New calling syntax in CasADi 3.0.0 RC3

```
% MATLAB
x=SX.sym('x');
y=SX.sym('y');
f=Function('f',{x,y},{sin(x)+y},...
           char('x','y'),char('r'));

% Old syntax
res = f({2, 3});
disp(res{1});
res = f(struct('x',2,'y',3));
disp(res.r)

% New syntax (I)
res = f.call({2, 3});
disp(res{1});
res = f.call(struct('x',2,'y',3));
disp(res.r)

% New syntax (II)
res = f(2, 3);
disp(res);
res = f('x',2,'y',3);
disp(res.r)
```

```
# Python
x=SX.sym('x')
y=SX.sym('y')
f=Function('f',[x,y],[sin(x)+y],\
           ['x','y'],['r'])

# Old syntax
res = f([2, 3])
print res[0]
res = f({'x',2,'y',3});
print res['r']

# New syntax (I)
res = f.call([2, 3])
print res[0]
res = f.call({'x',2,'y',3});
print res['r']

# New syntax (II)
res = f(2, 3)
print res
res = f(x=2,y=3);
print res['r']
```

Feel free to use for projects, but exercises use old syntax!

Concatenation in Python

```
1  # Python
2  x=SX.sym('x')
3  y=SX.sym('y')
4
5  # List
6  v = [x,y]
7
8  # Old syntax
9  v = vertcat([x,y])
10 v = horzcat([x,y])
11
12 # New syntax
13 v = vertcat(x,y)
14 v = horzcat(x,y)
15 v = vertcat(*[x,y])
16 v = horzcat(*[x,y])
```

SX and MX?

- SX
  - Low overhead
  - "Readable" output
  - More simplification, optimization
  - All function calls are "inlined"
- MX
  - Larger overhead
  - Efficient for vector/matrix-valued operations
  - Can contain function calls
- Idea: SX for low-level (e.g. DAE right-hand-side), MX as a "glue" (e.g. NLP objective & constraints)
- Can use MX everywhere, *expand* converts MX to SX

```
1 x=MX.sym('x')
2 y=MX.sym('y')
3 f = Function('f',{x,y},{sin(x)*y});
4 f = f.expand()
```