r_index(v, ...)

Arguments

v	in	a record array from which to extract elements Allowed:record (array)
...	in	slicing dimensions Allowed:integers, an index record, integer vectors

Returns
integer array

Description
r_index is a helper function for the record array functions. It returns an integer array with the selected elements.

Example

```
- print a := r_array(1:6,3,2) # create a record array
[*29=1, *30=2, *31=3, *32=4, *33=5, *34=6]
- r_index(a) # its index
[[1:3,]
    1 4
    2 5
    3 6]
- r_index(a):: # its shape
[shape=[3 2] ]
- r_index(a,2,)
[2 5]
- r_index(a,2,)::
[shape=2]
- r_index(a,2,2)
5
- r_index(a,2,2)::
[=]
```

1.6.6 quanta - Tool

Package general
Module quanta

quanta tool

Constructors

quanta	Construct quanta tool

Functions

abs	absolute value of quantity
acos	arccosine of quantity
add	add quantities
angle	show an angle as a formatted string
asin	arcsine of quantity
atan	arctangent of quantity
atan2	arctangent of two quantity
canon	get canonical value of quantity
canonical	get canonical value of quantity
ceil	ceil value of quantity
check	check for proper unit string
checkfreq	check for proper frequency unit
compare	compare dimensionality of units
constants	get a constant
convert	convert a quantity to another unit
convertdop	convert a doppler velocity quantity to another unit
convertfreq	convert a frequency quantity to another unit
cos	cosine of quantity
define	define a new unit name
div	divides quantities
done	free resources used by tool
eq	compare quantities
exp	exponential of quantity
fits	define some FITS units
floor	floor value of quantity
form.xxx	Format a quantity using globaly set formats
ge	compare quantities
getformat	get current output format
getunit	get the internal unit of a quantity
getvalue	get the internal value of a quantity
gt	compare quantities
gui	Use the quanta GUI interface
id	identification of tool
is_angle	check if valid angle or time quantity
le	compare quantities
log	logarithm of quantity
log10	logarithm of quantity
lt	compare quantities

map	list known unit names and constants
mul	multiply quantities
ne	compare quantities
neg	negate quantities
norm	normalise angle
pow	raise quantity to power
quantity	make a quantity from a string or from a value and a unit string
setformat	set format for output of numbers
sin	sine of quantity
splitdate	split a date/time into a record
sqrt	square root of quantity
sub	subtract quantities
tan	tangent of quantity
time	show a time (or date) as a formatted string
toangle	convert a time (or an angle) to an angle
tos	convert quantity to string
totime	convert an angle (or a time) to a time
type	type of tool

quanta - Constructor

Package	**general**
Module	**quanta**
Tool	**quanta**

Construct quanta tool

Synopsis
quanta(host, forceneweserver)

Arguments

host	in	host on which to run tool Allowed:valid host name string Default: ”
forceneweserver	in	force the use of a new server Allowed:bool Default: F

Returns
tool

Description
Create a quanta `tool`on the specified host (or by default the host you are running on).

---

quanta.gui - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

Use the quanta GUI interface

Synopsis
gui()

Arguments

Returns
Bool

Description

Start a GUI interface for the handling of units and quanta.
A menu bar will appear with 3 sets of menu headings.
The File menu contains options to close the current window (start again with dq.gui()).
The last menu is the Help menu, which can drive to different parts of the Reference manual. The '?' menu selects how popup help can be displayed (initially hover for short-help, MB3 down for longer help).
A way of starting automatically a GUI interface is by adding the following to your .aipsrc file

```
quanta.default: gui
```

Note that all actions available on the screen, are also available as tool functions for the command line interface. They are each described separately.
The screen has a top area, and three other areas. The *Map* menu in the top area allows you to check which units are recognised, and which constants are available.

The quantity calculator works as an *RPN* calculator. Values are typed in the *Value* input field, the top of the stack. Values can have either the units attached (e.g. 5km/s), or detached (e.g. 5 mm.s/A), or be of the recognised time/angle possibilities (e.g. today, 12:15:2.2, 12.3.5.7 12d3m5.7, 1996/12/2/12h15m). The top of the stack is always shown in canonical units in the window below the action buttons (the right side of the window) Some simple actions are available on the bar below the entry window:

$\wedge$ pops stack to enter window (e.g. for editing)

$\vee$ pushes top of stack down onto stack (like Return)

$x <> y$ interchanges top two values on stack

hand allow edit of top of stack

dup duplicates top of stack

copy copy top of stack to clipboard

paste paste clipboard to top of stack

Note that the internal precision of the numbers is not reflected in the output precision.
The bottom entry is used to enter units used in the *convert* and *define* actions. The Operators (right-hand) window works on the top, or top two, values of the stack. Most actions will be clear to the user. The last action is displayed in the action window. The actions in the left column and bottom row are:

pop pops top of stack to enter window

clear clears the whole stack

convert converts the top of stack value to the units entered in the Units window (do not forget to hit Return). Note that with the possibilities of entering angles as 5d11m, there is a difficulty having days as a unit. This is solved by having a decimal point in the number. I.e. 5d will be interpreted as 5 degrees, 5.d as 5 days.

canon convert top of stack to canonical units

define define the top of stack as your own unit in the Units field (e.g. define the value;1Jy as your unit:JY)

const put the selected constant on top of stack

FITS define some FITS units for use (check them with Map:User)

The functions available to the non-GUI user are the same as to the GUI user. They are, a.o.: *quantity, canonical, convert, define, map, fits, angle, time, add, sub, mul, div, neg, norm, sin, cos, tan, asin, acos, atan, atan2, abs, ceil, floor,*

compare, check, pow, constants, totime, toangle, tos (see list at top of tool description).
If at any stage you want to use a GUI result on the command line, the 'copy' button of the GUI will transfer the last result to the clipboard. Most of the fields in the GUI have individual popup help, which can be viewed hovering on the field.
Trigonometric functions expect either angle units, or no units. In the latter case the selected units in the bottom of the right-hand side are used.

---

quanta.setformat - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

set format for output of numbers

Synopsis
setformat(t, v)

Arguments

t	in	type Allowed: coded string indicating which format parameter to set Default: ”
v	in	format parameter value Allowed: numeric or coded string, depending on format type to be set Default: F

Returns
bool

Description
setformat allows to define output format parameters for a variety of value types. The *type* is indicated by the first parameter, and coded (see following list). The actual format parameter set depends on the value parameter *v* given. If this value is specified as the '...' string, the value will be asked interactively from the user. The formats specified (or the defaults) are used in the form functions. The formatting that can be specified:

prec set the output precision for general numbers. If the value given is an integer less than one, the precision will be set to 6.

aprec set the output prescion for angles. If the value given is an integer less than one, the precision will be set to 6. A value of 6 will output up to the seconds field, with decimal places if larger than 6.

tprec set the output prescion for times. If the value given is an integer less than one, the precision will be set to 6. A value of 6 will output up to the seconds field, with decimal places if larger than 6.

long longitude display. Valid formats: 'hms', 'dms', 'deg', 'rad', 'd', '+deg'.

lat latitude display. Valid formats: 'hms', 'dms', 'deg', 'rad', 'd'.

len length display. Valid format: any length dimension

dtime time type display: 'local', 'utc', 'last' or 'solar'

elev elevation limit rise/set: angle quantity

auto auto frame update interval: time quantity

vel velocity: velocity unit

freq wave characteristics: frequency, length, time, angle/time, /length, energy, impulse units

dop doppler type display: 'radio', 'opt', 'true'

unit default units: any valid unit

Example

```
- 1.234567890123456 # note that quanta set precision to 9
1.23456789 # (as compared to default Glish 6)
- a:=dq.quantity(1.2345678901234567, 'km/s')
- dq.form.vel(a)                # display as velocity in default units
1.23456789 km/s
- dq.setformat('vel', 'yd/a')   # try another unit
T
- dq.form.vel(a)
4.26071737e+10 yd/a
- dq.setformat('prec',12)       # or another precision
T
- dq.form.vel(a)
42607173719.8 yd/a
- dq.setformat('unit', 'm/s')   # set generic units
```

```
T
- dq.form.unit(a)              # and see what it looks like
1234.56789012 m/s
- b:=dq.quantity('5d2m3')      # an angle
- dq.form.long(b)              # as longitude gives it in hms
00:20:08.200
- dq.setformat('long', 'deg')  # display longitude in degrees
                               # (rather than default hms)
T
- dq.form.long(b)              # and show it
5.03416667 deg
- dq.form.lat(b)               # as a latitude it is default dms
+005.02.03.000
-
```

---

quanta.getformat - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

get current output format

Synopsis
getformat(t)

Arguments

t	in	type Allowed:coded string Default: ”

Returns
coded format

Description
getformat returns the current format value set for the different format possibilities. See the setformat function for the different format type descriptions. The known types are:
prec, aprec, tprec, long, lat, len, dtime, elev, auto, vel, freq, dop, unit.

Example

```
- dq.getformat('prec')
6
- dq.setformat('prec', 12) # set precision to 12 significant digits
T
- dq.getformat('prec')
12
- dq.getformat('long')
hms
```

---

quanta.form.xxx - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

Format a quantity using globaly set formats

Synopsis
form.xxx(v, showform)

Arguments

v	in	value to be converted Allowed:quantity
showform	in	show square brackets and separating , (not for all) Allowed:bool Default: F

Returns
string
If a multi-dimensional value is given for the value *v* in the case of *dtime*, *long* or *lat*, the returned value is a string vector of a length equal to last dimension. Each string has a number of fields equal to the number of elements in all earlier dimensions. If the *showform* is *T*, each vector element is surrounded by a pair of square brackets if there is more than one entry, and fields are separated by a ','.

Description

form.xxx (xxx can be lat, long, len, vel, freq, dtime, unit) will format the input into a string using the global format information set by setformat().

Example

```
- dq.setformat('freq','cm')
T
- dq.form.freq(dq.quantity('5GHz'))
5.99584916 cm
```

---

quanta.convertfreq - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

convert a frequency quantity to another unit

Synopsis
convertfreq(v, out)

Arguments

v	in	quantity to convert Allowed:quantity or string Default: '1Hz'
out	in	unit to convert to Allowed:string or quantity Default: 'Hz'

Returns
quantity

Description
convertfreq converts a frequency quantity to another unit. All units allowed in the setformat are allowed.

Example

```
- dq.convertfreq('5GHz','cm')
[value=5.99584916, unit=cm]
- dq.convertfreq('5cm','GHz')
[value=5.99584916, unit=GHz]
```

---

quanta.convertdop - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

convert a doppler velocity quantity to another unit

Synopsis
convertdop(v, out)

Arguments

v	in	quantity to convert Allowed:quantity or string Default: '0km/s'
out	in	unit to convert to Allowed:string or quantity Default: 'km/s'

Returns
quantity

Description
convertfreq converts a velocity quantity to another unit. Units are either velocity or dimensionless.

Example

```
- dq.convertdop('1','km/s')
[value=299792.458, unit=km/s]
- dq.convertdop('10km/s','1')
[value=3.33564095e-05, unit=]
```

---

quanta.quantity - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

make a quantity from a string or from a value and a unit string

Synopsis
quantity(v, name)

Arguments

v	in	value to convert to quantity Allowed:quantity or string or numeric Default: '1' or 1.0
name	in	unit string if v numeric Allowed:string Default: ''

Returns
quantity

Description
quantity makes a quantity from a string, or from a value and a string. Note that a function *unit* exists which is a synonym for quantity. If only a string is given, it can be a scalar string, a vector of strings, or an array of strings. The result will be a scalar quantity, or an array of quantities. If a numeric value and a string are given, the numeric value can be any numeric type, and can also be an array or a vector of them. See the introduction for an example of all the possibilities.

Example

```
- tu := dq.quantity('1Jy') # make quantity
- tu
[value=1, unit=Jy]
- dq.quantity(tu) # also accepts a quantity
[value=1, unit=Jy]
- tu := dq.unit('1Jy') # make quantity with synonym
```

```
- tu
[value=1, unit=Jy]
- dq.quantity(-1.3, 'Jy') # make quantity with separate value
[value=-1.3, unit=Jy]
```

---

quanta.getvalue - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

get the internal value of a quantity

Synopsis
getvalue(v)

Arguments

v	in	quantity Allowed:quantity or string or numeric

Returns
numeric scalar or array

Description
getvalue returns the internal value of a quantity. It also can handle an array of quantities.

Example

```
- tu := dq.quantity(-1.3, 'Jy')         # make quantity
- tu
[value=-1.3, unit=Jy]
- dq.getvalue(tu)
-1.3
- dq.getunit(tu)
Jy
- a := dq.quantity("5m/s 7A")
```

```
- a::
[id=quant, shape=2]
- dq.getvalue(a)
[5 7]
```

---

quanta.getunit - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

get the internal unit of a quantity

Synopsis
getunit(v)

Arguments

v	in	quantity Allowed:quantity or string or numeric

Returns
string scalar or array

Description
getunit returns the internal unit string of a quantity

Example

```
- tu := dq.quantity(-1.3, 'Jy')          # make quantity
- tu
[value=-1.3, unit=Jy]
- dq.getvalue(tu)
-1.3
- dq.getunit(tu)
Jy
```

---

quanta.canonical - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

get canonical value of quantity

Synopsis
canonical(v)

Arguments

v	in	value to convert Allowed:quantity or string Default: '1'

Returns
quantity

Description
canonical (with alias canon) gets the canonical value of a quantity

Example

```
- dq.canonical('1Jy') # canonical value of a string
[value=1e-26, unit=kg.s-2]
- dq.canon(dq.quantity('1Jy')) # canonical value of a unit
[value=1e-26, unit=kg.s-2]
```

---

quanta.canon - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

get canonical value of quantity

Synopsis
canon(v)

Arguments

v	in	value to convert Allowed:quantity or string Default: '1'

Returns
quantity

Description
canon gets the canonical value of a quantity

Example

```
- dq.canon('1Jy') # canonical value of a string
[value=1e-26, unit=kg.s-2]
- dq.canonical(dq.quantity('1Jy')) # canonical value of a unit
[value=1e-26, unit=kg.s-2]
```

---

quanta.convert - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

convert a quantity to another unit

Synopsis
convert(v, out)

Arguments

v	in	quantity to convert Allowed:quantity or string Default: '1'
out	in	unit to convert to Allowed:string or quantity Default: ''

Returns
quantity

Description
convert converts a quantity to another unit. If no output unit given, conversion is to canonical units

Example

```
- tu := dq.quantity('5Mm/s') # specify a quantity
- tu
[value=5, unit=Mm/s]
- dq.convert(tu, 'pc/a') # convert it to parsec per year
[value=0.00511356, unit=pc/a]
- dq.convert(tu) # convert to canonical units
[value=5e+06, unit=m.s-1]
```

---

quanta.define - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

define a new unit name

Synopsis
define(name, v)

Arguments

name	in	name of unit to define Allowed:string
v	in	quantity value of new unit Allowed:string or quantity Default: '1'

Returns
bool

Description
define defines the name and value of a user defined unit

Example

```
- dq.define('JY','1Jy') # your misspelling
T
- dq.define('VLAunit', '0.898 JY') # a special unit using it
T
- dq.quantity('5 VLAunit')  # check its use
[value=5, unit=VLAunit]
- dq.convert('5 VLAunit','Jy')
[value=4.49, unit=Jy]
```

---

quanta.map - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

list known unit names and constants

Synopsis
map(v)

Arguments

v	in	type of information to list Allowed:coded string Default: 'all'

Returns
string

Description
map lists the known mapping of units and constants. It has a single argument, which can be a coded string (no-case, minimax match):

all all of the following units (not constants): also the default

Prefix known decimal prefixes

SI known SI units

Customary a set of customary units known to programs

User units defined by the user

Constants known constants (note: only 'const', 'Const', 'constants' and 'Constants' recognised).

Example

```
- dq.map('pre') # list decimal prefixes
        == Prefix ==== 20 ====
        E         (exa)                         1e+18
        G         (giga)                        1000000000
        M         (mega)                        1000000
        P         (peta)                        1e+15
        T         (tera)                        1e+12
        Y         (yotta)                       1e+24
        Z         (zetta)                       1e+21
        a         (atto)                        1e-18
        c         (centi)                       0.01
        d         (deci)                        0.1
        da        (deka)                        10
        f         (femto)                       1e-15
        h         (hecto)                       100
        k         (kilo)                        1000
        m         (milli)                       0.001
        n         (nano)                        1e-09
        p         (pico)                        1e-12
        u         (micro)                       1e-06
        y         (yocto)                       1e-24
        z         (zepto)                       1e-21
- dq.map('Constants') # list known constants
```

```
== Constants ====
pi    3.14..                  3.14159
ee    2.71..                  2.71828
c     light vel.              2.99792e+08 m/s
G     grav. const             6.67259e-11 N.m2/kg2
h     Planck const            6.62608e-34 J.s
HI    HI line                 1420.41 MHz
R     gas const               8.31451 J/K/mol
NA    Avogadro #              6.02214e+23 mol-1
e     electron charge         1.60218e-19 C
mp    proton mass             1.67262e-27 kg
mp_me mp/me                   1836.15
mu0   permeability vac.       1.25664e-06 H/m
eps0  permittivity vac.       1.60218e-19 C
k     Boltzmann const         1.38066e-23 J/K
F     Faraday const           96485.3 C/mol
me    electron mass           9.10939e-31 kg
re    electron radius         2.8179e-15 m
a0    Bohr's radius           5.2918e-11 m
R0    solar radius            6.9599e+08 m
k2    IAU grav. const^2       0.000295912 AU3/d2/S0
```

---

quanta.fits - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

define some FITS units

Synopsis
fits()

Arguments

Returns
bool

Description
fits defines some unit names used in reading and writing FITS files.

Example

```
- dq.fits()
T
- dq.map('user')
        == User ==== 19 ====
        BEAM      (dimensionless beam)         1 _
        DAYS      (day)                        86400 s
        DEG       (degree)                     0.0174532925199 rad
        DEGREES   (degree)                     0.0174532925199 rad
        HZ        (hertz)                      1 s-1
        JY        (jansky)                     1e-26 kg.s-2
        KELVIN    (kelvin)                     1 K
        KELVINS   (kelvin)                     1 K
        KM        (km)                         1000 m
        M         (meter)                      1 m
        METERS    (meter)                      1 m
        PASCAL    (pascal)                     1 m-1.kg.s-2
        PIXEL     (dimensionless pixel)        1 _
        S         (second)                     1 s
        SEC       (second)                     1 s
        SECONDS   (second)                     1 s
        VOLTS     (volt)                       1 m2.kg.s-3.A-1
        YEAR      (year)                       31557600 s
        YEARS     (year)                       31557600 s
```

---

quanta.angle - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

show an angle as a formatted string

Synopsis
angle(v, prec, form, showform)

Arguments

v	in	value to output Allowed: angle quantity or string
prec	in	number of digits shown Allowed: numeric Default: 0
form	in	formatting information Allowed: coded string array Default: ”
showform	in	show square brackets and separating , Allowed: bool Default: F

Returns
string

Description
angle converts an angle quantity to a formatted string. The formatting information is a precision (0 is default, 6 includes +-ddd.mm.ss) and a string array of codes (no-case, minimax match): Codes include:

clean delete leading/trailing superfluous separators

no_d do not show degrees part

no_dm do not show degrees and minutes part

dig2 show only 2 digits of degrees in angle format

time show as time (hh:mm:ss.ttt) rather than as angle

If a multi-dimensional value is given for the value v, the returned value is a string vector of a length equal to last dimension. Each string has a number of fields equal to the number of elements in all earlier dimensions. If the *showform* is T, each vector element is surrounded by a pair of square brackets if there is more than one entry, and fields are separated by a ’,’.

Example

```
- tu := dq.quantity('5.7.12.345678') # define an angle
- tu
[value=5.1201, unit=deg]
- dq.angle(tu)     # default output
+005.07.12
```

```
- dq.angle(tu, prec=7) # 7 digits
+005.07.12.3
- dq.angle(tu, prec=4) # 4 digits
+005.07.
- dq.angle(tu, form="tim no_d") # as time, no hours shown
:20:29
```

---

quanta.time - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

show a time (or date) as a formatted string

Synopsis
time(v, prec, form, showform)

Arguments

v	in	value to output Allowed:time quantity or string
prec	in	number of digits shown Allowed:numeric Default: 0
form	in	formatting information Allowed:coded string array Default: ”
showform	in	show square brackets and separating , Allowed:bool Default: F

Returns
string

Description
time converts a time quantity to a formatted string. The formatting information is a precision (0 is default, 6 includes hh.mm.ss) and a string array of codes (no-case, minimax match): Codes include:

clean delete leading/trailing superfluous separators

no_d do not show hours part

no_dm do not show hours and minutes part

ymd include a date as yyyy/mm/dd (date is by default not shown)

dmy include a date as ddMMMyyyy (date is by default not shown)

mjd include a date as Modified Julian Day (date is by default not shown)

fits include a date and show time in FITS format: le from OS

angle show in angle (dd.mm.ss.ttt) rather than time format

day prefix day-of-week to output

local show local time rather than UTC (add timezone offset)

no_time suppress printing of time part

If a multi-dimensional value is given for the value v, the returned value is a string vector of a length equal to last dimension. Each string has a number of fields equal to the number of elements in all earlier dimensions. If the *showform* is T, each vector element is surrounded by a pair of square brackets if there is more than one entry, and fields are separated by a ','.

Example

```
- tu := dq.quantity('today') # a time
- tu
[value=50450.2243, unit=d]
- dq.time(tu) # default format
05:23:01.000
- dq.time(tu,form="dmy")   # show date
02-Jan-1997/05:23:01.000
- dq.time(tu,form="ymd day") # and day
Thu-1997/01/02/05:23:01.000
- dq.time(tu,form="fits")               # FITS format
1997-01-02T05:23:01.000
- dq.time(tu,form="fits local")         # local FITS format
1997-01-02T15:23:01.000+10:00
- dq.time(tu,form="ymd local")          # local time
1997/01/02/15:23:01.000
```

---

quanta.add - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

add quantities

Synopsis
add(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
quantity

Description
add adds two quantities

Example

```
- dq.add('5m', '2yd')
[value=6.8288, unit=m]
```

---

quanta.sub - Function

Package **general**
Module **quanta**
Tool **quanta**

subtract quantities

Synopsis
sub(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
quantity

Description
sub subtracts two quantities

Example

```
- dq.sub('5m', '2yd')
[value=3.1712, unit=m]
```

---

quanta.mul - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

multiply quantities

Synopsis
mul(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '1'

Returns
quantity

Description
mul multiplies two quantities

Example

```
- dq.mul('5m', '3s')
[value=15, unit=m.s]
```

---

quanta.div - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

divides quantities

Synopsis
div(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '1'

Returns
quantity

Description
div divides two quantities

Example

```
- dq.div('5m', '3s')
[value=1.66667, unit=m/(s)]
```

---

quanta.neg - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

negate quantities

Synopsis
neg(v)

Arguments

v	in	value Allowed:quantity or string Default: 1

Returns
quantity

Description
neg negates a quantity

Example

```
- dq.neg('5m')
[value=-5, unit=m]
```

---

quanta.norm - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

normalise angle

Synopsis

norm(v, a)

Arguments

v	in	value Allowed:angle quantity or string
a	in	lower interval boundary Allowed:numeric Default: -0.5

Returns
quantity

Description
norm normalise angles in interval of 2π radians. The default interval is from -0.5 to +0.5 of a full interval (i.e. from -180 to +180 degrees). The lower end of the interval can be set as a fraction of 2π

Example

```
- dq.norm('713deg') #default normalisation
[value=-7, unit=deg]
- dq.norm('713deg', -2.5)  # normalise to interval -900 - -540 deg
[value=-727, unit=deg]
```

---

quanta.le - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

compare quantities

Synopsis
le(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
boolean

Description
le compares two quantities for less than or equal.

Example

```
- dq.le('5m', '2yd')
F
```

---

quanta.lt - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

compare quantities

Synopsis
lt(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
boolean

Description
lt compares two quantities for less than.

Example

```
- dq.lt('5m', '2yd')
F
```

---

quanta.eq - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

compare quantities

Synopsis
eq(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
boolean

Description
eq compares two quantities for equality.

Example

```
- dq.eq('5m', '2yd')
F
```

---

quanta.ne - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

compare quantities

Synopsis
ne(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
boolean

Description
ne compares two quantities for non equality.

Example

```
- dq.ne('5m', '2yd')
T
```

---

quanta.gt - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

compare quantities

Synopsis
gt(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
boolean

Description
gt compares two quantities for greater than.

Example

```
- dq.gt('5m', '2yd')
T
```

---

quanta.ge - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

compare quantities

Synopsis
ge(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string Default: '0'

Returns
boolean

Description
ge compares two quantities for greater than or equal.

Example

```
- dq.ge('5m', '2yd')
T
```

---

quanta.sin - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

sine of quantity

Synopsis
sin(v)

Arguments

v	in	value Allowed:angle quantity or string

Returns
quantity

Description
sin gives sine of angle quantity

Example

```
- dq.sin('7deg')
[value=0.121869, unit=]
```

---

quanta.cos - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

cosine of quantity

Synopsis
cos(v)

Arguments

v	in	value Allowed:angle quantity or string

Returns
quantity

Description
cos gives cosine of angle quantity

Example

```
- dq.cos('7deg')
[value=0.992546, unit=]
```

---

quanta.tan - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

tangent of quantity

Synopsis
tan(v)

Arguments

v	in	value Allowed:angle quantity or string

Returns
quantity

Description
tan gives tangent of angle quantity

Example

```
- dq.tan('7deg')
[value=0.122785, unit=]
```

---

quanta.asin - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

arcsine of quantity

Synopsis
asin(v)

Arguments

v	in	value Allowed:non-dimensioned quantity or string

Returns
quantity

Description
asin gives arcsine of non-dimensioned quantity

Example

```
- dq.convert(dq.asin(dq.sin('7deg')), 'deg')
[value=7, unit=deg]
```

---

quanta.acos - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

arccosine of quantity

Synopsis
acos(v)

Arguments

v	in	value Allowed:non-dimensioned quantity or string

Returns
quantity

Description
acos gives arccosine of non-dimensioned quantity

Example

```
- dq.convert(dq.acos(dq.cos('7deg')), 'deg')
[value=7, unit=deg]
```

---

quanta.atan - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

arctangent of quantity

Synopsis
atan(v)

Arguments

v	in	value Allowed:non-dimensioned quantity or string

Returns
quantity

Description
atan gives arctangent of non-dimensioned quantity

Example

```
- dq.convert(dq.atan(dq.tan('7deg')), 'deg')
[value=7, unit=deg]
```

---

quanta.atan2 - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

arctangent of two quantity

Synopsis
atan2(v, a)

Arguments

v	in	value Allowed:non-dimensioned quantity or string
a	in	value Allowed:non-dimensioned quantity or string

Returns
quantity

Description
atan gives arctangent of two non-dimensioned quantity

Example

```
- dq.convert(dq.atan2(dq.sin('7deg'), dq.cos('7deg')), 'deg')
[value=7, unit=deg]
```

---

quanta.abs - Function

Package **general**
Module **quanta**
Tool **quanta**

absolute value of quantity

Synopsis
abs(v)

Arguments

v	in	value Allowed:quantity or string

Returns
quantity

Description
abs gives absolute value of quantity

Example

```
- dq.abs('-5km/s')
[value=5, unit=km/s]
```

---

quanta.ceil - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

ceil value of quantity

Synopsis
ceil(v)

Arguments

v	in	value Allowed:quantity or string

Returns
quantity

Description
ceil gives ceiling value of quantity

Example

```
- dq.ceil('5.1AU')
[value=6, unit=AU]
```

---

quanta.floor - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

floor value of quantity

Synopsis
floor(v)

Arguments

v	in	value Allowed:quantity or string

Returns
quantity

Description
floor gives flooring value of quantity

Example

```
- dq.floor('-5.1AU')
[value=-6, unit=AU]
```

---

quanta.log - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

logarithm of quantity

Synopsis
log(v)

Arguments

v	in	value Allowed:dimensionless quantity or string

Returns
quantity

Description
log gives natural logarithm of dimensionless quantity

Example

```
- dq.log('2')
[value=0.693147181, unit=]
```

---

quanta.log10 - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

logarithm of quantity

Synopsis
log10(v)

Arguments

v	in	value Allowed:dimensionless quantity or string

Returns
quantity

Description
log10 gives logarithm of dimensionless quantity

Example

```
- dq.log10('2')
[value=0.301029996, unit=]
```

---

quanta.exp - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

exponential of quantity

Synopsis
exp(v)

Arguments

v	in	value Allowed:dimensionless quantity or string

Returns
quantity

Description
exp gives exponential value of dimensionless quantity

Example

```
- dq.exp('2')
[value=7.3890561, unit=]
- dq.exp('2m')
SEVERE: Caught an exception! Event type=run exception=Quantum::exp
       illegal unit type 'm'
```

---

quanta.sqrt - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

square root of quantity

Synopsis
sqrt(v)

Arguments

v	in	value Allowed:dimensionless quantity or string

Returns
quantity

Description
sqrt gives square root of quantity with only even powered dimensions

Example

```
- dq.sqrt('2m2')
[value=1.41421356, unit=m]
- dq.sqrt('2s')
SEVERE: Caught an exception! Event type=run exception=UnitVal::UnitVal
        Illegal unit dimensions for root
```

---

quanta.compare - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

compare dimensionality of units

Synopsis
compare(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	value Allowed:quantity or string

Returns
bool

Description
compare compares the dimensionality of units of two qauntities

Example

```
- dq.compare('5yd/a', '6m/s')   # equal dimensions
T
- dq.compare('5yd', '5s') # unequal dimensions
F
```

---

quanta.check - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

check for proper unit string

Synopsis
check(v)

Arguments

v	in	value Allowed:string or quantity

Returns
bool

Description
check checks if the argument has a properly defined unit string

Example

```
- dq.check('5AE/Jy.pc5/s')
T
- dq.check('7MYs')
F
```

---

quanta.checkfreq - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

check for proper frequency unit

Synopsis
checkfreq(cm)

Arguments

cm	in	value Allowed:string or quantity

Returns
bool

Description
checkfreq checks if the argument has a properly defined frequency interpretable unit string

Example

```
- dq.checkfreq('5GHz')
T
- dq.checkfreq('5cm')
T
- dq.checkfreq('5cm/s2')
F
```

---

quanta.pow - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

raise quantity to power

Synopsis
pow(v, a)

Arguments

v	in	value Allowed:quantity or string
a	in	power Allowed:integer Default: 1

Returns
quantity

Description
pow raises a quantity to an integer power

Example

```
- dq.pow('7.2km/s', -3)
[value=0.00267918, unit=(km/s)-3]
```

---

quanta.constants - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

get a constant

Synopsis
constants(v)

Arguments

v	in	name Allowed:coded string Default: 'pi'

Returns
quantity

Description
constants gets a named constant quantity. Names (no-case, minimax) are:

```
pi    3.14..              3.14159
ee    2.71..              2.71828
c     light vel.          2.99792e+08 m/s
G     grav. const         6.67259e-11 N.m2/kg2
h     Planck const        6.62608e-34 J.s
HI    HI line             1420.41 MHz
R     gas const           8.31451 J/K/mol
NA    Avogadro #          6.02214e+23 mol-1
```

```
e      electron charge          1.60218e-19 C
mp     proton mass              1.67262e-27 kg
mp_me  mp/me                    1836.15
mu0    permeability vac.        1.25664e-06 H/m
eps0   permittivity vac.        1.60218e-19 C
k      Boltzmann const          1.38066e-23 J/K
F      Faraday const            96485.3 C/mol
me     electron mass            9.10939e-31 kg
re     electron radius          2.8179e-15 m
a0     Bohr's radius            5.2918e-11 m
R0     solar radius             6.9599e+08 m
k2     IAU grav. const^2        0.000295912 AU3/d2/S0
```

Example

```
- dq.constants()
[value=3.14159, unit=]
```

---

quanta.is_angle - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

check if valid angle or time quantity

Synopsis
is_angle(v)

Arguments

v	in	value Allowed:angle/time quantity or string

Returns
bool

Description
is_angle checks if the argument is a valid angle/time quantity.

Example

```
- dq.is_angle(dq.constants('pi'))
F
```

---

quanta.totime - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

convert an angle (or a time) to a time

Synopsis
totime(v)

Arguments

v	in	value Allowed:angle/time quantity or string

Returns
time quantity

Description
totime converts an angle quantity (or a time) to a time

Example

```
- dq.totime('2d5m')
[value=0.00578704, unit=d]
```

---

quanta.toangle - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

convert a time (or an angle) to an angle

Synopsis
toangle(v)

Arguments

v	in	value Allowed:angle/time quantity or string

Returns
angle quantity

Description
toangle converts a time quantity (or an angle) to an angle

Example

```
- dq.toangle('5h30m12.6')
[value=82.5525, unit=deg]
```

---

quanta.splitdate - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

split a date/time into a record

Synopsis
splitdate(v)

Arguments

v	in	value Allowed:angle/time quantity or string

Returns
record

Description
splitdate splits a date/time quantity into a record with constituent fields like year, yearday, month etc. All fields will be integer (to enable use as index and easy personal formatting), with the exception of the *s* field which is a double float. See the example for the fields returned.

Example

```
- dq.splitdate('today')
[mjd=52075.0361, year=2001, yearday=166, month=6, monthday=15,
  week=24, weekday=5, hour=0, min=52, sec=1,
   s=1.24799991, msec=247, usec=247999]
- dq.splitdate('183.33333333deg')
[mjd=0.509259259, year=1858, yearday=321, month=11, monthday=17,
   week=46, weekday=3, hour=12, min=13, sec=19,
    s=19.9999992, msec=999, usec=999999]
```

---

quanta.tos - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

convert quantity to string

Synopsis
tos(v, prec)

Arguments

v	in	value Allowed:quantity or string
prec	in	convert precision of value Allowed:integer Default: 0

Returns
string

Description
tos converts a quantity to a string with the precision defined with the *setformat('prec')* (which defaults to 9). If the optional *prec* argument is set to an integer value greater than 1, that precision is used in the conversion

Example

```
- print a := dq.quantity('2.56 yd/s')
[value=2.56, unit=yd/s]
- dq.tos(a)
2.56 yd/s
- a:=dq.quantity(1/7, 'km/s')
- dq.tos(a)
0.142857143 km/s
- dq.tos(a,2)
0.14 km/s
- dq.tos(a,20)
0.14285714285714284921 km/s
- dq.tos(a)
0.142857143 km/s
```

---

quanta.id - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

identification of tool

Synopsis
id()

Arguments

Returns
object id

Description
id will return the object identification. Its use is meant for the toolmanager.

Example

```
- dq.id()
[sequence=1, pid=4774, time=1020735188, host=cetus, agentid=3]
```

---

quanta.type - Function

Package	**general**
Module	**quanta**
Tool	**quanta**

type of tool

Synopsis
type()

Arguments

Returns
string

Description
type will return the tool name.

Example

```
- dq.type()
quanta
```