Code In Frame

Here is some python code. Note that the [fragile] keyword is required on the begin frame line. See the following site for additional details: `http://robfelty.com/2008/09/22/beamer-fragile-frames`

```
1   class GridNodeIterator(object):
2       def __init__(self, grid):
3           self.index = 0
4           return
5
6       def __iter__(self):
7           return self
8
9       def next(self):
10          if self.index == self.nodes - 1:
11              raise StopIteration()
12          nodeobj = self.get_nodeobj(self.index)
13          self.index += 1
14          return nodeobj$
```

Code In Figure In Frame

Here is some python code. In this case, the code was taken out of a python source file.

```
def binaryread(file, vartype, shape=(1), charlen=16):
    """Read text, a scalar value, or an array of values from a binary file.
       file is an open file object
       vartype is the return variable type: str, numpy.int32, numpy.float32,
           or numpy.float64
       shape is the shape of the returned array (shape(1) returns a single value)
           for example, shape = (nlay, nrow, ncol)
       charlen is the length of the text string.  Note that string arrays cannot
           be returned, only multi-character strings.  Shape has no affect on strings.
    """
    import struct
```

Code In Figure In Frame

Here is some python code. In this case, the code was taken out of a python source file.

```
#store the mapping from type to struct format (fmt)
typefmtd = {numpy.int32:'i', numpy.float32:'f', numpy.float64:'d'}

#read a string variable of length charlen
if vartype is str:
    result = file.read(charlen*1)

#read other variable types
else:
    fmt = typefmtd[vartype]
    #find the number of bytes for one value
    numbytes = vartype(1).nbytes
    #find the number of values
    nval = numpy.core.fromnumeric.prod(shape)
    fmt = str(nval) + fmt
    s = file.read(numbytes * nval)
    result = struct.unpack(fmt, s)
    if nval == 1:
        result = vartype(result[0])
    else:
        result = numpy.array(result, dtype=vartype)
        result = numpy.reshape(result, shape)
return result
```