```python
#
#
#
#
#
#
#
from casadi import *
```

CasADi provides a mechanism to add assertions in an MX expression graph This can be useful to debug yor code, e.g. debugging why the end-result of a computation yields NaN

Consider this example:

```python
x = MX.sym("x")
y = sin(x)
z = sqrt(y)

f = Function("f", [x], [z])

z0 = f(5)

print z0
```

```
-nan
```

For some mysterious reason we get NaN here

Next, we add an assertion:

```python
y = y.attachAssert(y>0, "bummer") # Add assertion here

z = sqrt(y)

f = Function("f", [x],[z])

try:
  z0 = f(5)
except Exception as e:
  print "An exception was raised here:"
  print e
```

```
An exception was raised here:
 on line 71 of file "/home/travis/build/casadi/binaries/casadi/casadi/core
     /mx/assertion.cpp"
Assertion error: bummer
```

You can combine this with Callback to do powerful assertions

```python
class Dummy(Callback):
  def __init__(self, name, opts={}):
    Callback.__init__(self)
    self.construct(name, opts)
  def get_n_in(self): return 1
  def get_n_out(self): return 1
  def eval(self, arg):
    import numpy
    x = arg[0]
    m = max(numpy.real(numpy.linalg.eig(blockcat([[x,-1],[-1,2]]))[0]))
    print "m=",m
    return [int(m>2)]

foo = Dummy("foo")

y = sin(x)

y = y.attachAssert(foo(y), "you are in trouble") # Add assertion here

z = sqrt(y)

f = Function("f", [x],[z])

z0 = f(5)
```

```
m= 2.30626130593
```