get_par

Synopsis

Module function for getting model parameter values, etc.

Syntax

```
Array_Type get_par([String_Type])
```

Error Return Values: NULL

Argument:

1. Model name, or model parameter name (default all model parameters)

Description

This function retrieves an array of structures, each of which contains information about a defined model parameter. A typical structure looks as follows:

```
sherpa> GAUSS[g]
sherpa> foo = get_par("g")
sherpa> print(foo[0])
```

```
name = g.fwhm
model = gauss1d
type = src
value = 10
min = 2.22507e-308
max = 1.79769e+308
delta = -1
units = NULL
frozen = 0
linked = 0
linkexpr = NULL
```

The fields of the structure are:

get_par Structure Fields

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>the name of the parameter</td>
</tr>
<tr>
<td>model</td>
<td>the model that the parameter belongs to</td>
</tr>
<tr>
<td>type</td>
<td>source/background–type (src) or instrument–type (inst) model</td>
</tr>
</tbody>
</table>
value the parameter value (either a number, or a string filename)
min the current (soft) lower bound on allowed parameter values
max the current (soft) upper bound on allowed parameter values
delta the initial step size for the parameter in fits (or −1 to use the default step size)
units parameter units, if known/appropriate
frozen if 1, the parameter value is frozen; if 0, it is thawed
link if 1, the parameter's value is linked to that of other parameter(s); if 0, it is not linked
linexpr if the parameter is linked to other parameters, the expression showing how it is linked

See the Sherpa command CREATE for more information.

Example 1
Get a parameter structure; change two fields; set back into Sherpa:

sherpa> GAUSS[g]
sherpa> foo = get_par("g.pos")
sherpa> print(foo)
name = g.pos
model = gauss1d
type = src
value = 0
min = -3.40282e+38
max = 3.40282e+38
delta = -1
units = NULL
frozen = 0
linked = 0
linexpr = NULL
sherpa> foo.value = 15.5
sherpa> foo.min = 0.0
sherpa> () = set_par(foo)
sherpa> SHOW g

gauss1d[g] (integrate: on)
<table>
<thead>
<tr>
<th>Param</th>
<th>Type</th>
<th>Value</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>fwhm</td>
<td>thawed</td>
<td>10</td>
<td>1.1755e-38</td>
<td>3.4028e+38</td>
<td></td>
</tr>
<tr>
<td>pos</td>
<td>thawed</td>
<td>15.5</td>
<td>0</td>
<td>3.4028e+38</td>
<td></td>
</tr>
<tr>
<td>ampl</td>
<td>thawed</td>
<td>-3.403e+38</td>
<td>3.4028e+38</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 2
Here we loop through all the defined parameters and display their name and current value:

sherpa> erase all
sherpa> paramprompt off
sherpa> xsmekal[gal]
sherpa> xsphabs[abs]
sherpa> ps = get_par()
sherpa> ps
Struct_Type[7]
sherpa> foreach(ps){p=();vmessage("Par %-14s = %g",p.name,p.value);}  
Par gal.kT = 1
Par gal.nH = 1
Par gal.Abund = 1
Par gal.Redshift = 0
Par gal.Switch = 1
Par gal.norm = 1
Par abs.nH = 0.1
The initial set of lines are to set up two models ("gal", which is an XSMEKAL model, and "abs", which is an XSHPABS model) with default parameter values. The ps variable is set here to an array of 7 structures, which we loop through using the S−Lang foreach function. Each member of the array is looped through, and stored in the variable p. This variable is then used to get the parameters name and value (p.name and p.value respectively). The whole 'foreach' statement must be on one line since Sherpa does not allow multi−line S−Lang statements. This restriction does not apply to code in a file executed via evalfile().

**Example 3**

Here we use a small S−Lang function we have written to display parameter values. If the file print_pars.sl contains:

```plaintext
define print_pars() {
    variable pars = get_par();
    if ( NULL == pars ) {
        print( "No parameter values found!" );
        return;
    }
    vmessage( "# Name model frozen value" );
    foreach ( pars ) {
        variable par = ();
        vmessage( "%−20s %−10s %d %g", 
            par.name, par.model, par.frozen, par.value 
        );
    }
}
```

then you can say:

```
sherpa> () = evalfile("print_pars.sl")
sherpa> erase all
sherpa> paramprompt off
sherpa> xsmekal[gal]
sherpa> xsphabs[abs]
sherpa> print_pars
```

<table>
<thead>
<tr>
<th># Name</th>
<th>model</th>
<th>frozen</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gal.kT</td>
<td>xsmekal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>gal.nH</td>
<td>xsmekal</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gal.Abund</td>
<td>xsmekal</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gal.Redshift</td>
<td>xsmekal</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>gal.Switch</td>
<td>xsmekal</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gal.norm</td>
<td>xsmekal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>abs.nH</td>
<td>xsphabs</td>
<td>0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

although you may find either list_par() or SHOW more useful.

**Bugs**

See the Sherpa bug pages online for an up−to−date listing of known bugs.

**See Also**

sherpa

autoest, background, create, create_model, createparamset, fit, freeze, get defined models,
get model params, get models, get num par, get stackexpr, getx, gety, guess, instrument, integrate,
is paramset, jointmode, kernel, lineid, linkparam, mdl, modelexpr, modelstack, nestedmodel, noise,
paramprompt, paramset, pileup, rename, run fit, set par, set paramset, set stackexpr, source, thaw,
truncate, unlink
Example 3