NumPy for R (and S-Plus) users

Help

R/S-Plus	Python	Description
help.start()	help()	Browse help interactively
help()	help	Help on using help
help(plot) *or* ?plot	help(plot) *or* ?plot	Help for a function
help(package='splines')	help(pylab)	Help for a toolbox/library package
demo()		Demonstration examples
example(plot)		Example using a function

Searching available documentation

R/S-Plus	Python	Description
help.search('plot')		Search help files
apropos('plot')		Find objects by partial name
library()	help(); modules [Numeric]	List available packages
find(plot)	help(plot)	Locate functions
methods(plot)		List available methods for a function

Using interactively

R/S-Plus	Python	Description
Rgui	ipython -pylab	Start session
	TAB	Auto completion
source('foo.R')	execfile('foo.py') *or* run foo.py	Run code from file
history()	hist -n	Command history
savehistory(file=".Rhistory")		Save command history

q(save='no')	CTRL-D
	CTRL-Z # windows
	sys.exit()
		End session

Operators

R/S-Plus	Python	Description
help(Syntax)		Help on operator syntax

Arithmetic operators

R/S-Plus	Python	Description
a<-1; b<-2	a=1; b=1	Assignment; defining a number
a + b	a + b *or* add(a,b)	Addition
a - b	a - b *or* subtract(a,b)	Subtraction
a * b	a * b *or* multiply(a,b)	Multiplication
a / b	a / b *or* divide(a,b)	Division

`a ^ b`	`a ** b`
	`power(a,b)`
	`pow(a,b)`
		Power, a^b

`a %% b`	`a % b`
	`remainder(a,b)`
	`fmod(a,b)`
		Remainder

`a %/% b`		Integer division
	`a+=b` *or* `add(a,b,a)`	In place operation to save array creation overhead
`factorial(a)`		Factorial, $n!$

Relational operators

R/S-Plus	Python	Description
`a == b`	`a == b` *or* `equal(a,b)`	Equal
`a < b`	`a < b` *or* `less(a,b)`	Less than
`a > b`	`a > b` *or* `greater(a,b)`	Greater than
`a <= b`	`a <= b` *or* `less_equal(a,b)`	Less than or equal
`a >= b`	`a >= b` *or* `greater_equal(a,b)`	Greater than or equal
`a != b`	`a != b` *or* `not_equal(a,b)`	Not Equal

Logical operators

R/S-Plus	Python	Description
`a && b`	`a and b`	Short-circuit logical AND
`a || b`	`a or b`	Short-circuit logical OR
`a & b`	`logical_and(a,b)` *or* `a and b`	Element-wise logical AND
`a | b`	`logical_or(a,b)` *or* `a or b`	Element-wise logical OR
`xor(a, b)`	`logical_xor(a,b)`	Logical EXCLUSIVE OR
`!a`	`logical_not(a)` *or* `not a`	Logical NOT

root and logarithm

R/S-Plus	Python	Description
`sqrt(a)`	`math.sqrt(a)`	Square root
`log(a)`	`math.log(a)`	Logarithm, base e (natural)
`log10(a)`	`math.log10(a)`	Logarithm, base 10
`log2(a)`	`math.log(a, 2)`	Logarithm, base 2 (binary)
`exp(a)`	`math.exp(a)`	Exponential function

Round off

R/S-Plus	Python	Description
`round(a)`	`around(a)` *or* `math.round(a)`	Round
`ceil(a)`	`ceil(a)`	Round up
`floor(a)`	`floor(a)`	Round down
	`fix(a)`	Round towards zero

Mathematical constants

R/S-Plus	Python	Description
pi	math.pi	$\pi=3.141592$
exp(1)	math.e *or* math.exp(1)	$e=2.718281$

Missing values; IEEE-754 floating point status flags

R/S-Plus	Python	Description
	nan	Not a Number
	inf	Infinity, ∞
	plus_inf	Infinity, $+\infty$
	minus_inf	Infinity, $-\infty$
	plus_zero	Plus zero, $+0$
	minus_zero	Minus zero, -0

Complex numbers

R/S-Plus	Python	Description
1i	z = 1j	Imaginary unit
z <- 3+4i	z = 3+4j *or* z = complex(3,4)	A complex number, $3+4i$
abs(3+4i) *or* Mod(3+4i)	abs(3+4j)	Absolute value (modulus)
Re(3+4i)	z.real	Real part
Im(3+4i)	z.imag	Imaginary part
Arg(3+4i)		Argument
Conj(3+4i)	z.conj(); z.conjugate()	Complex conjugate

Trigonometry

R/S-Plus	Python	Description
atan2(b,a)	atan2(b,a)	Arctangent, $\arctan(b/a)$
	hypot(x,y)	Hypotenus; Euclidean distance

Generate random numbers

R/S-Plus	Python	Description
runif(10)	random.random((10,)) random.uniform((10,))	Uniform distribution
runif(10, min=2, max=7)	random.uniform(2,7,(10,))	Uniform: Numbers between 2 and 7
matrix(runif(36),6)	random.uniform(0,1,(6,6))	Uniform: 6,6 array
rnorm(10)	random.standard_normal((10,))	Normal distribution

Vectors

R/S-Plus	Python	Description
a <- c(2,3,4,5)	a=array([2,3,4,5])	Row vector, $1 \times n$-matrix
adash <- t(c(2,3,4,5))	array([2,3,4,5])[:,NewAxis] array([2,3,4,5]).reshape(-1,1) r_[1:10,'c']	Column vector, $m \times 1$-matrix

Sequences

R/S-Plus	Python	Description
seq(10) *or* 1:10	arange(1,11, dtype=Float) range(1,11)	1,2,3, ... ,10
seq(0,length=10)	arange(10.)	0.0,1.0,2.0, ... ,9.0
seq(1,10,by=3)	arange(1,11,3)	1,4,7,10
seq(10,1) *or* 10:1	arange(10,0,-1)	10,9,8, ... ,1
seq(from=10,to=1,by=-3)	arange(10,0,-3)	10,7,4,1
seq(1,10,length=7)	linspace(1,10,7)	Linearly spaced vector of n=7 points
rev(a)	a[::-1] *or*	Reverse
	a.fill(3), a[:] = 3	Set all values to same scalar value

Concatenation (vectors)

R/S-Plus	Python	Description
c(a,a)	concatenate((a,a))	Concatenate two vectors
c(1:4,a)	concatenate((range(1,5),a), axis=1)	

Repeating

R/S-Plus	Python	Description
rep(a,times=2)	concatenate((a,a))	1 2 3, 1 2 3
rep(a,each=3)	a.repeat(3) *or*	1 1 1, 2 2 2, 3 3 3
rep(a,a)	a.repeat(a) *or*	1, 2 2, 3 3 3

Miss those elements out

R/S-Plus	Python	Description
a[-1]	a[1:]	miss the first element
a[-10]		miss the tenth element
a[-seq(1,50,3)]		miss 1,4,7, ...
	a[-1]	last element
	a[-2:]	last two elements

Maximum and minimum

R/S-Plus	Python	Description
pmax(a,b)	maximum(a,b)	pairwise max
max(a,b)	concatenate((a,b)).max()	max of all values in two vectors
v <- max(a) ; i <- which.max(a)	v,i = a.max(0),a.argmax(0)	

Vector multiplication

R/S-Plus	Python	Description
a*a	a*a	Multiply two vectors
	dot(u,v)	Vector dot product, $u \cdot v$

Matrices

R/S-Plus	Python	Description
`rbind(c(2,3),c(4,5))` `array(c(2,3,4,5), dim=c(2,2))`	`a = array([[2,3],[4,5]])`	Define a matrix

Concatenation (matrices); rbind and cbind

R/S-Plus	Python	Description
`rbind(a,b)`	`concatenate((a,b), axis=0)` `vstack((a,b))`	Bind rows
`cbind(a,b)`	`concatenate((a,b), axis=1)` `hstack((a,b))`	Bind columns
	`concatenate((a,b), axis=2)` `dstack((a,b))`	Bind slices (three-way arrays)
	`concatenate((a,b), axis=None)`	Concatenate matrices into one vector
`rbind(1:4,1:4)`	`concatenate((r_[1:5],r_[1:5])).reshape(2,-1)` `vstack((r_[1:5],r_[1:5]))`	Bind rows (from vectors)
`cbind(1:4,1:4)`		Bind columns (from vectors)

Array creation

R/S-Plus	Python	Description
`matrix(0,3,5)` *or* `array(0,c(3,5))`	`zeros((3,5),Float)`	0 filled array
	`zeros((3,5))`	0 filled array of integers
`matrix(1,3,5)` *or* `array(1,c(3,5))`	`ones((3,5),Float)`	1 filled array
`matrix(9,3,5)` *or* `array(9,c(3,5))`		Any number filled array
`diag(1,3)`	`identity(3)`	Identity matrix
`diag(c(4,5,6))`	`diag((4,5,6))`	Diagonal
	`a = empty((3,3))`	Empty array

Reshape and flatten matrices

R/S-Plus	Python	Description
`matrix(1:6,nrow=3,byrow=T)`	`arange(1,7).reshape(2,-1)` `a.setshape(2,3)`	Reshaping (rows first)
`matrix(1:6,nrow=2)` `array(1:6,c(2,3))`	`arange(1,7).reshape(-1,2).transpose()`	Reshaping (columns first)
`as.vector(t(a))`	`a.flatten()` *or*	Flatten to vector (by rows, like comics)
`as.vector(a)`	`a.flatten(1)`	Flatten to vector (by columns)
`a[row(a) <= col(a)]`		Flatten upper triangle (by columns)

Shared data (slicing)

R/S-Plus	Python	Description
`b = a`	`b = a.copy()`	Copy of a

Indexing and accessing elements (Python: slicing)

R/S-Plus	Python	Description

R/S-Plus	Python	Description
a <- rbind(c(11, 12, 13, 14), c(21, 22, 23, 24), c(31, 32, 33, 34))	a = array([[11, 12, 13, 14], [21, 22, 23, 24], [31, 32, 33, 34]])	Input is a 3,4 array
a[2,3]	a[1,2]	Element 2,3 (row,col)
a[1,]	a[0,]	First row
a[,1]	a[:,0]	First column
	a.take([0,2]).take([0,3], axis=1)	Array as indices
a[-1,]	a[1:,]	All, except first row
	a[-2:,]	Last two rows
	a[::2,:]	Strides: Every other row
	a[...,2]	Third in last dimension (axis)
a[-2,-3]		All, except row,column (2,3)
a[,-2]	a.take([0,2,3],axis=1)	Remove one column
	a.diagonal(offset=0)	Diagonal

Assignment

R/S-Plus	Python	Description
a[,1] <- 99	a[:,0] = 99	
a[,1] <- c(99,98,97)	a[:,0] = array([99,98,97])	
a[a>90] <- 90	(a>90).choose(a,90) a.clip(min=None, max=90)	Clipping: Replace all elements over 90
	a.clip(min=2, max=5)	Clip upper and lower values

Transpose and inverse

R/S-Plus	Python	Description
t(a)	a.conj().transpose()	Transpose
	a.transpose()	Non-conjugate transpose
det(a)	linalg.det(a) *or*	Determinant
solve(a)	linalg.inv(a) *or*	Inverse
ginv(a)	linalg.pinv(a)	Pseudo-inverse
	norm(a)	Norms
eigen(a)$values	linalg.eig(a)[0]	Eigenvalues
svd(a)$d	linalg.svd(a)	Singular values
	linalg.cholesky(a)	Cholesky factorization
eigen(a)$vectors	linalg.eig(a)[1]	Eigenvectors
rank(a)	rank(a)	Rank

Sum

R/S-Plus	Python	Description
apply(a,2,sum)	a.sum(axis=0)	Sum of each column
apply(a,1,sum)	a.sum(axis=1)	Sum of each row
sum(a)	a.sum()	Sum of all elements
	a.trace(offset=0)	Sum along diagonal
apply(a,2,cumsum)	a.cumsum(axis=0)	Cumulative sum (columns)

Sorting

R/S-Plus	Python	Description
	`a = array([[4,3,2],[2,8,6],[1,4,7]])`	Example data
`t(sort(a))`	`a.ravel().sort()` *or*	Flat and sorted
`apply(a,2,sort)`	`a.sort(axis=0)` *or* `msort(a)`	Sort each column
`t(apply(a,1,sort))`	`a.sort(axis=1)`	Sort each row
	`a[a[:,0].argsort(),]`	Sort rows (by first row)
`order(a)`	`a.ravel().argsort()`	Sort, return indices
	`a.argsort(axis=0)`	Sort each column, return indices
	`a.argsort(axis=1)`	Sort each row, return indices

Maximum and minimum

R/S-Plus	Python	Description
`apply(a,2,max)`	`a.max(0)` *or* `amax(a [,axis=0])`	max in each column
`apply(a,1,max)`	`a.max(1)` *or* `amax(a, axis=1)`	max in each row
`max(a)`	`a.max()` *or*	max in array
`i <- apply(a,1,which.max)`		return indices, i
`pmax(b,c)`	`maximum(b,c)`	pairwise max
`apply(a,2,cummax)`		
	`a.ptp(); a.ptp(0)`	max-to-min range

Matrix manipulation

R/S-Plus	Python	Description
`a[,4:1]`	`fliplr(a)` *or* `a[:,::-1]`	Flip left-right
`a[3:1,]`	`flipud(a)` *or* `a[::-1,]`	Flip up-down
	`rot90(a)`	Rotate 90 degrees
`kronecker(matrix(1,2,3),a)`	`kron(ones((2,3)),a)`	Repeat matrix: [a a a ; a a a]
`a[lower.tri(a)] <- 0`	`triu(a)`	Triangular, upper
`a[upper.tri(a)] <- 0`	`tril(a)`	Triangular, lower

Equivalents to "size"

R/S-Plus	Python	Description
`dim(a)`	`a.shape` *or* `a.getshape()`	Matrix dimensions
`ncol(a)`	`a.shape[1]` *or* `size(a, axis=1)`	Number of columns
`prod(dim(a))`	`a.size` *or* `size(a[, axis=None])`	Number of elements
	`a.ndim`	Number of dimensions
`object.size(a)`	`a.nbytes`	Number of bytes used in memory

Matrix- and elementwise- multiplication

R/S-Plus	Python	Description
`a * b`	`a * b` *or* `multiply(a,b)`	Elementwise operations
`a %*% b`	`matrixmultiply(a,b)`	Matrix product (dot product)
	`inner(a,b)` *or*	Inner matrix vector multiplication $a\cdot b'$
`outer(a,b)` *or* `a %o% b`	`outer(a,b)` *or*	Outer product

`crossprod(a,b)` *or* `t(a) %*% b`		Cross product
`kronecker(a,b)`	`kron(a,b)`	Kronecker product
`solve(a,b)`	`linalg.solve(a,b)`	Left matrix division, $b^{-1}{\cdot}a$ \newline (solve linear equations)
	`vdot(a,b)`	Vector dot product
	`cross(a,b)`	Cross product

Find; conditional indexing

R/S-Plus	Python	Description
`which(a != 0)`	`a.ravel().nonzero()`	Non-zero elements, indices
`which(a != 0, arr.ind=T)`	`(i,j) = a.nonzero()` `(i,j) = where(a!=0)`	Non-zero elements, array indices
`ij <- which(a != 0, arr.ind=T);` `v <- a[ij]`	`v = a.compress((a!=0).flat)` `v = extract(a!=0,a)`	Vector of non-zero values
`which(a>5.5)`	`(a>5.5).nonzero()`	Condition, indices
`ij <- which(a>5.5, arr.ind=T); v <- a[ij]`	`a.compress((a>5.5).flat)`	Return values
	`where(a>5.5,0,a)` *or* `a * (a>5.5)`	Zero out elements above 5.5
	`a.put(2,indices)`	Replace values

Multi-way arrays

R/S-Plus	Python	Description
	`a = array([[[1,2],[1,2]], [[3,4],[3,4]]])`	Define a 3-way array
	`a[0,...]`	

File input and output

R/S-Plus	Python	Description
`f <- read.table("data.txt")`	`f = fromfile("data.txt")` `f = load("data.txt")`	Reading from a file (2d)
`f <- read.table("data.txt")`	`f = load("data.txt")`	Reading from a file (2d)
`f <- read.table(file="data.csv", sep=";")`	`f = load('data.csv', delimiter=';')`	Reading fram a CSV file (2d)
`write(f,file="data.txt")`	`save('data.csv', f, fmt='%.6f', delimiter=';')`	Writing to a file (2d)
	`f.tofile(file='data.csv', format='%.6f', sep=';')`	Writing to a file (1d)
	`f = fromfile(file='data.csv', sep=';')`	Reading from a file (1d)

Plotting

Basic x-y plots

R/S-Plus	Python	Description
`plot(a, type="l")`	`plot(a)`	1d line plot
`plot(x[,1],x[,2])`	`plot(x[:,0],x[:,1],'o')`	2d scatter plot

R/S-Plus	Python	Description
	`plot(x1,y1,'bo', x2,y2,'go')`	Two graphs in one plot

`plot(x1,y1)`
`matplot(x2,y2,add=T)`
	`plot(x1,y1,'o')`
	`plot(x2,y2,'o')`
	`show() # as normal`
		Overplotting: Add new plots to current

	`subplot(211)`	subplots
`plot(x,y,type="b",col="red")`	`plot(x,y,'ro-')`	Plotting symbols and color

Axes and titles

R/S-Plus	Python	Description
`grid()`	`grid()`	Turn on grid lines
`plot(c(1:10,10:1), asp=1)`	`figure(figsize=(6,6))`	1:1 aspect ratio
`plot(x,y, xlim=c(0,10),` `ylim=c(0,5))`	`axis([ 0, 10, 0, 5 ])`	Set axes manually
`plot(1:10, main="title",` `xlab="x-axis", ylab="y-axis")`		Axis labels and titles
	`text(2,25,'hello')`	Insert text

Log plots

R/S-Plus	Python	Description
`plot(x,y, log="y")`	`semilogy(a)`	logarithmic y-axis
`plot(x,y, log="x")`	`semilogx(a)`	logarithmic x-axis
`plot(x,y, log="xy")`	`loglog(a)`	logarithmic x and y axes

Filled plots and bar plots

R/S-Plus	Python	Description

`plot(t,s, type="n", xlab="",`
`ylab="")`
`polygon(t,s, col="lightblue")`
`polygon(t,c, col="lightgreen")`
	`fill(t,s,'b', t,c,'g',`
	`alpha=0.2)`
		Filled plot

`stem(x[,3])`		Stem-and-Leaf plot

Functions

R/S-Plus	Python	Description
`f <- function(x) sin(x/3) -` `cos(x/5)`		Defining functions

`plot(f, xlim=c(0,40), type='p')`	`x = arrayrange(0,40,.5)`
	`y = sin(x/3) - cos(x/5)`
	`plot(x,y, 'o')`
		Plot a function for given range

Polar plots

R/S-Plus	Python	Description
	`theta = arange(0,2*pi,0.001)` `r = sin(2*theta)`	
	`polar(theta, rho)`	

Histogram plots

R/S-Plus	Python	Description
`hist(rnorm(1000))`		
`hist(rnorm(1000), breaks= -4:4)`		
`hist(rnorm(1000), breaks=c(seq(-5,0,0.25), seq(0.5,5,0.5)), freq=F)`		
`plot(apply(a,1,sort),type="l")`		

3d data

Contour and image plots

R/S-Plus	Python	Description
`contour(z)`	`levels, colls = contour(Z, V, origin='lower', extent= (-3,3,-3,3))` `clabel(colls, levels, inline=1, fmt='%1.1f', fontsize=10)`	Contour plot
`filled.contour(x,y,z, nlevels=7, color=gray.colors)`	`contourf(Z, V, cmap=cm.gray, origin='lower', extent=(-3,3,-3,3))`	Filled contour plot
`image(z, col=gray.colors(256))`	`im = imshow(Z, interpolation='bilinear', origin='lower', extent=(-3,3,-3,3))`	Plot image data
	`# imshow() and contour() as above`	Image with contours
	`quiver()`	Direction field vectors

Perspective plots of surfaces over the x-y plane

R/S-Plus	Python	Description
`f <- function(x,y) x*exp(-x^2- y^2)` `n <- seq(-2,2, length=40)` `z <- outer(n,n,f)`	`n=arrayrange(-2,2,.1)` `[x,y] = meshgrid(n,n)` `z = x*power(math.e,-x**2-y**2)`	
`persp(x,y,z, theta=30, phi=30, expand=0.6, ticktype='detailed')`		Mesh plot
`persp(x,y,z, theta=30, phi=30, expand=0.6, col='lightblue', shade=0.75, ltheta=120, ticktype='detailed')`		Surface plot

Scatter (cloud) plots

R/S-Plus	Python	Description
`cloud(z~x*y)`		3d scatter plot

Save plot to a graphics file

R/S-Plus	Python	Description

| ```
postscript(file="foo.eps")
plot(1:10)
dev.off()
``` | `savefig('foo.eps')` | PostScript |
`pdf(file='foo.pdf')`	`savefig('foo.pdf')`	PDF
`devSVG(file='foo.svg')`	`savefig('foo.svg')`	SVG (vector graphics for www)
`png(filename = "Rplot%03d.png"`	`savefig('foo.png')`	PNG (raster graphics)

Data analysis

Set membership operators

R/S-Plus	Python	Description
| ```
a <- c(1,2,2,5,2)
b <- c(2,3,4)
``` | ```
a = array([1,2,2,5,2])
b = array([2,3,4])
a = set([1,2,2,5,2])
b = set([2,3,4])
``` | Create sets |
| `unique(a)` | ```
unique1d(a)
unique(a)
set(a)
``` | Set unique |
| `union(a,b)` | ```
union1d(a,b)
a.union(b)
``` | Set union |
| `intersect(a,b)` | ```
intersect1d(a)
a.intersection(b)
``` | Set intersection |
| `setdiff(a,b)` | ```
setdiff1d(a,b)
a.difference(b)
``` | Set difference |
| `setdiff(union(a,b),intersect(a,b))` | ```
setxor1d(a,b)
a.symmetric_difference(b)
``` | Set exclusion |
| `is.element(2,a)` *or* `2 %in% a` | ```
2 in a
setmember1d(2,a)
contains(a,2)
``` | True for set member |

Statistics

R/S-Plus	Python	Description
| `apply(a,2,mean)` | ```
a.mean(axis=0)
mean(a [,axis=0])
``` | Average |
`apply(a,2,median)`	`median(a)` *or* `median(a [,axis=0])`	Median
`apply(a,2,sd)`	`a.std(axis=0)` *or* `std(a [,axis=0])`	Standard deviation
`apply(a,2,var)`	`a.var(axis=0)` *or* `var(a)`	Variance
`cor(x,y)`	`correlate(x,y)` *or* `corrcoef(x,y)`	Correlation coefficient
`cov(x,y)`	`cov(x,y)`	Covariance

Interpolation and regression

R/S-Plus	Python	Description
| ```
z <- lm(y~x)
plot(x,y)
abline(z)
``` | ```
(a,b) = polyfit(x,y,1)
plot(x,y,'o', x,a*x+b,'-')
``` | Straight line fit |
| `solve(a,b)` | `linalg.lstsq(x,y)` | Linear least squares $y = ax + b$ |
| | `polyfit(x,y,3)` | Polynomial fit |

Non-linear methods

Polynomials, root finding

R/S-Plus	Python	Description
	`poly()`	Polynomial
`polyroot(c(1,-1,-1))`	`roots()`	Find zeros of polynomial
	`polyval(array([1,2,1,2]),arange(1,11))`	Evaluate polynomial

Differential equations

R/S-Plus	Python	Description
	`diff(x, n=1, axis=0)`	Discrete difference function and approximate derivative

Fourier analysis

R/S-Plus	Python	Description
`fft(a)`	`fft(a)` *or*	Fast fourier transform
`fft(a, inverse=TRUE)`	`ifft(a)` *or*	Inverse fourier transform
	`convolve(x,y)`	Linear convolution

Symbolic algebra; calculus

R/S-Plus Python Description

Programming

R/S-Plus	Python	Description
`.R`	`.py`	Script file extension
`#`	`#`	Comment symbol (rest of line)
`library(RSvgDevice)`	`from pylab import *`	Import library functions
`string <- "a <- 234"` `eval(parse(text=string))`	`string="a=234"` `eval(string)`	Eval

Loops

R/S-Plus	Python	Description
`for(i in 1:5) print(i)`	`for i in range(1,6): print(i)`	for-statement

`for(i in 1:5) {`
`print(i)`
`print(i*2)`
`}`
	`for i in range(1,6):`
	`print(i)`
	`print(i*2)`
		Multiline for statements

Conditionals

R/S-Plus	Python	Description
`if (1>0) a <- 100`	`if 1>0: a=100`	if-statement
`ifelse(a>0,a,0)`		Ternary operator (if?true:false)

Debugging

R/S-Plus	Python	Description
`.Last.value`		Most recent evaluated expression
`objects()`		List variables loaded into memory
`rm(x)`		Clear variable x from memory
`print(a)`	`print a`	Print

Working directory and OS

R/S-Plus	Python	Description
`list.files()` *or* `dir()`	`os.listdir(".")`	List files in directory
`list.files(pattern="\.r$")`	`grep.grep("*.py")`	List script files in directory
`getwd()`	`os.getcwd()`	Displays the current working directory
`setwd('foo')`	`os.chdir('foo')`	Change working directory
`system("notepad")`	`os.system('notepad')` `os.popen('notepad')`	Invoke a System Command