IDL

It saves time for science

An Introduction to Interactive Data Language

http://www.ast.cam.ac.uk/~vasily/idl.htm

The (brief) history of IDL

- 1970s: David Stern at LASP, Boulder creates *Rufus* and *SOL*, first array-oriented languages, to analyse Mars Mariner data
- 1977: RSI founded by David Stern sells first IDL version (VAX/VMS) to NASA
- 1987: Re-written in C for UNIX release
- 1992-94: available on Linux, Windows and Mac
- 2004: RSI morphs into ITT

Who uses IDL?

- Primarily, astronomers and space scientists

Best Features

- Scientific community contribution
- Optimised array operations
- Large suite of versatile maths, data analysis and visualisation routines
- Interactive
- High quality graphics output
- Access to (almost) all code
- Fast, i.e. not slow
- Data structures
- Dynamic variable typing

How does it compare?

http://www.ast.cam.ac.uk/~vasily/idl.htm

- IRAF, MIDAS and IDL are used for astronomical data analysis. IDL is not a replacement for IRAF, but it is easier to read and customise.
- FORTRAN, C and IDL are similar programming languages. IDL is slower, but also provides an environment for scientific data analysis.
- MATHEMATICA, MAPLE, MATLAB and IDL can do maths. IDL can not solve symbolically, but has more powerful graphics and knows astronomy.
- SUPERMONGO, PGPLOT and IDL produce quality graphics. IDL not only plots data, but also computes and analyses it - it's a programming language.

IDL One Liners

- Coordinate System Transformations, Galactic Extinction, Spectral Line Shapes, k-Corrections, Distances in Cosmology

Community Coding

- Not precompiled, highly portable code
- No installation required
- No centralised distribution, the libraries are published online by the scientists (Chandra, SDSS, HST, Los Alamos and many other universities and data centres)

Download IDL libraries and keep them organised, make sure they're added to your !PATH
http://www.ast.cam.ac.uk/~vasily/idl.htm

IDL Session

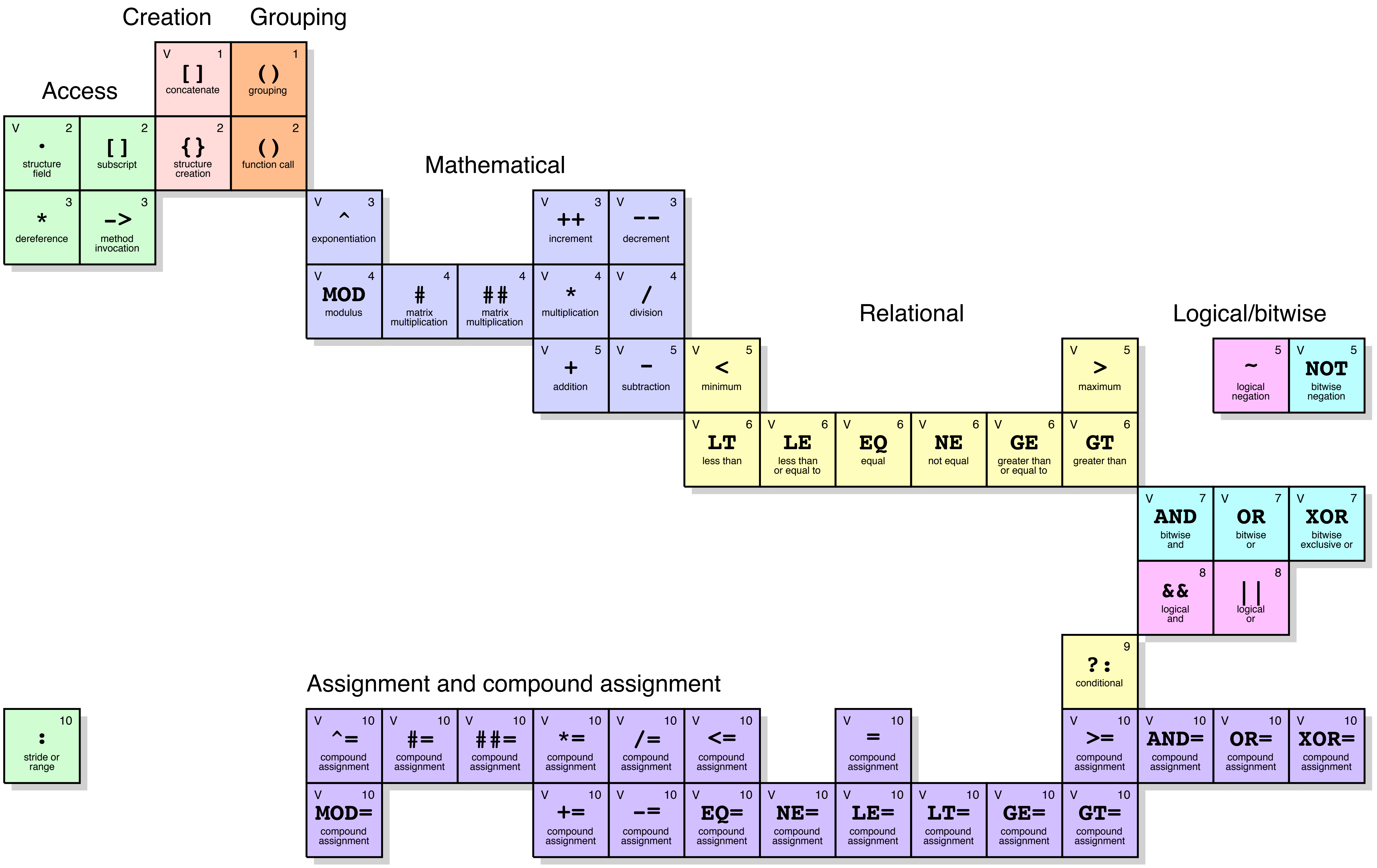
- idl command line
- use command recall
- variables created on the fly
- create synthetic data
- **HELP**
- online help
- **SAVE** session
- compose scripts
- create procedure and functions
- **emacs IDLWAVE**: fast editing, compiling, completion

How To Learn IDL

- IDL Help file
- David Fanning's Website
- IDLWAVE Google Group
- Various Tutorials online

Periodic Table of IDL Operators

Michael Galloy, 2006 - michaelgalloy.com



Creation

V	1	[ ]	concatenate
	2	{ }	structure creation

Grouping

	1	( )	grouping
	2	( )	function call

Access

V	2	.	structure field
V	2	[ ]	subscript
V	3	*	dereference
V	3	->	method invocation

Mathematical

V	3	^	exponentiation
V	3	++	increment
V	3	--	decrement
V	4	MOD	modulus
V	4	#	matrix multiplication
V	4	##	matrix multiplication
V	4	*	multiplication
V	4	/	division
V	5	+	addition
V	5	-	subtraction

Relational

V	5	<	minimum
V	5	>	maximum
V	6	LT	less than
V	6	LE	less than or equal to
V	6	EQ	equal
V	6	NE	not equal
V	6	GE	greater than or equal to
V	6	GT	greater than
	9	?:	conditional

Logical/bitwise

	5	~	logical negation
V	5	NOT	bitwise negation
V	7	AND	bitwise and
V	7	OR	bitwise or
V	7	XOR	bitwise exclusive or
	8	&&	logical and
	8	||	logical or

Assignment and compound assignment

V	10	^=	compound assignment
V	10	#=	compound assignment
V	10	##=	compound assignment
V	10	*=	compound assignment
V	10	/=	compound assignment
V	10	<=	compound assignment
V	10	=	compound assignment
V	10	>=	compound assignment
V	10	AND=	compound assignment
V	10	OR=	compound assignment
V	10	XOR=	compound assignment
V	10	MOD=	compound assignment
V	10	+=	compound assignment
V	10	-=	compound assignment
V	10	EQ=	compound assignment
V	10	NE=	compound assignment
V	10	LE=	compound assignment
V	10	LT=	compound assignment
V	10	GE=	compound assignment
V	10	GT=	compound assignment

	10	:	stride or range

Data Input

ASCII number array	Mixed type ASCII	FITS

INTO ARRAY
`data = fltarr(nx, ny)`
`openr, lun, path, /get_lun`
`readf, lun, data`
`free_lun, lun`
Convenient
	INTO VECTORS
	`readcol, path, v1, v2, v3, format=fmt`
	Fast
		INTO STRUCTURE
		`data = mrdfits(path, ext)`
		Fast Convenient

INTO VECTORS
`readcol, path, v1, v2, v3, format=fmt`
Fast
	INTO STRUCTURE Convenient
	`template = ascii_template(path)`
	`data = read_ascii(path, template=template)`	

INTO STRUCTURE
`template = ascii_template(path)`
`data = read_ascii(path, template=template)`		

Data Output

- For you: IDL SAVE
- For your collaborators: IDL SAVE, FITS
- For the world: FITS, ASCII

2 Most Useful IDL Commands

- WHERE
- HISTOGRAM

X Window Graphics Gotchas

- Undecomposed colour
- Backing store not set

Other Common IDL Gotchas

http://www.dfanning.com/code_tips/mostcommon.html

- Integer data type operations
- “Hidden” dimensions
- WHERE count
- Keywords in plotting routines
- Calling function as a procedure and vice versa

IDL Philosophy

- Do not re-invent the wheel, in 99% cases it has been coded already.
- Medium-weight projects, preferably ~100 lines.
- Medium size dataset, up to 1Gb (loaded into RAM)
- Avoid FOR loops if you can

Keep It Organised

Quick data lookup

- Command line: use IDLWAVE for more recall and completion. Use JOURNAL to keep the log of the session.

Recurrent Tasks

- Scripts: NO need for PRO statement, it's just a bunch of commands in a file! Use GOTO, CASE and IF.

Big Project

- Procedures and Functions. Naming is important. Pass data in structures. Avoid COMMON blocks.