

# IDL DeCal Week 2

Facilitator: Paul Higgins ([phiggins@ssl.berkeley.edu](mailto:phiggins@ssl.berkeley.edu))

Faculty Sponsor: Carl Heiles

## Vectors and Arrays

Dimensions

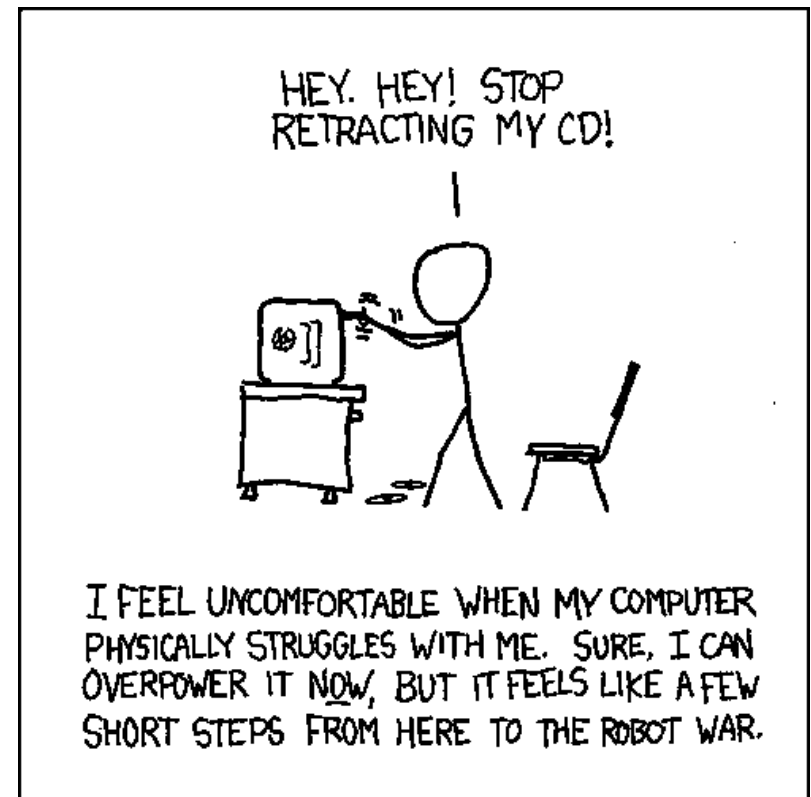
Vector Syntax

Subscripting

Array Functions

Website:

[http://ugastro.berkeley.edu/~phiggins/idl\\_decal/index.html](http://ugastro.berkeley.edu/~phiggins/idl_decal/index.html)



# Dimensions

**indices: 0 1 2 3 4 5**

vector = [1, 2, 3, 4, 5, 6]

→ [ 2 2 9 5 3 1 ] **1 Dim**

array = [ [3,4,5], [6,7,8], [9,10,11] ]

→  $\begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \end{bmatrix}$  **2 Dim**

Arrays can have **8** dimensions maximum!

Concatenation:

vect1 = [1, 7, 3, 9]      vect2 = [0, 2, 8, 5]

vect3 = **[vect1, vect2]**      → [ 1 7 3 9 0 2 8 5 ] **horizontal**

array = **[ [vect1], [vect2] ]**      →  $\begin{bmatrix} 1 & 7 & 3 & 9 \\ 0 & 2 & 8 & 5 \end{bmatrix}$  **vertical**

# Generating Arrays

Byte*:	<b>BYTARR()</b>
Integer:	<b>INTARR()</b>
Long:	<b>LONG()</b>
<b>Floating Point**:</b>	<b>FLTARR()</b>
Double-precision:	<b>DBLARR()</b>
Complex:	<b>COMPLEXARR()</b>
String*:	<b>STRARR()</b>

# Subscripting

## 1D

arr = **findgen**(100)

newarr = arr[**a:b**] → **range** subscripting- new array is a **subset** of the old

w = **where**(arr **gt** 4) → find **elements that fulfill** the “where” argument. (WHERE returns INDICES)

ones = arr[w] → use **where** result to retrieve elements

## 2D

arr2d = **findgen**(10, 10) → generate a 2D array

element = arr2d[**4, 6**] → subscript 1<sup>st</sup> dim, 2<sup>nd</sup> dim, 3<sup>rd</sup> dim etc..

subset = arr2d[**2:5, 3:6**] → subscript ranges in each dimension

# More on Using Where

Where returns **vector of subscripts**

Represent circled elements:

`where_result = where(arr lt 6)`

`where_result = [0,1,2]`

`row1 = arr[[0,1,2]]`

`elem1 = arr[5]`

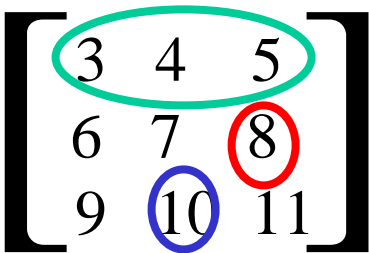
`elem2 = arr[7]`

Computer stores all arrays as a single string of elements

Subscript any element in any dimension array with **one number**

**OR**, subscript using a **comma** to separate dimensions

ex: `element = arr[column, row]`

arr = 

# Array Functions and Syntax

sum = **total**(arr) → **sum** the elements of an array

avg = **mean**(arr) → **average** an array

arrsz = **size**(arr) → return vector with dimensional info

bigarr = **rebin**(arr, 2\*arrsz[1], 3\*arrsz[2], /sample) → **resize** an array

arr2x3 = **reform**(arr, 2, 3) → **change dim** of arr but not elements

stretcharr = **congrid**(arr, 325, 641) → **resize** arr to any size (interp)

ordered\_arr = **sort**(arr) → **order** the array's elements by value

unique\_elem = **uniq**(arr) → after sorting, return only the **unique** values

rev\_arr = **reverse**(arr) → **reverse** the order of the array's elements

shifted\_arr = **shift**(arr, shiftx, shifty) → shifts array up, down, left, or right

# Array Math

```
arr=findgen(100,400)
```

```
wresult=where(arr eq 1)
```

```
xind = wresult mod xdim
```

```
yind = wresult/fix(ydim)
```

```
element=arr[xind, yind]
```

Multiplication: \*, #, ##

Addition:      sumarr = arr1 + arr2

hiarr = arr1 + 10

Subtraction:    diffarr = arr2 - arr1

loarr = arr2 - 30

Division: /

# Array Manipulation Vs Loops

IDL is Array Based – IDL is OPTIMIZED for arrays.

Array functions are essentially loop shortcuts. They run **faster**.

**Use Array Functions whenever possible!!**

**Compare:**

```
IDL> x = findgen(1000000)
```

```
IDL> sum = total(x)
```

```
IDL> print, sum
```

**WITH**

```
IDL> sum = 0
```

```
IDL> for i = 0, n_elements(x)-1 do sum=sum+x[i]
```

```
IDL> print, sum
```

**Which one runs faster?? (runs at all)?**

**TRY IT OUT!**

# Plotting

```
plot, xarr, yarr, title= ‘’, xtitle=‘’, ytitle=‘’, thick=3 , $  
    xstyle=1 , ystyle=1 , xthick=2 , ythick=2 , $  
    charsize=1.4 , charthick=2 , color=!red, psym=-4 , $  
    linestyle=2, /noerase
```

## Example:

```
IDL> xarr=findgen(100)
```

```
IDL> yarr=(xarr+10.)^2.
```

```
IDL> window, 8, xsize=800, ysize=400
```

```
IDL> wset, 8
```

```
IDL> wshow, 8
```

```
IDL> plot, xarr, yarr, ytitle=‘Y’, xtitle=‘X’, $
```

```
IDL> title=‘Y(X) = ‘+textoidl(‘X^2’)
```

# Histogram

**xarr** and **yarr** from prev. examp.

## Example:

```
histy = histogram(yarr, bin=bin, min=min, max=max)
```

```
minmaxy = minmax(yarr) & rangey = abs(minmaxy[1] - minmaxy[0])
```

```
histx = findgen(n_elements(histy))*(rangey) + minmaxy[0]
```

```
plot, histx, histy, ps=10, title='This is a Histogram!'
```

# Environment Variables

Defined in file: `~/.idlenv`

(Paths for finding and compiling procedures and functions, etc.)

**De-referencing** an env. var.: `IDL> print, !path`

Bang, or `!` is used to call env. var.'s

**Setting** an env. var.: `IDL> !p.multi=0`

Other Environment Variables:

**Plotting:** `!p.multi, !p.thick, !p.charsize, !p.title, !p.font, !p.color,`  
`!p.psym, !p.linestyle`

**X-Axis and Y-Axis:** `!x.thick, !x.title, !x.style, !y.thick, !y.title, !y.style`

**Also, TRY:**

`help, !p, /str`

Also try this with: `!version, !x, !y, !prompt`