CasADi Python cheatsheet

		dense	sparse
SX	sym	`SX.sym("x",n,m)`	`SX.sym("x",sp)`
	num	`SX(d)`	`SX(sp,d)`
MX	sym	`MX.sym("x",n,m)`	`MX.sym("x",sp)`
	num	`MX(d)`	`MX(DM(sp,d))`
	DM	`DM(d)`	`DM(sp,d)`

Table 1: `d` is a real number, `n` and `m` are integers

Header

```
from casadi import *
```

SX

```
x = SX.sym("x")
y = SX.sym("y",10,2)
a,b,c = SX.sym("[a,b,c]")
```

MX

```
x = MX.sym("x")
y = MX.sym("y",10,2)
```

Transpose

```
B = A.T
```

Products

```
v = mtimes(A,x)  # Matrix product
v = mtimes([x.T,A,x])  # Matrix product
v = A*A       # Element-wise product
```

Concatentation

```
x = vertcat([a,b,c])
x = horzcat([a,b,c])
```

Reshaping

```
column_matrix = vec(m)
reshaped_matrix = reshape(m,[3,4])
```

Slicing

```
x[0,0]
x[:,0]
x[-1,:]
```

Calculus

```
jacobian(sin(a)*b + c,vertcat([a,b,c]))
```

Function SISO

```
f = SXFunction(’f’, [x],[x**2])
```

Function MIMO

```
g = SXFunction(’g’, [x,y],[x**2,x*y,vertcat([x,2*x])])
```

Function MIMO with scheme

```
solver = SXFunction(’nlp’,
  nlpIn(x=x),
  nlpOut(f=f,g=vertcat([x,2*x]))
)
```

Evaluate SISO

```
f.setInput(3)
f.evaluate()
print f.getOutput()
```

Evaluate MIMO

```
g.setInput(5,0)
g.setInput(range(20),1)
g.evaluate()
print g.getOutput(0), g.getOutput(1)
```

Evaluate MIMO with scheme

```
solver.setInput(5,"x")
solver.evaluate()
print solver.getOutput("f"), solver.getOutput("g")
```

Caveats in Python

```
1/2 # integer division => 0
```