NumPy for IDL users

Help

IDL	Python	Description
`?`	`help()`	Browse help interactively
`?help`	`help`	Help on using help
`?plot` *or* `man,'plot`	`help(plot)` *or* `?plot`	Help for a function
	`help(pylab)`	Help for a toolbox/library package
`demo`		Demonstration examples

Searching available documentation

IDL	Python	Description
	`help(); modules [Numeric]`	List available packages
	`help(plot)`	Locate functions

Using interactively

IDL	Python	Description
`idlde`	`ipython -pylab`	Start session
	`TAB`	Auto completion
`@"foo.idlbatch"` *or* `.run 'foo.pro'`	`execfile('foo.py')` *or* `run foo.py`	Run code from file
`help,/rec`	`hist -n`	Command history
`journal,'IDLhistory'`		Save command history

`exit` *or* `CTRL-D`	`CTRL-D`
	`CTRL-Z # windows`
	`sys.exit()`
		End session

Operators

IDL Python Description

Arithmetic operators

IDL	Python	Description
`a=1 & b=1`	`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 MOD b`	`a % b`
	`remainder(a,b)`
	`fmod(a,b)`
		Remainder

`++a` *or* `a+=1`		Increment, return new value
`a++`		Increment, return old value
`a+=1`	`a+=b` *or* `add(a,b,a)`	In place operation to save array creation overhead

Relational operators

IDL	Python	Description
`a eq b`	`a == b` *or* `equal(a,b)`	Equal
`a lt b`	`a < b` *or* `less(a,b)`	Less than
`a gt b`	`a > b` *or* `greater(a,b)`	Greater than
`a le b`	`a <= b` *or* `less_equal(a,b)`	Less than or equal
`a ge b`	`a >= b` *or* `greater_equal(a,b)`	Greater than or equal
`a ne b`	`a != b` *or* `not_equal(a,b)`	Not Equal

Logical operators

IDL	Python	Description
	`a and b`	Short-circuit logical AND
	`a or b`	Short-circuit logical OR
`a and b`	`logical_and(a,b)` *or* `a and b`	Element-wise logical AND
`a or b`	`logical_or(a,b)` *or* `a or b`	Element-wise logical OR
`a xor b`	`logical_xor(a,b)`	Logical EXCLUSIVE OR
`not a`	`logical_not(a)` *or* `not a`	Logical NOT

root and logarithm

IDL	Python	Description
`sqrt(a)`	`math.sqrt(a)`	Square root
`alog(a)`	`math.log(a)`	Logarithm, base e (natural)
`alog10(a)`	`math.log10(a)`	Logarithm, base 10
	`math.log(a, 2)`	Logarithm, base 2 (binary)
`exp(a)`	`math.exp(a)`	Exponential function

Round off

IDL	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

IDL	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

IDL	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

IDL	Python	Description
`complex(0,1)`	`z = 1j`	Imaginary unit
`z = complex(3,4)`	`z = 3+4j` *or* `z = complex(3,4)`	A complex number, $3+4i$
`abs(z)`	`abs(3+4j)`	Absolute value (modulus)
`real_part(z)`	`z.real`	Real part
`imaginary(z)`	`z.imag`	Imaginary part
`conj(z)`	`z.conj(); z.conjugate()`	Complex conjugate

Trigonometry

IDL	Python	Description
	`atan2(b,a)`	Arctangent, $\arctan(b/a)$
	`hypot(x,y)`	Hypotenus; Euclidean distance

Generate random numbers

IDL	Python	Description
`randomu(seed, 10)`	`random.random((10,))` `random.uniform((10,))`	Uniform distribution
`2+5*randomu(seed, 10)`	`random.uniform(2,7,(10,))`	Uniform: Numbers between 2 and 7
`randomu(seed,[6,6])`	`random.uniform(0,1,(6,6))`	Uniform: 6,6 array
`randomn(seed, 10)`	`random.standard_normal((10,))`	Normal distribution

Vectors

IDL	Python	Description

`a = [2, 3, 4, 5]`	`a=array([2,3,4,5])`	Row vector, $1 \times n$-matrix

`transpose([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

IDL	Python	Description
`indgen(10)+1` `dindgen(10)+1`	`arange(1,11, dtype=Float)` `range(1,11)`	1,2,3, ... ,10
`dindgen(10)`	`arange(10.)`	0.0,1.0,2.0, ... ,9.0
`indgen(4)*3+1`	`arange(1,11,3)`	1,4,7,10
	`arange(10,0,-1)`	10,9,8, ... ,1
	`arange(10,0,-3)`	10,7,4,1
	`linspace(1,10,7)`	Linearly spaced vector of n=7 points
`reverse(a)`	`a[::-1]` *or*	Reverse
	`a.fill(3), a[:] = 3`	Set all values to same scalar value

Concatenation (vectors)

IDL	Python	Description
`[a,a]` *or* `rebin(a,2,size(a))`	`concatenate((a,a))`	Concatenate two vectors
`[indgen(3)+1,a]`	`concatenate((range(1,5),a), axis=1)`	

Repeating

IDL	Python	Description
	`concatenate((a,a))`	1 2 3, 1 2 3
	`a.repeat(3)` *or*	1 1 1, 2 2 2, 3 3 3
	`a.repeat(a)` *or*	1, 2 2, 3 3 3

Miss those elements out

IDL	Python	Description
	`a[1:]`	miss the first element
	`a[-1]`	last element
	`a[-2:]`	last two elements

Maximum and minimum

IDL	Python	Description
	`maximum(a,b)`	pairwise max
	`concatenate((a,b)).max()`	max of all values in two vectors

```
v,i = a.max(0),a.argmax(0)
```

Vector multiplication

IDL	Python	Description
	`a*a`	Multiply two vectors
`crossp(u,v)`		Vector cross product, $u \times v$
	`dot(u,v)`	Vector dot product, $u \cdot v$

Matrices

IDL	Python	Description
`a = [[2,3],[4,5]]`	`a = array([[2,3],[4,5]])`	Define a matrix

Concatenation (matrices); rbind and cbind

IDL	Python	Description
	`concatenate((a,b), axis=0)` `vstack((a,b))`	Bind rows
	`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
	`concatenate((r_[1:5],r_[1:5])).reshape(2,-1)` `vstack((r_[1:5],r_[1:5]))`	Bind rows (from vectors)

Array creation

IDL	Python	Description
`dblarr(3,5)`	`zeros((3,5),Float)`	0 filled array
`intarr(3,5)`	`zeros((3,5))`	0 filled array of integers
`dblarr(3,5)+1`	`ones((3,5),Float)`	1 filled array
`intarr(3,5)+9`		Any number filled array
`identity(3)`	`identity(3)`	Identity matrix
`diag_matrix([4,5,6])`	`diag((4,5,6))`	Diagonal
	`a = empty((3,3))`	Empty array

Reshape and flatten matrices

IDL	Python	Description
`reform(a,2,3)`	`arange(1,7).reshape(2,-1)` `a.setshape(2,3)`	Reshaping (rows first)
	`arange(1,7).reshape(-1,2).transpose()`	Reshaping (columns first)

	`a.flatten()` *or*	Flatten to vector (by rows, like comics)
	`a.flatten(1)`	Flatten to vector (by columns)

Shared data (slicing)

IDL	Python	Description
	`b = a.copy()`	Copy of a

Indexing and accessing elements (Python: slicing)

IDL	Python	Description

`a = [[ 11, 12, 13, 14 ], $`
`[ 21, 22, 23, 24 ], $`
`[ 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(2,1)`	`a[1,2]`	Element 2,3 (row,col)
`a(*,0)`	`a[0,]`	First row
`a(0,*)`	`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.take([0,2,3],axis=1)`	Remove one column
	`a.diagonal(offset=0)`	Diagonal

Assignment

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

Transpose and inverse

IDL	Python	Description
`transpose(a)`	`a.conj().transpose()`	Transpose
	`a.transpose()`	Non-conjugate transpose
`determ(a)`	`linalg.det(a)` *or*	Determinant
`invert(a)`	`linalg.inv(a)` *or*	Inverse
	`linalg.pinv(a)`	Pseudo-inverse
	`norm(a)`	Norms

`hqr(elmhes(a))`	`linalg.eig(a)[0]`	Eigenvalues
`svdc,A,w,U,V`	`linalg.svd(a)`	Singular values
	`linalg.cholesky(a)`	Cholesky factorization
	`linalg.eig(a)[1]`	Eigenvectors
	`rank(a)`	Rank

Sum

IDL	Python	Description
`total(a,2)`	`a.sum(axis=0)`	Sum of each column
`total(a,1)`	`a.sum(axis=1)`	Sum of each row
`total(a)`	`a.sum()`	Sum of all elements
	`a.trace(offset=0)`	Sum along diagonal
	`a.cumsum(axis=0)`	Cumulative sum (columns)

Sorting

IDL	Python	Description
	`a = array([[4,3,2],[2,8,6], [1,4,7]])`	Example data
	`a.ravel().sort()` *or*	Flat and sorted
`sort(a)`	`a.sort(axis=0)` *or* `msort(a)`	Sort each column
	`a.sort(axis=1)`	Sort each row
	`a[a[:,0].argsort(),]`	Sort rows (by first row)
	`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

IDL	Python	Description
`max(a,DIMENSION=2)`	`a.max(0)` *or* `amax(a [,axis=0])`	max in each column
`max(a,DIMENSION=1)`	`a.max(1)` *or* `amax(a, axis=1)`	max in each row
`max(a)`	`a.max()` *or*	max in array
	`maximum(b,c)`	pairwise max
	`a.ptp(); a.ptp(0)`	max-to-min range

Matrix manipulation

IDL	Python	Description
`reverse(a)`	`fliplr(a)` *or* `a[:,::-1]`	Flip left-right
`reverse(a,2)`	`flipud(a)` *or* `a[::-1,]`	Flip up-down
`rotate(a,1)`	`rot90(a)`	Rotate 90 degrees
	`kron(ones((2,3)),a)`	Repeat matrix: [a a a ; a a a]

	`triu(a)`	Triangular, upper
	`tril(a)`	Triangular, lower

Equivalents to "size"

IDL	Python	Description
`size(a)`	`a.shape` *or* `a.getshape()`	Matrix dimensions
`s=size(a) & s[1]`	`a.shape[1]` *or* `size(a, axis=1)`	Number of columns
`n_elements(a)`	`a.size` *or* `size(a[, axis=None])`	Number of elements
	`a.ndim`	Number of dimensions
	`a.nbytes`	Number of bytes used in memory

Matrix- and elementwise- multiplication

IDL	Python	Description
	`a * b` *or* `multiply(a,b)`	Elementwise operations
`a # b` *or* `b ## a`	`matrixmultiply(a,b)`	Matrix product (dot product)
`transpose(a) # b`	`inner(a,b)` *or*	Inner matrix vector multiplication $a\cdot b'$
`a # b`	`outer(a,b)` *or*	Outer product
	`kron(a,b)`	Kronecker product
`cramer(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

IDL	Python	Description
	`a.ravel().nonzero()`	Non-zero elements, indices
`where(a NE 0)`	`(i,j) = a.nonzero()` `(i,j) = where(a!=0)`	Non-zero elements, array indices
`a(where(a NE 0))`	`v = a.compress((a!=0).flat)` `v = extract(a!=0,a)`	Vector of non-zero values
`where(a GE 5.5)`	`(a>5.5).nonzero()`	Condition, indices
`a(where(a GE 5.5))`	`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

IDL	Python	Description
	`a = array([[[1,2],[1,2]], [[3,4],[3,4]]])`	Define a 3-way array

	`a[0,...]`	

File input and output

IDL	Python	Description
`read()`	`f = fromfile("data.txt")` `f = load("data.txt")`	Reading from a file (2d)
`read()`	`f = load("data.txt")`	Reading from a file (2d)
`x = read_ascii(data_start=1,delimiter=';')`	`f = load('data.csv', delimiter=';')`	Reading fram a CSV file (2d)
	`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

IDL	Python	Description
`plot, a`	`plot(a)`	1d line plot
`plot, x(1,*), x(2,*)`	`plot(x[:,0],x[:,1],'o')`	2d scatter plot
	`plot(x1,y1,'bo', x2,y2,'go')`	Two graphs in one plot

`plot, x1, y1`
`oplot, x2, y2`
	`plot(x1,y1,'o')`
	`plot(x2,y2,'o')`
	`show() # as normal`
		Overplotting: Add new plots to current

`!p.multi(0,2,1)`	`subplot(211)`	subplots
`plot, x,y, line=1, psym=-1`	`plot(x,y,'ro-')`	Plotting symbols and color

Axes and titles

IDL	Python	Description
	`grid()`	Turn on grid lines
	`figure(figsize=(6,6))`	1:1 aspect ratio
`plot, x(1,*), x(2,*), xran=[0,10], yran=[0,5]`	`axis([ 0, 10, 0, 5 ])`	Set axes manually
`plot, x,y, title='title', xtitle='x-axis', ytitle='y-axis'`		Axis labels and titles
`xyouts, 2,25, 'hello'`	`text(2,25,'hello')`	Insert text

Log plots

IDL	Python	Description

`plot, x,y, /YLOG` *or* `plot_io, x,y`	`semilogy(a)`	logarithmic y-axis
`plot, x,y, /XLOG` *or* `plot_oi, x,y`	`semilogx(a)`	logarithmic x-axis
`plot_oo, x,y`	`loglog(a)`	logarithmic x and y axes

Filled plots and bar plots

IDL	Python	Description
	`fill(t,s,'b', t,c,'g', alpha=0.2)`	Filled plot

Functions

IDL	Python	Description

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

Polar plots

IDL	Python	Description

	`theta = arange(0,2*pi,0.001)`
	`r = sin(2*theta)`
	`polar(theta, rho)`	

Histogram plots

IDL	Python	Description
`plot, histogram(randomn(5,1000))`		

3d data

Contour and image plots

IDL	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
`contour, z, nlevels=7, /fill` `contour, z, nlevels=7, /overplot, /downhill`	`contourf(Z, V, cmap=cm.gray, origin='lower', extent=(-3,3,-3,3))`	Filled contour plot
`tv, z` `loadct,0`	`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

IDL	Python	Description

	`n=arrayrange(-2,2,.1)`
	`[x,y] = meshgrid(n,n)`
	`z = x*power(math.e,-x**2-y**2)`	

`surface, z`		Mesh plot
`shade_surf, z` `loadct,3`		Surface plot

Scatter (cloud) plots

IDL Python Description

Save plot to a graphics file

IDL	Python	Description

`set_plot,'PS'`
`device, file='foo.eps', /land`
`plot x,y`
`device,/close &`
`set_plot,'win'`
	`savefig('foo.eps')`	PostScript

	`savefig('foo.pdf')`	PDF
	`savefig('foo.svg')`	SVG (vector graphics for www)
	`savefig('foo.png')`	PNG (raster graphics)

Data analysis

Set membership operators

IDL	Python	Description

	`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

	`unique1d(a)`
	`unique(a)`
	`set(a)`
		Set unique

	`union1d(a,b)` `a.union(b)`	Set union
	`intersect1d(a)` `a.intersection(b)`	Set intersection

	`setdiff1d(a,b)` `a.difference(b)`	Set difference
	`setxor1d(a,b)` `a.symmetric_difference(b)`	Set exclusion

	`2 in a`
	`setmember1d(2,a)`
	`contains(a,2)`
		True for set member

Statistics

IDL	Python	Description
`mean(a)`	`a.mean(axis=0)` `mean(a [,axis=0])`	Average
`median(a)`	`median(a)` *or* `median(a [,axis=0])`	Median
`stddev(a)`	`a.std(axis=0)` *or* `std(a [,axis=0])`	Standard deviation
`variance(a)`	`a.var(axis=0)` *or* `var(a)`	Variance
`correlate(x,y)`	`correlate(x,y)` *or* `corrcoef(x,y)`	Correlation coefficient
	`cov(x,y)`	Covariance

Interpolation and regression

IDL	Python	Description
`poly_fit(x,y,1)`	`(a,b) = polyfit(x,y,1)` `plot(x,y,'o', x,a*x+b,'-')`	Straight line fit
	`linalg.lstsq(x,y)`	Linear least squares $y = ax + b$
	`polyfit(x,y,3)`	Polynomial fit

Non-linear methods

Polynomials, root finding

IDL	Python	Description
	`poly()`	Polynomial
	`roots()`	Find zeros of polynomial
	`polyval(array([1,2,1,2]),arange(1,11))`	Evaluate polynomial

Differential equations

IDL	Python	Description
	`diff(x, n=1, axis=0)`	Discrete difference function and approximate derivative

Fourier analysis

IDL	Python	Description
`fft(a)`	`fft(a)` *or*	Fast fourier transform
`fft(a),/inverse`	`ifft(a)` *or*	Inverse fourier transform
`convol()`	`convolve(x,y)`	Linear convolution

Symbolic algebra; calculus

IDL Python Description

Programming

IDL	Python	Description
`.idlbatch`	`.py`	Script file extension
`;`	`#`	Comment symbol (rest of line)
	`from pylab import *`	Import library functions
	`string="a=234"` `eval(string)`	Eval

Loops

IDL	Python	Description
`for k=1,5 do print,k`	`for i in range(1,6): print(i)`	for-statement

`for k=1,5 do begin $`
`print, i &$`
`print, i*2 &$`
`end`
	`for i in range(1,6):`
	`print(i)`
	`print(i*2)`
		Multiline for statements

Conditionals

IDL	Python	Description
`if 1 gt 0 then a=100`	`if 1>0: a=100`	if-statement
`if 1 gt 0 then a=100 else a=0`		if-else-statement
`a>0?a:0`		Ternary operator (if?true:false)

Debugging

IDL	Python	Description
`help`		List variables loaded into memory
`print, a`	`print a`	Print

Working directory and OS

IDL	Python	Description
`dir`	`os.listdir(".")`	List files in directory
	`grep.grep("*.py")`	List script files in directory
`sd`	`os.getcwd()`	Displays the current working directory

`cd,'foo` *or* `sd,'foo`	`os.chdir('foo')`	Change working directory
`spawn,'notepad'`	`os.system('notepad')` `os.popen('notepad')`	Invoke a System Command