# Basic IDL Tutorial

Neus Agueda

August 3, 2016

IDL is a data analysis and visualization package that is widely used in the solar and astronomical research community. Becoming familiar with the basics of this programming language will provide you the foundation necessary to gain expertise in *any* programming language.

Here is a short IDL tutorial in 20 easy steps:

1. To begin an IDL session type `idl` on the command line. To quit, type `exit`

2. To search for and learn about built-in functions and procedures type e.g.
   `IDL> doc_library, 'open'`,
   or check the online searchable help website.

3. To create an array use the functions `strarr`, `intarr`, `lonarr`, `fltarr` or `dblarr` to obtain a string, integer, longword integer ($> 32000$), single-precision float, or double-precision float array of the specified dimensions. For example:

   `a = fltarr(5,4)`, `a` is 5-element by 4-element float array
   `b = intarr(100)`, `b` is a 100-element integer array
   `c = intarr(250,400)`, `c` is a 250-element by 400-element integer array

4. To create new variables
   `a = 5.3` or `b = [6,4,3]` (no need to declare them first)

   If you previously created the variable, you can access its content or modify it, e.g. `b(2) = 5` to change the third element of `b` to 5. Note that indices in IDL start with 0.

   Other examples:
   `IDL> c = fltarr(5)`
   `IDL> c(2:4) = 1.3` would leave `c = [0,0,1.3,1.3,1.3]`
   `IDL> c(2:*) = 5.` would assign the value 5. to all the elements of `c` from index 2 to the last index of the array.

5. To change a variable type you can use the following functions: `double`, `fix`, `long`, `string`, `float`, which change the array type into double-precision floats, integers, long integers, strings or floats.

6. To check a variable type use `help`
   `IDL> help, a`
   provides the type of variable `a`

7. To create an array of consecutive numbers use e.g. `indgen` and `findgen`. For example

   `IDL> a = indgen(5)` creates a 5-element integer array with the values 0,1,2,3,4.
   `IDL> b = 4*findgen(5)+3` is `b = [3.,7.,11.,15.,19.]`

8. To operate use `+` (addition), `-` (subtraction), `*` (multiplication), `/` (division), `^` (exponentiation). Examples:

   `IDL> c = a/5.`, where `c` is an array with the same dimensions as `a` where each element is divided by 5. (`c` is a float)

   `IDL> c = a*b`, where both `a` and `b` have the same dimensions. `c` has the same dimensions and its elements are the product of each of the elements of `a` and `b`

9. To sum all the elements of an array use `total`. The sum of the array elements over a given dimension is returned if the dimension argument is present.
   `IDL> result = total(array [, dimension])`

   For example, if `a` is a 400-element by 200-element float array
   `b = total(a)` is the sum of all the elements in `a`
   `c = total(a,1)` is a 200-element array, each one contains the sum of the corresponding 400 elements
   `d = total(a,2)` is a 400-element array, each one contains the sum of the corresponding 200 elements
   Another example, if `f` is an array with dimensions `n1`, `n2`, and `n3`
   `IDL> g = total(f,3)` is an array with dimensions `n1`, `n2`
   `IDL> h = total(f,1)` is an array with dimensions `n2`, `n3`

10. To locate the maximum or minimum value (and its index) of an array use `max` or `min`

```
IDL> a = [0.,3.5,7.6,8.9,10.8]
IDL> maxvalue = max(a,imax)
IDL> minvalue = min(a,imin)
IDL> print,maxvalue,imax
10.8 4
IDL> print,minvalue,imin
0.  0
```

11. To determine which element(s) of a vector satisfy a particular condition use `where`

```
IDL> a = [0.,3.5,7.6,8.9,10.8]
IDL> gtzero = where(a gt 0.)
IDL> onset = min(gtzero)
IDL> print,'The first non-zero value is at index',onset
The first non-zero value is at index 1
```

12. For most calculations we do not use IDL interactively. We place the IDL commands in a program because it is then easier to ask IDL to repeat the same calculations and correct mistakes (i.e. debug your code). You can use any editor to create your program. The extension of your program should be `.pro`, e.g. *myprogram.pro*

    You can run your program by typing
    `IDL> .r myprogram`

13. To create a loop and a conditional inside a program

    ```
    FOR variable = init, limit [, increment] DO BEGIN
    statements
    ENDFOR
    ```
    For example
    ```
    for i = 0,50 do begin
    a(i,*) = b(i,*)/c(i)
    endfor
    ```

    ```
    IF expression THEN statement
    ```
    or
    ```
    IF expression THEN BEGIN
    statements
    ENDIF [ ELSE BEGIN
    statements
    ENDELSE ]
    ```

    For example `if (a ne 2) then a = 2`
    ```
    if (a gt 1) and (b lt 3.)  then c=b/a
    if (a ge 1) and (b le 3.)  then c=b/a
    ```

14. To open an existing file for input or output (access/store data)
    To open a file to read it: `openr,unit,filename`
    For example `openr,1,'/scratch/solar/file.dat'`

    To open a file to write in it: `openw,unit,filename`
    For example `openw,1,'/scratch/solar/file.dat'`

    To close a file: `close,unit` or `close,/all`

15. To read and write a file

    To read unformatted binary data from a file into IDL variables: `readu,unit,variable(s)`. For example `readu,1,t1,t2`, where `t1` and `t2` should have the correct dimensions.

    To read formatted data from a file into IDL variables: `readf,unit,variable`, where `t1` and `t2` should have the correct dimensions.

    To write unformatted binary data from an expression into a file: `writeu,unit,variable(s)`. For example `writeu,1,anything` (it would be read with `readu`)

    To write formatted data into a file: `printf,unit,variable`
    For example `printf,1,time,intensity,format='((f8.3),1x,4(f7.4,2x))'`

    To write the data in the standard output stream (terminal): `prinf,variable`

16. To plot on the X window

    `plot,y` plots a 1-dimensional array
    `plot,x,y` plots `y` vs. `x`
    `plot,u,v,xrange=[0.,2.],xs=1,yrange=[1.,7.],ys=1` sets the range of the x- and y- axis
    `plot,x,y,/ylog` to get a logarithmic y-axis
    `plot,x,y,xtitle='X Axis',ytitle='Y Axis'`
    `plot,x,y,line=1` (line style 0=solid, 1=dotted, 2=dashed etc)
    `plot,x,y,psym=1` (symbol style 1=plus sign, 2=asterisk, 3=period, 4=diamond, 5=triangle etc)

    `oplot,x1,y1` plots on top of the previous plot (here you can not fix the axis range or title any more)

    `xyouts,0.1,0.5,'Legend',/normal` draws the text `Legend` on the plot. The keyword `/normal` indicates that the positioning coordinates supplied are specified in the normalized coordinate system, and range from 0 to 1. The default coordinate system is `/data`.

17. To add colors in your plot `loadct, palette_number` You can see different color palettes online. Also `tek_color` loads a great color palette, where 0=black, 1= white, 2=red, 3=green, 4=blue etc.

18. To group several plots on a page use `!p.multi`. Set !P.MULTI equal to an integer vector in which: The second element of the vector contains the number of columns. The third element contains the number of plots rows. For example, to set up two plots vertically on each page, use the following statement:
    `!p.multi = [0, 1, 2]`
    The first element, `!p.multi(0)`, is set to zero to cause the next plot to begin a new page. To make four plots per page with two columns and two rows, use the following

statement: `!p.multi = [0, 2, 2]` To reset to the default of one plot per page, set the value of !P.MULTI to 0: `!p.multi = 0`

19. To save a plot in a postscript file instead of to an X window.

```
set_plot,'ps'
device,filename='myplot.ps',/color
tek_color
plot,time,intensities,psym=4,xtitle='Universal Time [h]',ytitle='Intensities
[p/(s sr cm!u2!n MeV)]',title='Particle Event at 1 AU'
oplot,time,intensities,color=2
device,/close
```

`set_plot,'x'` (sets the plotting area back to the X window)

You can usually view postscript files on a Linux machine by using the command "evince" or "gv"

```
> evince myplot.ps
> gv myplot.ps
```

20. Take it easy and write your code in pieces. If it doesn't work, read the error message (it usually points to the line where the code stopped). Don't be afraid to ask other colleagues including the wise Google. Good luck!