$ (low accuracy)	<code>suminf(<math>n = a, expr</math>)</code>
sum of alternating/positive series	<code>sumalt, sumpos</code>
sum of series using Euler-Maclaurin	<code>sumnum(<math>n = a, f, \{T\}</math>)</code>
... Sidi summation	<code>sumnumsidi(<math>n = a, f</math>)</code>
$\sum_{n \geq a} F(n)$ , $F$ rational function	<code>sumnumrat(<math>F, a</math>)</code>
$\dots \sum_{p \geq a} F(p^s)$	<code>sumeulerrat(<math>F, \{s = 1\}, \{a = 2\}</math>)</code>
weights for sumnum, $a$ as in DE	<code>sumnuminit(<math>\{\infty, a\}</math>)</code>
sum of series by Monien summation	<code>sumnummonien(<math>n = a, f, \{T\}</math>)</code>
weights for sumnummonien	<code>sumnummonieninit(<math>\{\infty, a\}</math>)</code>
sum of series using Abel-Plana	<code>sumnumap(<math>n = a, f, \{T\}</math>)</code>
weights for sumnumap, $a$ as in DE	<code>sumnumapinit(<math>\{\infty, a\}</math>)</code>
sum of series using Lagrange	<code>sumnumlagrange(<math>n = a, f, \{T\}</math>)</code>
weights for sumnumlagrange	<code>sumnumlagrangeinit</code>
<b>Products</b>	
product $a \leq X \leq b$ , initialized at $x$	<code>prod(<math>X = a, b, expr, \{x\}</math>)</code>
product over primes $a \leq X \leq b$	<code>prodeuler(<math>X = a, b, expr</math>)</code>
infinite product $a \leq X \leq \infty$	<code>prodinf(<math>X = a, expr</math>)</code>
$\prod_{n \geq a} F(n)$ , $F$ rational function	<code>prodnunrat(<math>F, a</math>)</code>
$\prod_{p \geq a} F(p^s)$	<code>prodeulerrat(<math>F, \{s = 1\}, \{a = 2\}</math>)</code>

Other numerical methods

real root of $f$ in $[a, b]$ ; bracketed root	<code>solve(<math>X = a, b, f</math>)</code>
...interval splitting, step $s$	<code>solvestep(<math>X = a, b, s, f, \{flag = 0\}</math>)</code>
limit of $f(t)$ , $t \rightarrow \infty$	<code>limitnum(<math>f, \{\alpha\}</math>)</code>
asymptotic expansion of $f$ (rational)	<code>asypnum(<math>f, \{\alpha\}</math>)</code>
... $N + 1$ terms as floats	<code>asypnumraw(<math>f, N, \{\alpha\}</math>)</code>
numerical derivation w.r.t $x$ : $f'(a)$	<code>derivnum(<math>x = a, f</math>)</code>
evaluate continued fraction $F$ at $t$	<code>contfraceval(<math>F, t, \{L\}</math>)</code>
power series to cont. fraction ( $L$ terms)	<code>contfracinit(<math>S, \{L\}</math>)</code>
Padé approximant (deg. denom. $\leq B$ )	<code>bestapprPade(<math>S, \{B\}</math>)</code>

Elementary Arithmetic Functions

vector of binary digits of $ x $	<code>binary(<math>x</math>)</code>
bit number $n$ of integer $x$	<code>bittest(<math>x, n</math>)</code>
Hamming weight of integer $x$	<code>hammingweight(<math>x</math>)</code>
digits of integer $x$ in base $B$	<code>digits(<math>x, \{B = 10\}</math>)</code>
sum of digits of integer $x$ in base $B$	<code>sumdigits(<math>x, \{B = 10\}</math>)</code>
integer from digits	<code>fromdigits(<math>v, \{B = 10\}</math>)</code>
ceiling/floor/fractional part	<code>ceil, floor, frac</code>
round $x$ to nearest integer	<code>round(<math>x, \{\&amp;e\}</math>)</code>
truncate $x$	<code>truncate(<math>x, \{\&amp;e\}</math>)</code>
gcd/LCM of $x$ and $y$	<code>gcd(<math>x, y</math>), lcm(<math>x, y</math>)</code>
gcd of entries of a vector/matrix	<code>content(<math>x</math>)</code>

Primes and Factorization

extra prime table	<code>addprimes()</code>
add primes in $v$ to prime table	<code>addprimes(<math>v</math>)</code>
remove primes from prime table	<code>removeprimes(<math>v</math>)</code>
Chebyshev $\pi(x)$ , $n$ -th prime $p_n$	<code>primepi(<math>x</math>), prime(<math>n</math>)</code>
vector of first $n$ primes	<code>primes(<math>n</math>)</code>
smallest prime $\geq x$	<code>nextprime(<math>x</math>)</code>
largest prime $\leq x$	<code>precprime(<math>x</math>)</code>
factorization of $x$	<code>factor(<math>x, \{lim\}</math>)</code>
...selecting specific algorithms	<code>factorint(<math>x, \{flag = 0\}</math>)</code>
$n = df^2$ , $d$ squarefree/fundamental	<code>core(<math>n, \{fl\}</math>), coredisc</code>
certificate for (prime) $N$	<code>primecert(<math>N</math>)</code>
verifies a certificate $c$	<code>primecertisvalid(<math>c</math>)</code>
convert certificate to Magma/PRIMO	<code>primecertexport</code>
recover $x$ from its factorization	<code>factorback(<math>f, \{e\}</math>)</code>
$x \in \mathbf{Z}$ , $ x  \leq X$ , $\gcd(N, P(x)) \geq N$	<code>zncoppersmith(<math>P, N, X, \{B\}</math>)</code>
divisors of $N$ in residue class $r$ mod $s$	<code>divisorslensstra(<math>N, r, s</math>)</code>

Divisors and multiplicative functions

number of prime divisors $\omega(n)$ / $\Omega(n)$	<code>omega(<math>n</math>), bigomega</code>
divisors of $n$ / number of divisors $\tau(n)$	<code>divisors(<math>n</math>), numdiv</code>
sum of ( $k$ -th powers of) divisors of $n$	<code>sigma(<math>n, \{k\}</math>)</code>
Möbius $\mu$ -function	<code>moebius(<math>x</math>)</code>
Ramanujan's $\tau$ -function	<code>ramanujantau(<math>x</math>)</code>

Combinatorics

factorial of $x$	<code>x!</code> or <code>factorial(<math>x</math>)</code>
binomial coefficient $\binom{x}{k}$	<code>binomial(<math>x, \{k\}</math>)</code>
Bernoulli number $B_n$ as real/rational	<code>bernreal(<math>n</math>), bernfrac</code>
$[B_0, B_2, \dots B_{2k}]$	<code>bernvec(<math>k</math>)</code>
Bernoulli polynomial $B_n(x)$	<code>bernpol(<math>n, \{x\}</math>)</code>
Euler numbers	<code>eulerfrac, eulerreal, eulervec</code>
Euler polynomial $E_n(x)$	<code>eulerpol(<math>n, \{x\}</math>)</code>
Eulerian polynomial $A_n(x)$	<code>eulerianpol</code>
Fibonacci number $F_n$	<code>fibonacci(<math>n</math>)</code>
Harmonic number $H_{n,r} = 1^{-r} + \dots + n^{-r}$	<code>harmonic(<math>n, r</math>)</code>
Stirling numbers $s(n, k)$ and $S(n, k)$	<code>stirling(<math>n, k, \{flag\}</math>)</code>

Pari-GP reference card

(PARI-GP version 2.17.3)

number of partitions of $n$	<code>numbpart(<math>n</math>)</code>
$k$ -th permutation on $n$ letters	<code>numtoperm(<math>n, k</math>)</code>
...index $k$ of permutation $v$	<code>permtotnum(<math>v</math>)</code>
order of permutation $p$	<code>permorder(<math>p</math>)</code>
signature of permutation $p$	<code>permsign(<math>p</math>)</code>
cyclic decomposition of permutation $p$	<code>permcycles(<math>p</math>)</code>

Multiplicative groups  $(\mathbf{Z}/N\mathbf{Z})^*$ ,  $\mathbf{F}_q^*$

Euler $\phi$ -function	<code>eulerphi(<math>x</math>)</code>
multiplicative order of $x$ (divides $\phi$ )	<code>znorder(<math>x, \{o\}</math>), fforder</code>
primitive root mod $q$ / $x$ .mod	<code>znprimroot(<math>q</math>), fprimroot(<math>x</math>)</code>
structure of $(\mathbf{Z}/n\mathbf{Z})^*$	<code>znstar(<math>n</math>)</code>
discrete logarithm of $x$ in base $g$	<code>znlog(<math>x, g, \{o\}</math>), fflag</code>
Kronecker-Legendre symbol $(\frac{x}{y})$	<code>kronecker(<math>x, y</math>)</code>
quadratic Hilbert symbol (at $p$ )	<code>hilbert(<math>x, y, \{p\}</math>)</code>

Euclidean algorithm, continued fractions

CRT: solve $z \equiv x$ and $z \equiv y$	<code>chinese(<math>x, y</math>)</code>
minimal $u, v$ so $xu + yv = \gcd(x, y)$	<code>gcdext(<math>x, y</math>)</code>
half-gcd algorithm	<code>halfgcd(<math>x, y</math>)</code>
continued fraction of $x$	<code>contfrac(<math>x, \{b\}, \{lmax\}</math>)</code>
last convergent of continued fraction $x$	<code>contfracpnqn(<math>x</math>)</code>
rational approximation to $x$ (den. $\leq B$ )	<code>bestappr(<math>x, \{B\}</math>)</code>
recognize $x \in \mathbf{C}$ as polmod mod $T \in \mathbf{Z}[X]$	<code>bestapprnf(<math>x, T</math>)</code>

Miscellaneous

integer square / $n$ -th root of $x$	<code>sqrtnint(<math>x, n</math>)</code>
largest integer $e$ s.t. $b^e \leq x$ , $e = \lfloor \log_b(x) \rfloor$	<code>logint(<math>x, b, \{\&amp;z\}</math>)</code>

Characters

Let  $\chi = [d_1, \dots, d_k]$  represent an abelian group  $G = \oplus (\mathbf{Z}/d_j\mathbf{Z}) \cdot g_j$  or any structure  $G$  affording a `.cyc` method; e.g. `znstar( $q, 1$ )` for Dirichlet characters. A character  $\chi$  is coded by  $[c_1, \dots, c_k]$  such that  $\chi(g_j) = e(n_j/d_j)$ .  
 $\chi \cdot \psi$ ;  $\chi^{-1}$ ;  $\chi \cdot \psi^{-1}$ ;  $\chi^k$       `charmul, charconj, chardiv, charpow`  
order of  $\chi$       `charorder( $cyc, \chi$ )`  
kernel of  $\chi$       `charker( $cyc, \chi$ )`  
 $\chi(x)$ ,  $G$  a GP group structure      `chareval( $G, \chi, x, \{z\}$ )`  
Galois orbits of characters      `chargalois( $G$ )`

Dirichlet Characters

initialize  $G = (\mathbf{Z}/q\mathbf{Z})^*$       `G = znstar( $q, 1$ )`  
convert datum  $D$  to  $[G, \chi]$       `znchar( $D$ )`  
is  $\chi$  odd?      `zncharisodd( $G, \chi$ )`  
real  $\chi \rightarrow$  Kronecker symbol  $(D/\cdot)$       `znchartokronecker( $G, \chi$ )`  
conductor of  $\chi$       `zncharconductor( $G, \chi$ )`  
 $[G_0, \chi_0]$  primitive attached to  $\chi$       `znchartoprimitive( $G, \chi$ )`  
induce  $\chi \in \hat{G}$  to  $\mathbf{Z}/N\mathbf{Z}$       `zncharinduce( $G, \chi, N$ )`  
 $\chi p$       `znchardecompose( $G, \chi, p$ )`  
 $\prod_p |(Q, N) \chi p$       `znchardecompose( $G, \chi, Q$ )`  
complex Gauss sum  $G_a(\chi)$       `znchargauss( $G, \chi$ )`

Conrey labelling

Conrey label  $m \in (\mathbf{Z}/q\mathbf{Z})^* \rightarrow$  character      `znconreychar( $G, m$ )`  
character  $\rightarrow$  Conrey label      `znconreyexp( $G, \chi$ )`  
log on Conrey generators      `znconreylog( $G, m$ )`  
conductor of  $\chi$  ( $\chi_0$  primitive)      `znconreyconductor( $G, \chi, \{\chi_0\}$ )`

True-False Tests

is $x$ the disc. of a quadratic field?	<code>isfundamental(<math>x</math>)</code>
is $x$ a prime?	<code>isprime(<math>x</math>)</code>
is $x$ a strong pseudo-prime?	<code>ispseudoprime(<math>x</math>)</code>
is $x$ square-free?	<code>issquarefree(<math>x</math>)</code>
is $x$ a square?	<code>issquare(<math>x, \{\&amp;n\}</math>)</code>
is $x$ a perfect power?	<code>ispower(<math>x, \{k\}, \{\&amp;n\}</math>)</code>
is $x$ a perfect power of a prime? ( $x = p^n$ )	<code>isprimepower(<math>x, \&amp;n</math>)</code>
... of a pseudoprime?	<code>ispseudoprimepower(<math>x, \&amp;n</math>)</code>
is $x$ powerful?	<code>ispowerful(<math>x</math>)</code>
is $x$ a totient? ( $x = \varphi(n)$ )	<code>istotient(<math>x, \{\&amp;n\}</math>)</code>
is $x$ a polygonal number? ( $x = P(s, n)$ )	<code>ispolygonal(<math>x, s, \{\&amp;n\}</math>)</code>
is $pol$ irreducible?	<code>polisirreducible(<math>pol</math>)</code>

Graphic Functions

crude graph of $expr$ between $a$ and $b$	<code>plot(<math>X = a, b, expr</math>)</code>
High-resolution plot (immediate plot)	
plot $expr$ between $a$ and $b$	<code>plotth(<math>X = a, b, expr, \{flag\}, \{n\}</math>)</code>
plot points given by lists $lx, ly$	<code>plotthraw(<math>lx, ly, \{flag\}</math>)</code>
terminal dimensions	<code>plotsizes()</code>

Rectwindow functions

init window $w$ , with size $x, y$	<code>plotinit(<math>w, x, y</math>)</code>
erase window $w$	<code>plotkill(<math>w</math>)</code>
copy $w$ to $w_2$ with offset $(dx, dy)$	<code>plotcopy(<math>w, w_2, dx, dy</math>)</code>
scale contents of $w$	<code>plotclip(<math>w</math>)</code>
scale coordinates in $w$	<code>plotscale(<math>w, x_1, x_2, y_1, y_2</math>)</code>
plotth in $w$	<code>plotrecth(<math>w, X = a, b, expr, \{flag\}, \{n\}</math>)</code>
plotthraw in $w$	<code>plotrecthraw(<math>w, data, \{flag\}</math>)</code>
draw window $w_1$ at $(x_1, y_1), \dots$	<code>plotdraw(<math>[[w_1, x_1, y_1], \dots]</math>)</code>

Low-level Rectwindow Functions

set current drawing color in $w$ to $c$	<code>plotcolor(<math>w, c</math>)</code>
current position of cursor in $w$	<code>plotcursor(<math>w</math>)</code>
write $s$ at cursor's position	<code>plotstring(<math>w, s</math>)</code>
move cursor to $(x, y)$	<code>plotmove(<math>w, x, y</math>)</code>
move cursor to $(x + dx, y + dy)$	<code>plotrmove(<math>w, dx, dy</math>)</code>
draw a box to $(x_2, y_2)$	<code>plotbox(<math>w, x_2, y_2</math>)</code>
draw a box to $(x + dx, y + dy)$	<code>plotrbox(<math>w, dx, dy</math>)</code>
draw polygon	<code>plotlines(<math>w, lx, ly, \{flag\}</math>)</code>
draw points	<code>plotpoints(<math>w, lx, ly</math>)</code>
draw ellipse	<code>plotarc(<math>w, lx, ly, \{flag\}</math>)</code>
draw line to $(x + dx, y + dy)$	<code>plotrline(<math>w, dx, dy</math>)</code>
draw point $(x + dx, y + dy)$	<code>plotrpoint(<math>w, dx, dy</math>)</code>

Convert to Postscript or Scalable Vector Graphics

The format  $f$  is either "ps" or "svg".  
as plotth      `plotthexport( $f, X = a, b, expr, \{flag\}, \{n\}$ )`  
as plotthraw      `plotthrawexport( $f, lx, ly, \{flag\}$ )`  
as plotdraw      `plotexport( $f, [[w_1, x_1, y_1], \dots]$ )`