# GammaLib - Feature #2909

## convert GTime to python datetime object and back

06/25/2019 12:58 PM - Kelley-Hoskins Nathan

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 06/25/2019 |
| **Priority:** | Normal | **Due date:** | |
| **Assigned To:** | Kelley-Hoskins Nathan | **% Done:** | 100% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | 1.7.0 | | |

**Description**

I wrote a python function in my code to convert a GTime object to a python datetime object.

```
def gtime2datetime( gt ) :
  """Convert GTime object to a python datetime object

  **Args:**
    gt : GTime object

  **Returns:**
    datetime.datetime object
  """

  # gt.utc() can spit out seconds=60, which can do wonky
  # things to a simple datetime.datetime object (which
  # requires seconds to be 0-59 inclusive).  This code
  # seems to get around this problem, from:
  # http://stackoverflow.com/a/21029510/2500768
  f = '%Y-%m-%dT%H:%M:%S %Z'
  utc_time_tuple = time.strptime( gt.utc() + ' UTC', f )
  dt = datetime.datetime(1970,1,1) + datetime.timedelta( seconds= calendar.timegm( utc_time_tuple ) )
  return dt
```

I'd like to add this, as well as the reverse operation datetime2gtime(), to the GTime class.  I think since a datetime object doesn't really exist in the c++ world, these functions will probably just be added to the GTime.i swig interface.  In the end their usage would look something like:

```
t = gammalib.GTime()
d = t.datetime() # returns a datetime object
```

```
d = datetime.datetime()
# and then either:
t = gammalib.GTime(d)
# or
t = gammalib.GTime().datetime(d)
```

## History

**#1 - 06/25/2019 12:58 PM - Kelley-Hoskins Nathan**

*- Assigned To set to Kelley-Hoskins Nathan*

**#2 - 06/26/2019 05:08 PM - Kelley-Hoskins Nathan**

*- Status changed from New to Pull request*

#### #3 - 06/26/2019 05:34 PM - Kelley-Hoskins Nathan

GTime.utc() doesn't return fractional seconds (YYYY-MM-DDThh:mm:ss), so I had to edit GTime.cpp to YYYY-MM-DDThh:mm:ss.ssssss in order for the converter to work properly.

#### #4 - 06/26/2019 05:35 PM - Kelley-Hoskins Nathan

*- Status changed from Pull request to In Progress*

#### #5 - 06/27/2019 01:36 PM - Kelley-Hoskins Nathan

*- Status changed from In Progress to Pull request*

Changing the format of the returned utc() string may affect other code, so I added a variable 'precision' argument that defaults to zero.

#### #6 - 06/27/2019 03:56 PM - Kelley-Hoskins Nathan

*- % Done changed from 0 to 80*

#### #7 - 07/02/2019 01:57 PM - Knödlseder Jürgen

*- Target version set to 1.7.0*

*- % Done changed from 80 to 100*

I needed to adapt the code so that it still works on Python version earlier than 2.6. Here is how the code now looks like:

```
%pythoncode %{
    def datetime(self, *args):
        """
        Convert the GTime data into a datetime.datetime object.

        Usage 1:
        import gammalib
        t = gammalib.GTime()
        d = t.datetime()   # returns a datetime.datetime object

        Usage 2:
        import gammalib, datetime
        d = datetime.datetime.now()
        t = gammalib.GTime()
        t.datetime(d)      # set the GTime to the datetime's time.

        Parameters
        ----------
        args[0] : datetime.datetime object, if present, sets gtime to
                  this time.

        Returns
        -------
        datetime : datetime.datetime object if no input argument,
                   otherwise returns nothing
        """
        # Import modules
        #import time, datetime, calendar
        import time, datetime

        # If no datetime argument then return a datetime object
        if len(args) == 0:
            #f = '%Y-%m-%dT%H:%M:%S.%f %Z'
            #d = datetime.datetime.strptime(self.utc(6) + ' UTC', f)
            (t, msecs) = self.utc(6).strip().split('.')
            d = datetime.datetime(*(time.strptime(t, '%Y-%m-%dT%H:%M:%S')[0:6]))
            d = d.replace(microsecond=int(msecs))
            return d

        # ... otherwise, if an argument is given, set the gtime to the datetime
        # argument
        elif len(args) == 1:
```

```
        dt = args[0]
        if type(dt) is datetime.datetime:
            #s = dt.strftime('%Y-%m-%dT%H:%M:%S.%f')
            s = dt.strftime('%Y-%m-%dT%H:%M:%S')+'.%6.6d' % (dt.microsecond)
            self.utc(s)
        else:
            msg  = 'Argument must be a datetime.datetime object, is '
            msg += 'currently ' + str(dt.__class__)
            raise TypeError(msg)

    # ... otherwise raise an exception
    else:
        msg  = 'GTime.datetime() needs 0 or 1 arguments.  It was given '
        msg += '%d.' % len(args)
        raise ValueError(msg)

    # Return
    return
%}
```

Everything is merged into devel.

**#8 - 07/02/2019 01:58 PM - Knödlseder Jürgen**

*- Status changed from Pull request to Closed*