1 “In-text” listing highlighting

```
class MyClass(Yourclass):
    def __init__(self, my, yours):
        bla = '5␣1␣2␣3␣4'
        print bla
```

2 External listing highlighting

```
import src.blocks as b
import src.flux as f
import src.problem as p

from numpy import cumsum

################# Geometries
# radius -> [inner, tubing, insulation]
L = 0.3
tubeGeom = {'type':'cyl','r':cumsum([3.0,1.675,9.525])*1e-3/2,'L':L,\
'cL':L,'m':['silicon_tubing','silicon_insulation']}
windowGeom = {'type':'plate','w':0.3,'L':L,'cL':0.006,'m':['glass']}
IGUGeom = {'type':'plate','w':0.3,'L':L,'cL':0.006,'m':['glass','argon','glass']}

# define inlet water
w0 = b.Block('water0','water')
w0.var['T'] = 13
# define inlet air
a0 = b.Block('air0','air')
a0.var['T'] = 20

w0.mdot = 8.5e-07*w0.m['rho'](w0.var)
a0.mdot = 2.0*a0.m['rho'](a0.var)
# define Exterior boundary condition
aExt = b.Block('Exterior','air')
aExt.var['T'] = 25.0
# define Interior boundary condition
aInt = b.Block('Interior','air')
aInt.var['T'] = 22.5

################# Sources
qw = 0.008 # Heat flow into water from Module Heat Receiver
qa = 0.003 # Heat flow into air from Heat Loss from the Module
# Define as token function, of a block
def Sa(b):
        return {'T':-qa}
def Sw(b):
        return {'T':-qw}
```

```
n = 3
# There are n modules, and n+1 "tube" regions
water = [w0]
air = [a0]

for i in range(1,2*n+2):
        # odd regions are "tube" regions
        water.append(b.Block('water' + str(i),'water'))
        air.append(b.Block('air' + str(i),'air'))
        if(i % 2 == 1):
                water[i].addFlux(f.Flux(water[i],air[i],'heatConduction',tubeGeom))
                air[i].addFlux(f.Flux(water[i],air[i],'heatConduction',tubeGeom))
                air[i].addFlux(f.Flux(aInt,air[i],'heatConduction',IGUGeom))
                air[i].addFlux(f.Flux(aExt,air[i],'heatConduction',windowGeom))
        else:
                # These are "boundary" region
                water[i].addSource(Sw)
                air[i].addSource(Sa)

        air[i].addFlux(f.Flux(air[i-1],air[i],'heatConvection'))
        water[i].addFlux(f.Flux(water[i-1],water[i],'heatConvection'))
        air[i].var['T'] = 22
        water[i].var['T'] = 15
        air[i].mdot = a0.mdot
        water[i].mdot = w0.mdot

# Start the problem with solvable blocks
ICSolar = p.Problem(air[1::]+water[1::])
ICSolar.solve()
ICSolar.printSolution()
```

3 Inline highlighting

Definition `class MyClass` means . . .