Working with masks

Example 1 – getting and applying a mask

```
# Let us open a data file that contains surface type
# (land fraction) data and extract data
>>> import cdms, MV
>>> f_surface = cdms.open('sftlf_ta.nc')
>>> surf = f_surface('sftlf')
# Designate land where "surf" has values
# not equal to 100
>>> land_only = MV.masked_not_equal(surf, 100.)
>>> land_mask = MV.getmask(land_only)
# Now extract a variable from another file
>>> f = cdms.open('ta_1994-1998.nc')
>>> ta = f('ta')
# Apply this mask to retain only land values.
>>> ta_land = cdms.createVariable(ta, mask=land_mask,
copy=0, id='ta_land')
```

Example 2 – creating MVs and masking

```
>>> a = MV.array([1,2,3]) # Create array a, with no mask
>>> b = MV.array([4,5,6]) # Same for b
>>> a+b
variable_13
array([5,7,9,])
>>> a[1]=MV.masked # Mask the second value of a
>>> a.mask() # View the mask
[0,1,0,]
>>> a+b # The sum is masked also
variable_14
array(
data = [5,0,9,],
mask = [0,1,0,],
fill_value=[0,]
)
```

MA-specific functions (1)

`MA._get_print_limits()/MA._set_print_limits() #` prints the number of values included 300 default

`MA.getmask(array)` # Return mask associated with array (None if no mask)

`MA.masked_greater(array,array2 or value),` … # Mask array where array is greater/less… than array2 or value passed

`MA.masked_inside/masked_outside(array,a1/v1,a2/v2)` # creates an array with values inside/outside the closed interval [v1, v2] masked. v1 and v2 may be in either order.

MA-specific functions (2)

`MA.masked_where(condition, data, copy=1)`

Creates a masked array whose shape is that of condition, whose values are those of data, and which is masked where elements of condition are true. Condition can be something like “`MA.greater(data, value)`”.

```
MA.masked_equal/masked_values(data, value,
   rtol=1.0000000000000001e-05, atol=1e-08, copy=1,
   savespace=0)
```

Creates a masked array where value=value; mask is None if possible. If copy==0, and otherwise possible, result may share data values with original array.

Let d = filled(data, value). Returns d masked where:

abs(data-value)<=atol + rtol * abs(value)

`MA.mask_or(a1,a2)`

Create a mask using a1 values or a2 values (if a1 value is None). Use None if they are both None.