Function constructors

```
#
#
#
#
#
#
#
```

```
from casadi import *

x = SX.sym("x")      # A scalar (1-by-1 matrix) symbolic primitive

y = SX.sym("y",2)    # A vector (n-by-1 matrix) symbolic primitive

z = SX.sym("z",2,3) # An n-by-m matrix symbolic primitive


ins =  [x,y] # function inputs

outs = [x,y,vertcat(x,y),y*x,0]

print outs
```

```
[SX(x), SX([y_0, y_1]), SX([x, y_0, y_1]), SX([(y_0*x), (y_1*x)]), 0]
```

```

f = Function("f", ins, outs)
```

f now has two inputs and a 4 outputs:

```
print f.n_in()
```

```
2
```

```
print f.n_out()
```

```
5
```

The outputs has the following string representation. Note how all elements of out have been converted to SX by automatic typecasting functionality

```

f_out = f(*f.sx_in())
for i in range(3):
  print f_out[i]
```

```
x
[y_0, y_1]
[x, y_0, y_1]
```