Homeyer schrieb:
> On Apr 8, 6:02 pm, Homeyer <cam.the.weather....@gmail.com> wrote:
>> On Apr 8, 8:23 am, David Fanning <n...@dfanning.com> wrote:
>>
>>
>>
>>> Folks,
>>> I don't have time today to think about this, so I've
>>> decided to get you to do my thinking for me. :-)
>>> Suppose you are expecting a data array and you suspect
>>> the data has "missing values" in it that you wish to
>>> exclude from further processing. You write a keyword
>>> for your routine that allows the user to pass in what
>>> he is using for the "missing value".
>>>   PRO Junker, data, MISSING_VALUE=missing_value
>>> How would you write the code to assure that this missing
>>> value would be excluded from further processing?
>>> You should assume:
>>>   1. The data can be any data type except complex or string.
>>>   2. The missing value *could* be !VALUES.F_NAN.
>>>   3. Unsophisticated users might be using your program,
>>>   so, for example, they might pass in a missing value
>>>   such as 594.32.
>>> No, this is NOT my homework! But I do need it ASAP. ;-)
>>> Cheers,
>>> David
>>> --
>>> David Fanning, Ph.D.
>>> Fanning Software Consulting, Inc.
>>> Coyote's Guide to IDL Programming:http://www.dfanning.com/
>>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
>> The following should suffice:
>>
>> IF (missing_value GT 0) THEN BEGIN
>>   type = SIZE(array, /TNAME)
>>
>>   CASE type OF
>>   'COMPLEX' : MESSAGE, 'Complex values not allowed for removing
>> MISSING data.'
>>   'STRING' : MESSAGE, 'String values not allowed for removing
>> MISSING data.'
>>   ELSE   : invalid = WHERE((array EQ missing_value), iv_count,
>> COMPLEMENT = valid, NCOMPLEMENT = v_count)
>>   ENDCASE

```
>>
>>    IF (v_count GT 0) THEN array = array[valid]
>> ENDIF
>>
>> That is, if you dont care about overwriting the original data or it is
>> not gridded. If it is gridded (data points with missing values should
>> remain), then you could replace those with NaNs in the final array
>> using the indices above.
>>
>> Cheers,
>> Cameron Homeyer
>
> oops, missed the NaN part:
>
> This would go before the last IF (v_count...
>
> IF (~FINITE(missing)) THEN invalid = WHERE(~FINITE(array), iv_count,
> COMPLEMENT = valid, NCOMPLEMENT = v_count)
>
```

What is the meaning of ~ in front of the function name?