```
p = Function(name='p', grid=grid, space_order=2)
pd = Function(name='pd', grid=grid, space_order=2)
p.data[:] = 0.
pd.data[:] = 0.

# Initialize the source term `b`
b = Function(name='b', grid=grid)
b.data[:] = 0.
b.data[int(nx / 4), int(ny / 4)] = 100
b.data[int(3 * nx / 4), int(3 * ny / 4)] = -100

# Create Laplace equation base on `pd`
eq = Eq(pd.laplace, b, subdomain=grid.interior)
# Let SymPy solve for the central stencil point
stencil = solve(eq, pd)
# Now we let our stencil populate our second buffer `p`
eq_stencil = Eq(p, stencil)

# Create boundary condition expressions
x, y = grid.dimensions
t = grid.stepping_dim
bc = [Eq(p[x, 0], 0.)]
bc += [Eq(p[x, ny-1], 0.)]
bc += [Eq(p[0, y], 0.)]
bc += [Eq(p[nx-1, y], 0.)]

# Now we can build the operator that we need
op = Operator([eq_stencil] + bc)

# Run the outer loop explicitly in Python
for i in range(nt):
    # Determine buffer order
    if i % 2 == 0:
        _p = p
        _pd = pd
    else:
        _p = pd
        _pd = p

    # Apply operator
    op(p=_p, pd=_pd)
```