















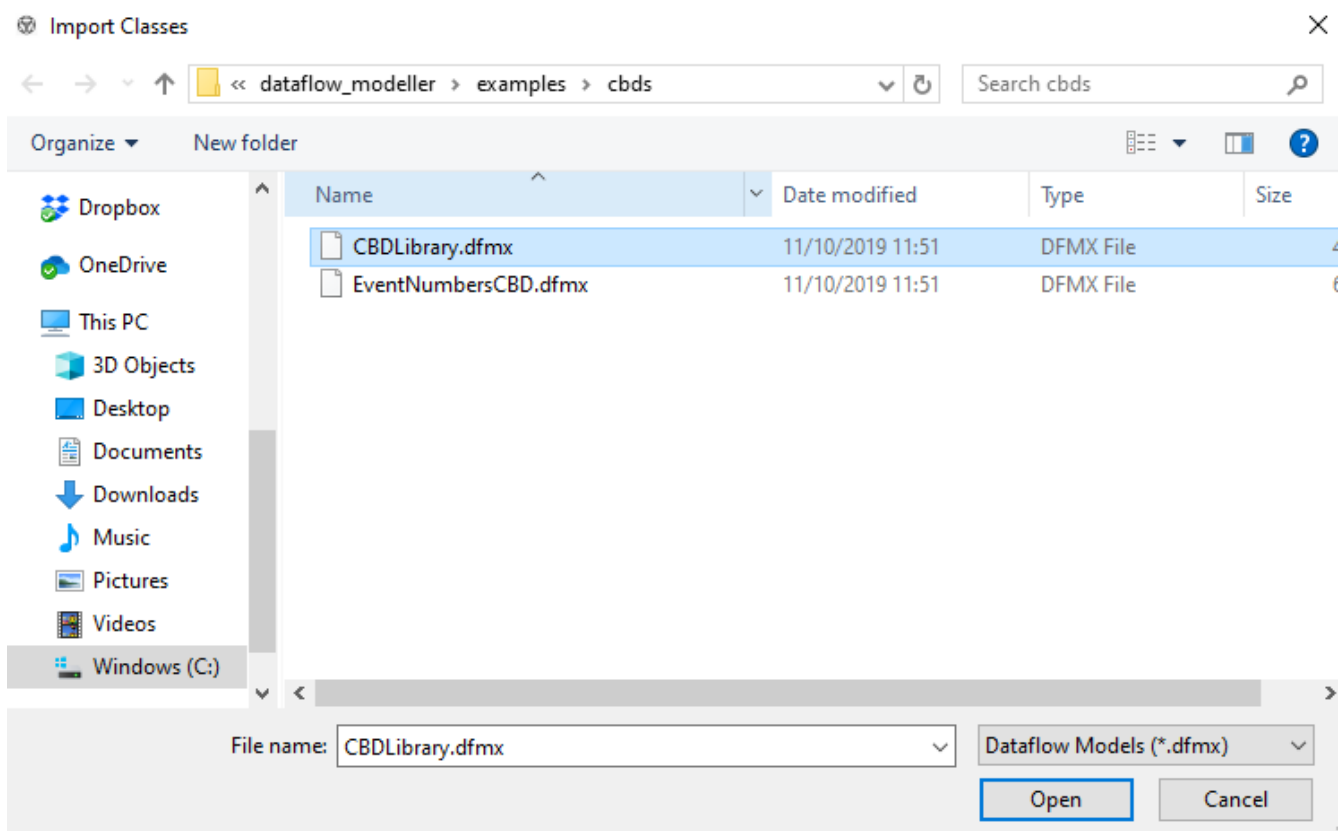
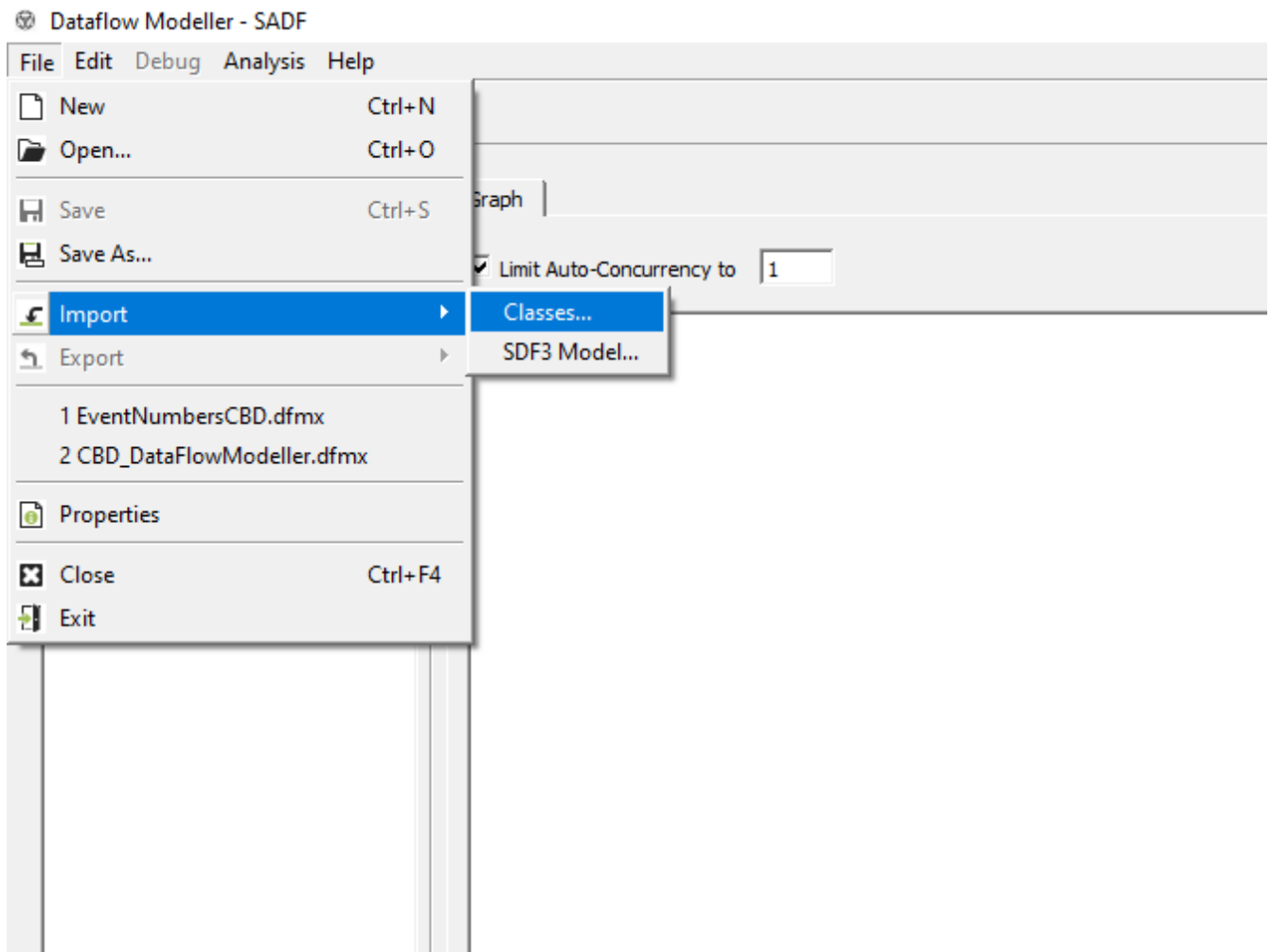
# Dataflow Modeler for CBD Modelling

## Tutorial: Developing a single CBD

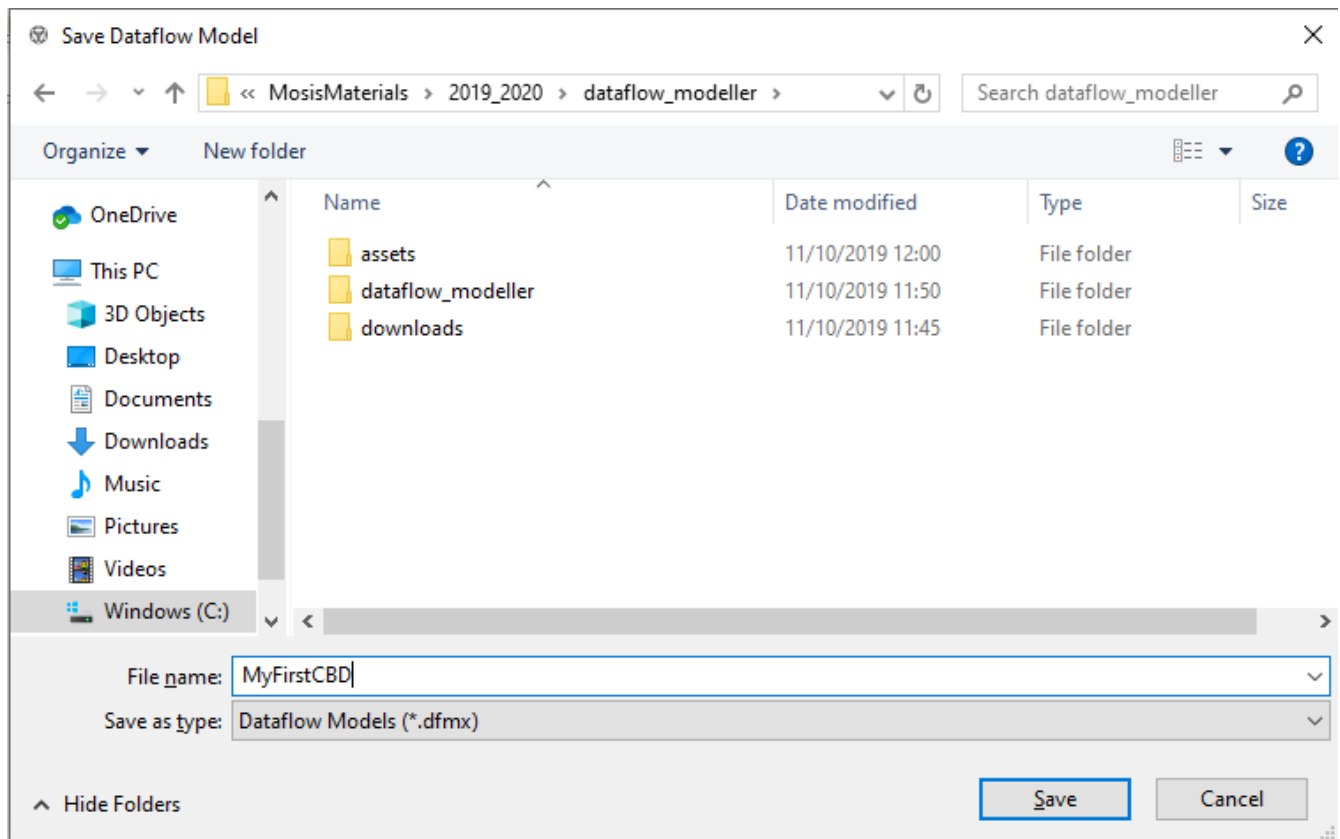
Step 1. Open Dataflow Modeler by executing dfm.exe:

Name	Date modified	Type	Size
 examples	11/10/2019 11:51	File folder	
 platforms	11/10/2019 11:50	File folder	
 plugins	11/10/2019 11:50	File folder	
 dfm.exe	11/10/2019 11:50	Application	1,060 KB
 dfmcore.dll	11/10/2019 11:50	Application exten...	540 KB
 libgcc_s_dw2-1.dll	11/10/2019 11:50	Application exten...	118 KB
 libstdc++-6.dll	11/10/2019 11:50	Application exten...	1,505 KB
 libwinpthread-1.dll	11/10/2019 11:50	Application exten...	78 KB
 LICENSE	11/10/2019 11:50	File	8 KB
 Qt5Core.dll	11/10/2019 11:50	Application exten...	6,069 KB
 Qt5Gui.dll	11/10/2019 11:50	Application exten...	6,339 KB
 Qt5Widgets.dll	11/10/2019 11:50	Application exten...	6,108 KB
 Qt5Xml.dll	11/10/2019 11:50	Application exten...	211 KB
 README	11/10/2019 11:50	File	2 KB

Step 2. Import the `CBDLibrary` classes:

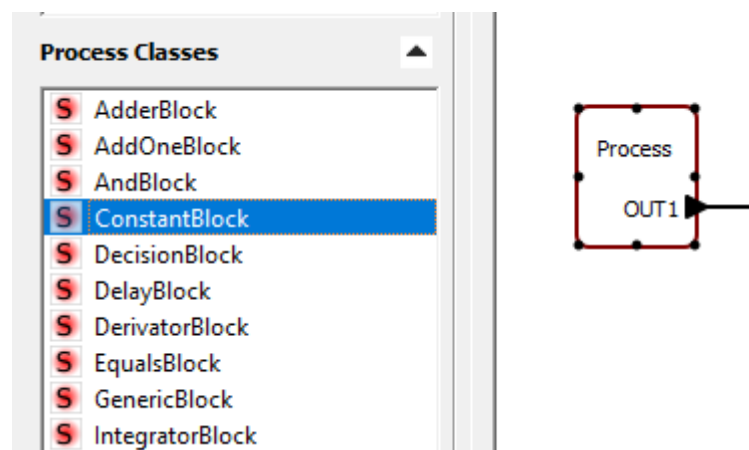


**Step 3. Save your empty CBD model:**

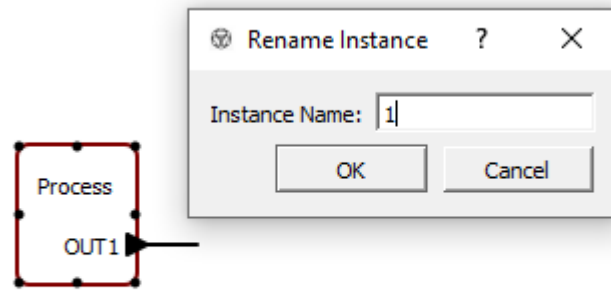


#### Step 4. Create a counter CBD.

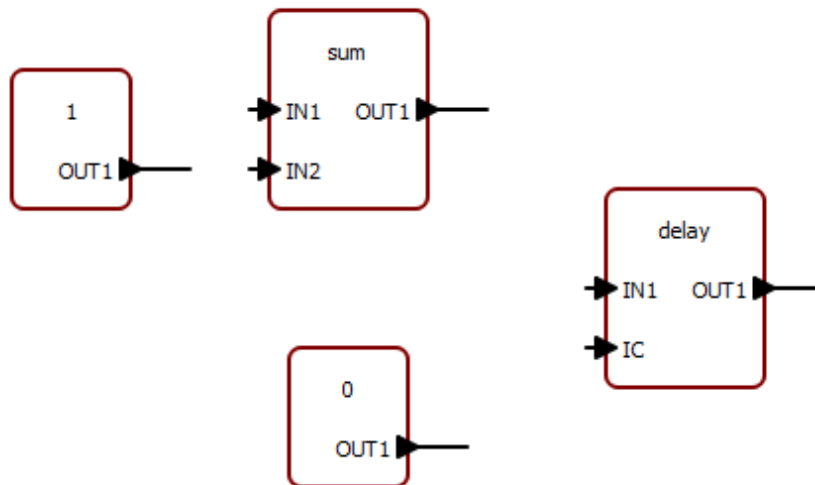
Drag and drop a ConstantBlock into the canvas:



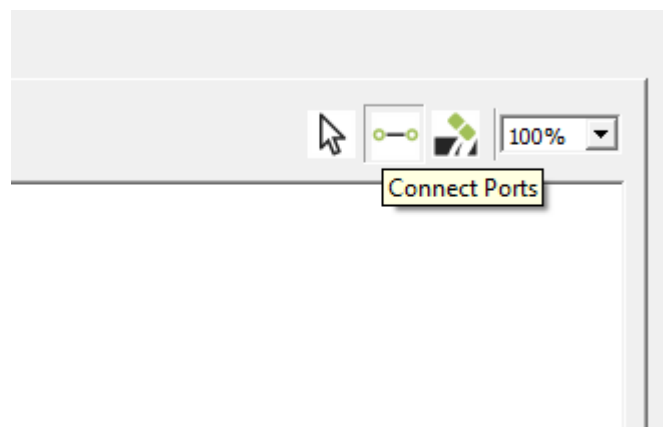
Right click on the `Process` block and rename it to "1":



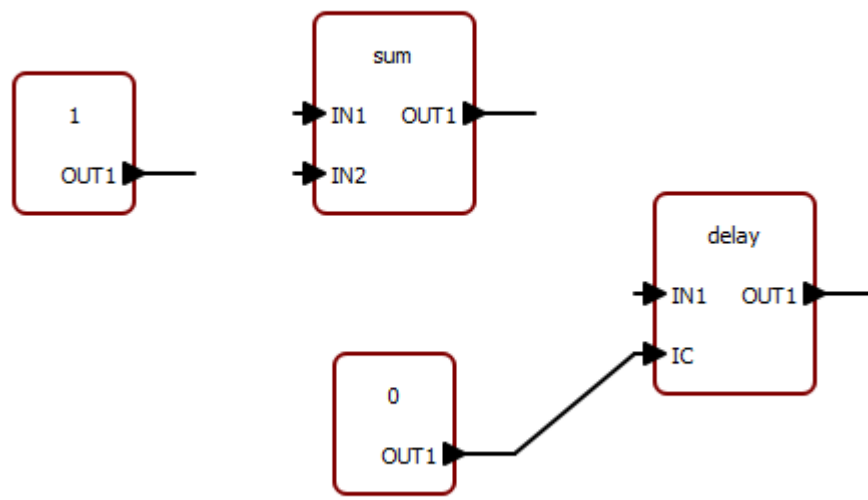
Add the remaining blocks (a `ConstantBlock`, `AdderBlock`, and a `DelayBlock`), and rename them accordingly:



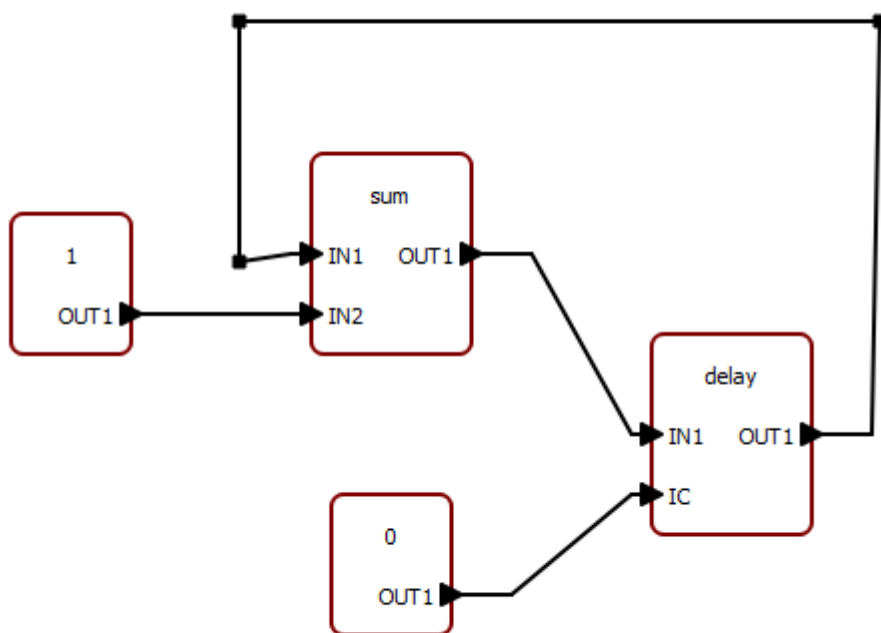
Select the `Connect Ports` tool:



Connect the `out1` output port of the `0` block to the `IC` input port of the delay block by clicking once in the first output port, and clicking a second time in the target input port:



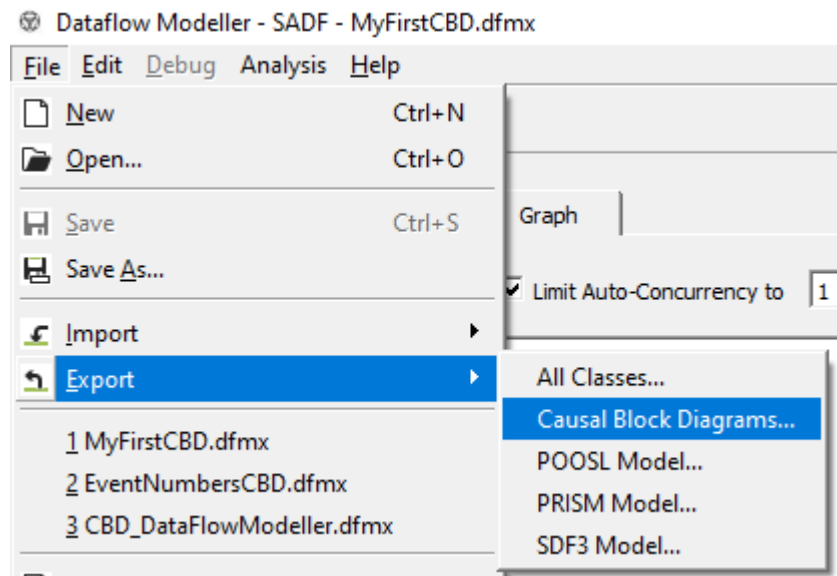
You can create "elbows" by clicking anywhere in the canvas.  
Create the remaining connections according to the figure:



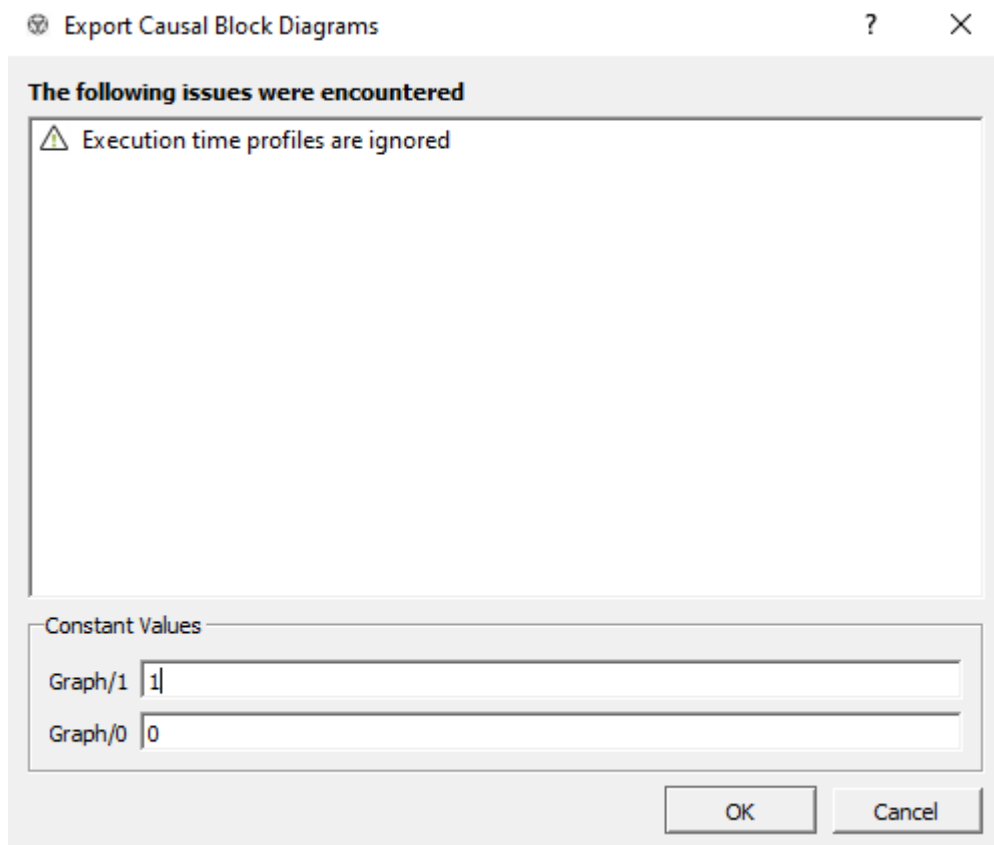
Don't forget to keep saving your CBD.

### Step 5. Export the CBD Code.

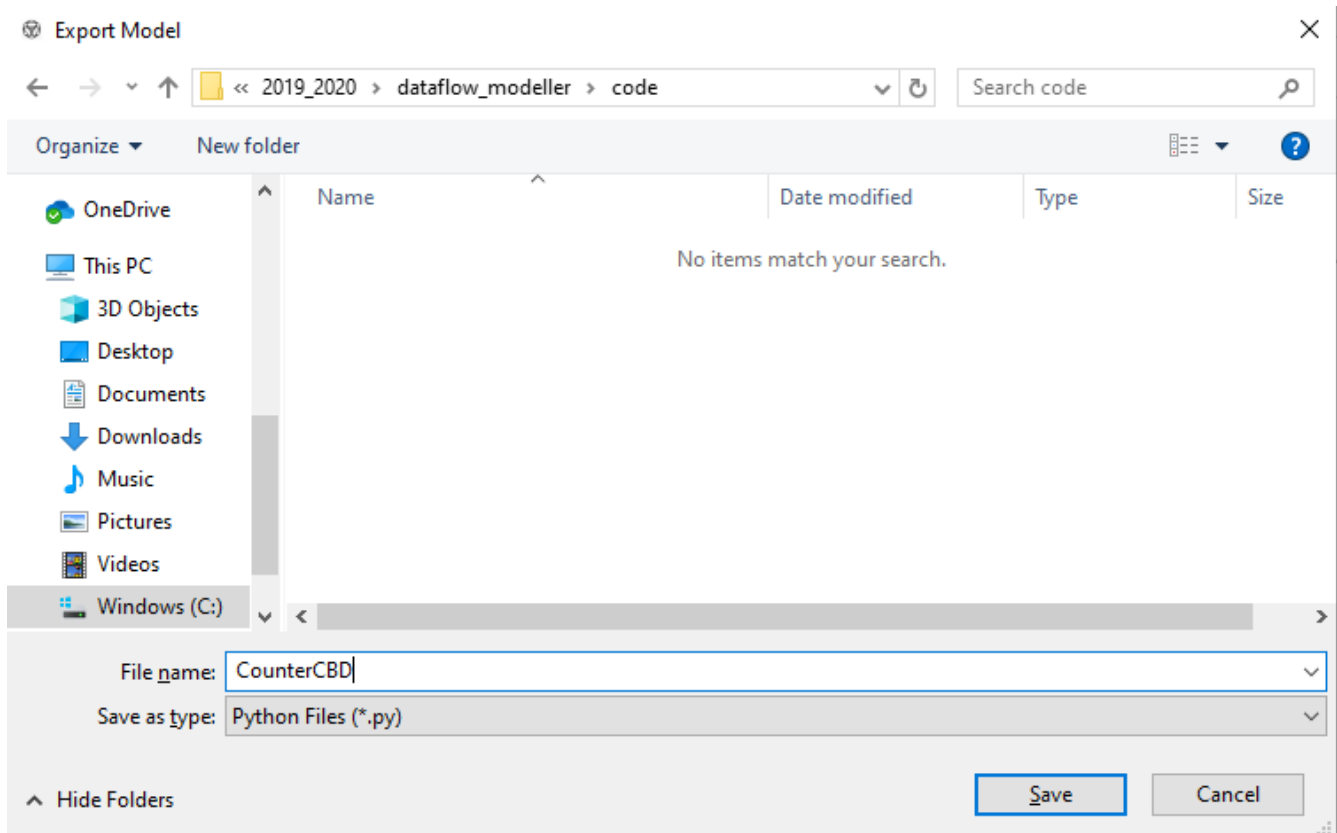
Go to File, Export, Causal Block Diagrams:



Define the following constant values:



Hit **OK** and select the location where the code is to be exported:



The code looks like the following:

```

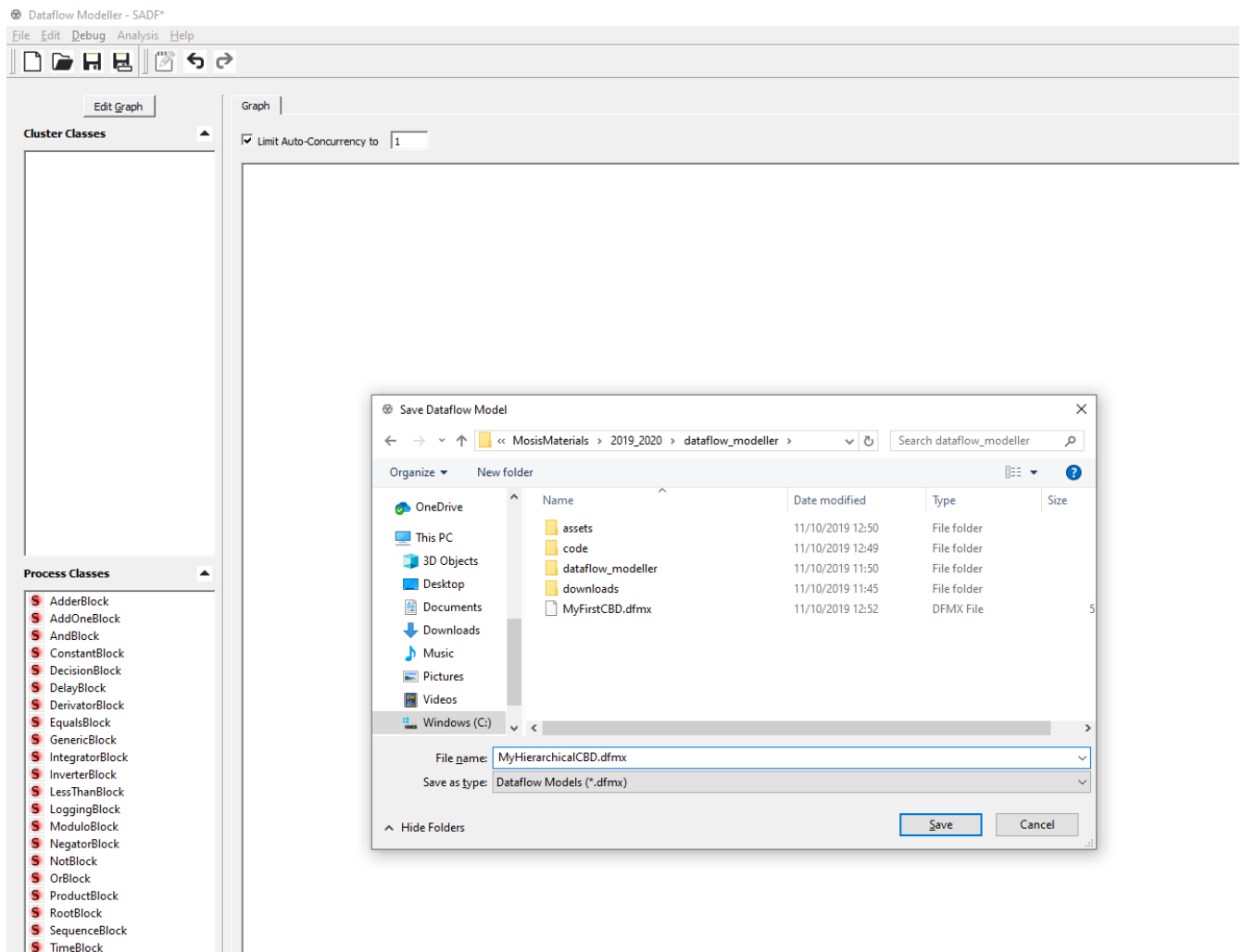
1  #!/usr/bin/env python
2  from CBDLibrary import *
3  from CBDDraw import *
4
5  class Graph(CBD):
6      def __init__(self, block_name):
7          CBD.__init__(self, block_name)
8          self.addBlock(ConstantBlock(block_name="1", value=1))
9          self.addBlock(DelayBlock(block_name="delay"))
10         self.addBlock(ConstantBlock(block_name="0", value=0))
11         self.addBlock(AdderBlock(block_name="sum"))
12         self.addConnection("0", "delay", input_port_name="IC", output_port_name="OUT1")
13         self.addConnection("1", "sum", input_port_name="IN2", output_port_name="OUT1")
14         self.addConnection("sum", "delay", input_port_name="IN1", output_port_name="OUT1")
15         self.addConnection("delay", "sum", input_port_name="IN1", output_port_name="OUT1")
16
17     CBD = Graph("CBD")
18

```

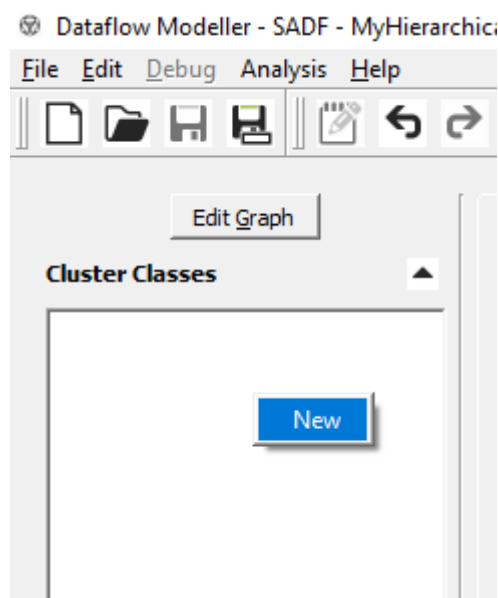
## Tutorial: Developing a Hierarchical CBD

**Step 0.** Follow the previous tutorial if you haven't yet.

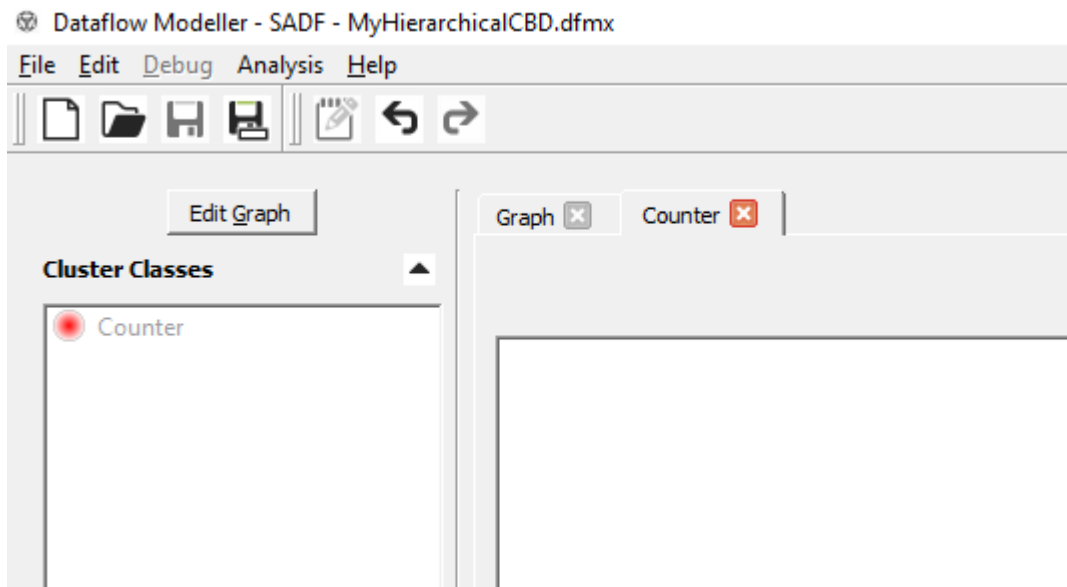
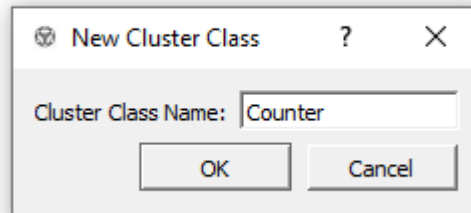
**Step 1.** Create a new empty CBD file, and call it: `MyHierarchicalCBD.dfm`.



Step 2. Right click in cluster classes and create a new one named **Counter**

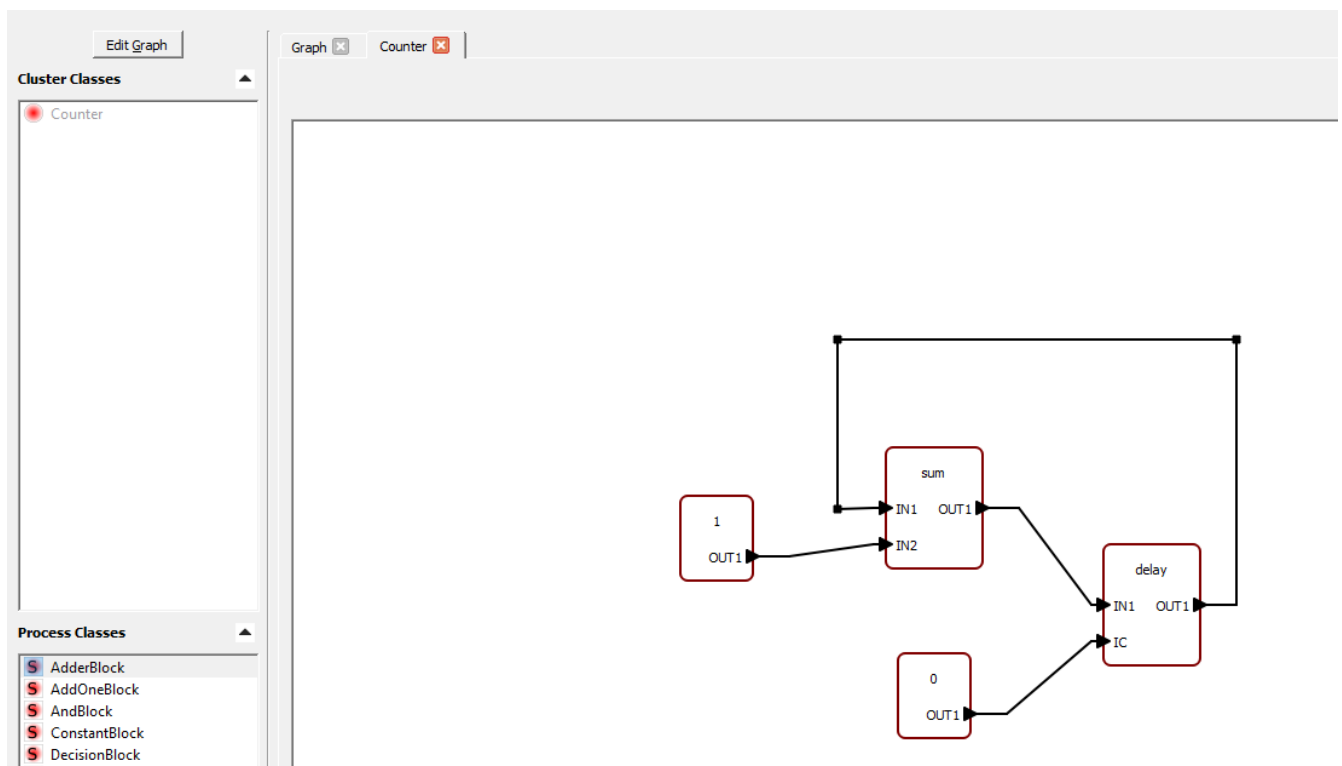






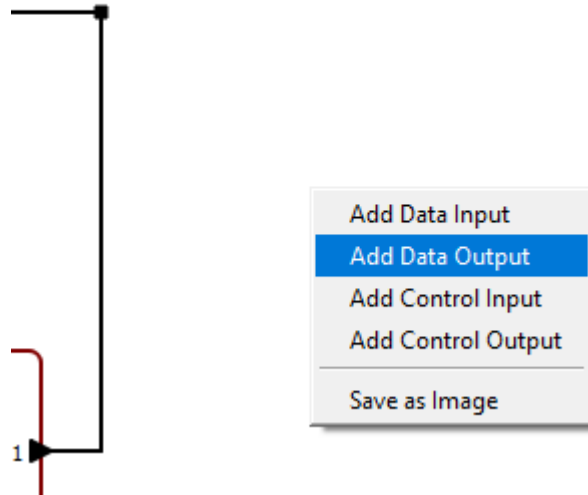
### Step 3. Create the counter CBD from the first tutorial.

Note that the CBD is created inside the Counter cluster class:

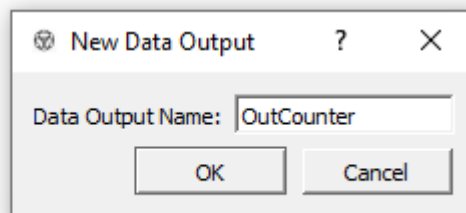


#### Step 4. Add an output to the Counter CBD.

Right click on the canvas, and select **Add Data Output**:

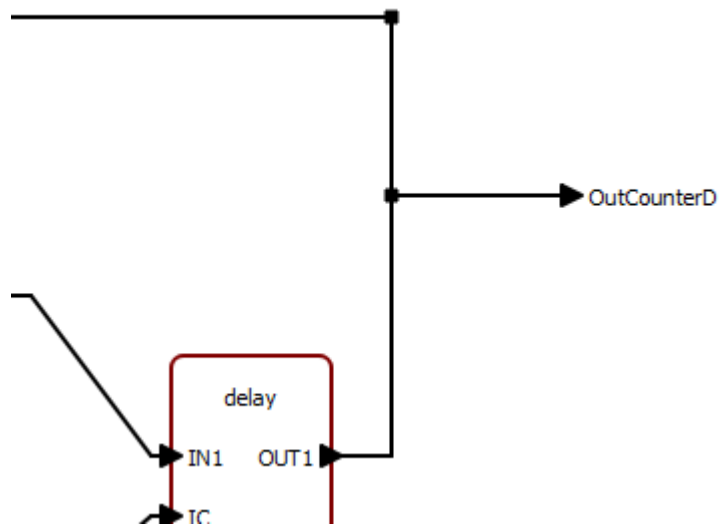


Name it **OutCounter**:



#### Step 5. Connect the output of the delay to the **OutCounter** port.

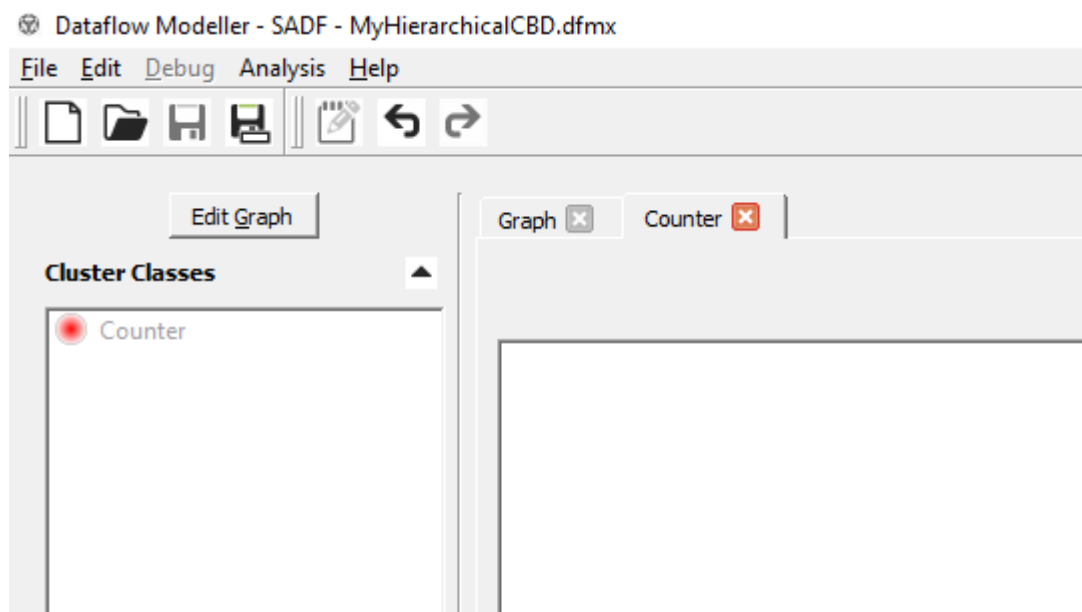
Using the **Connect Ports** tool, click on the **OutCounter** port and then click on the connection leaving the output of the delay block:



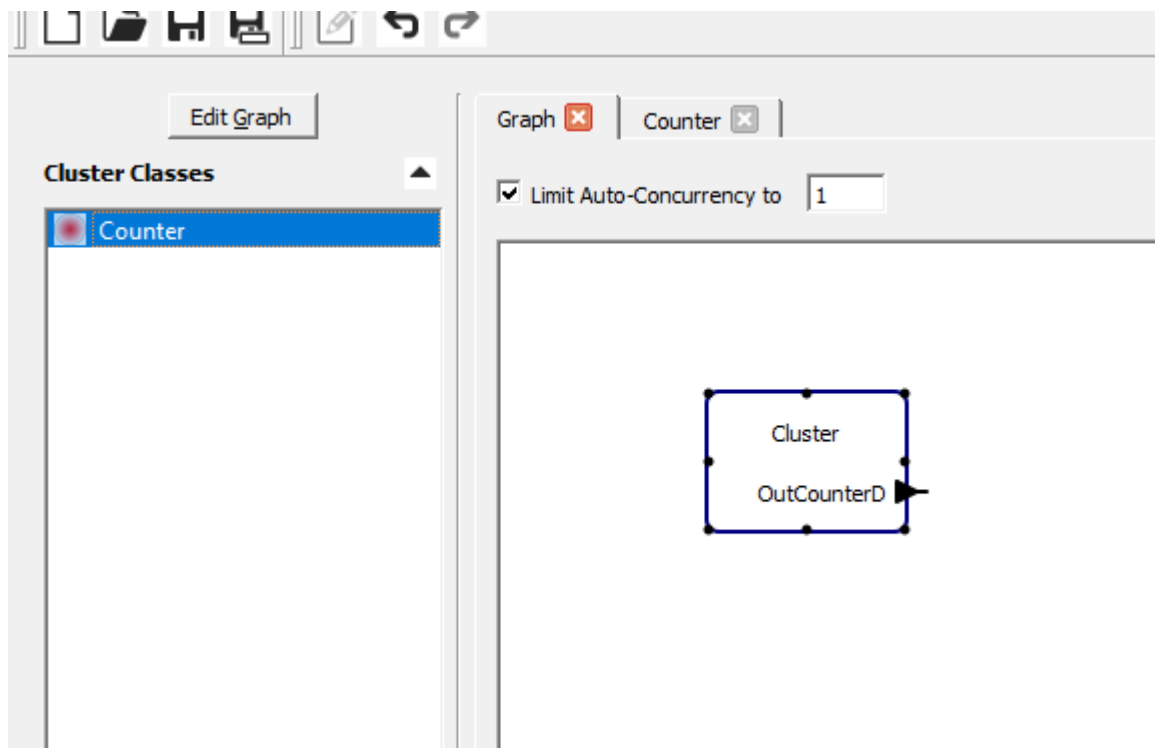
Don't forget to keep saving your CBD.

### Step 6. Instantiate the Counter CBD.

Click on the **Edit Graph** button:



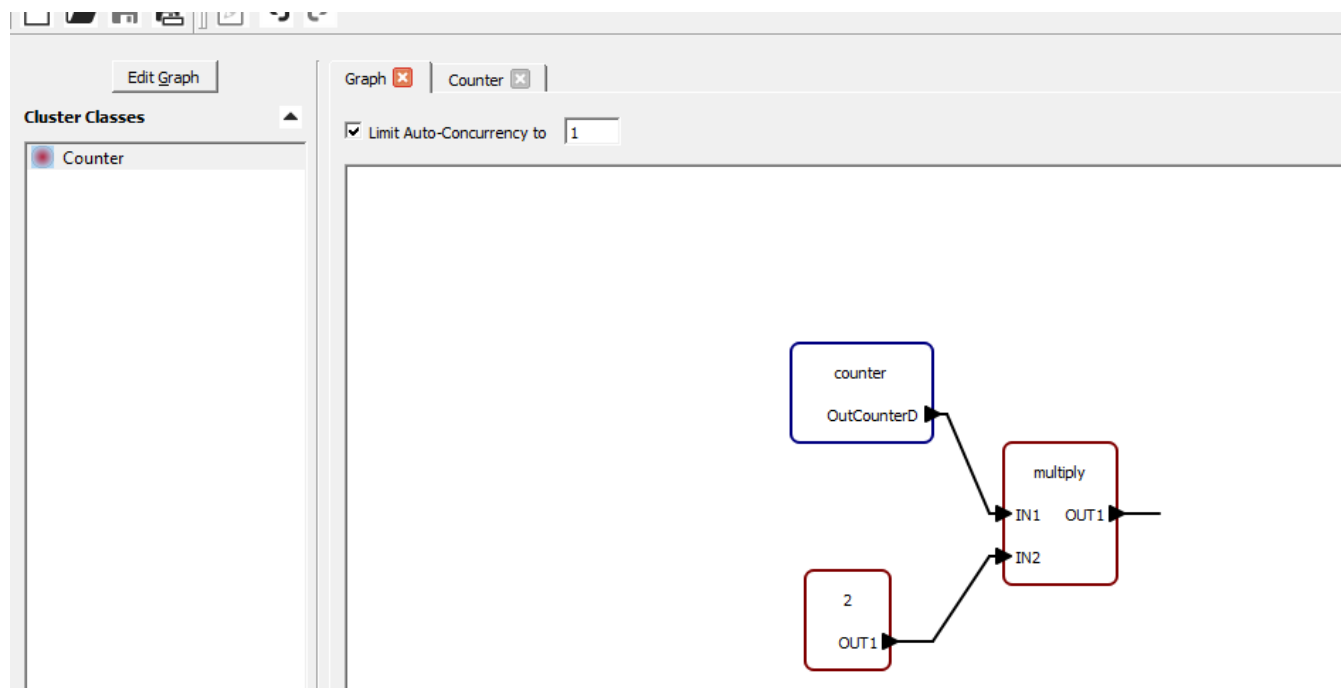
Drag and drop the **counter** cluster class:



And rename it to `counter`.

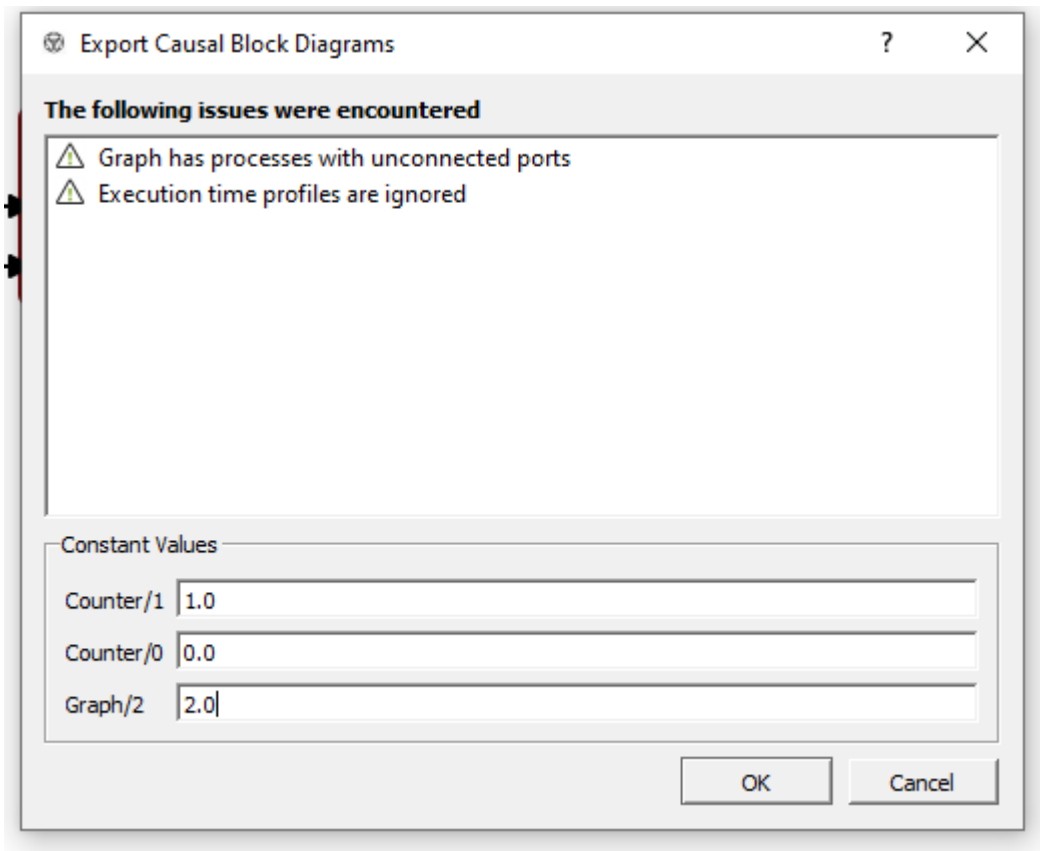
### Step 7. Finish the CBD

On the Graph, add a few more block, as follows:

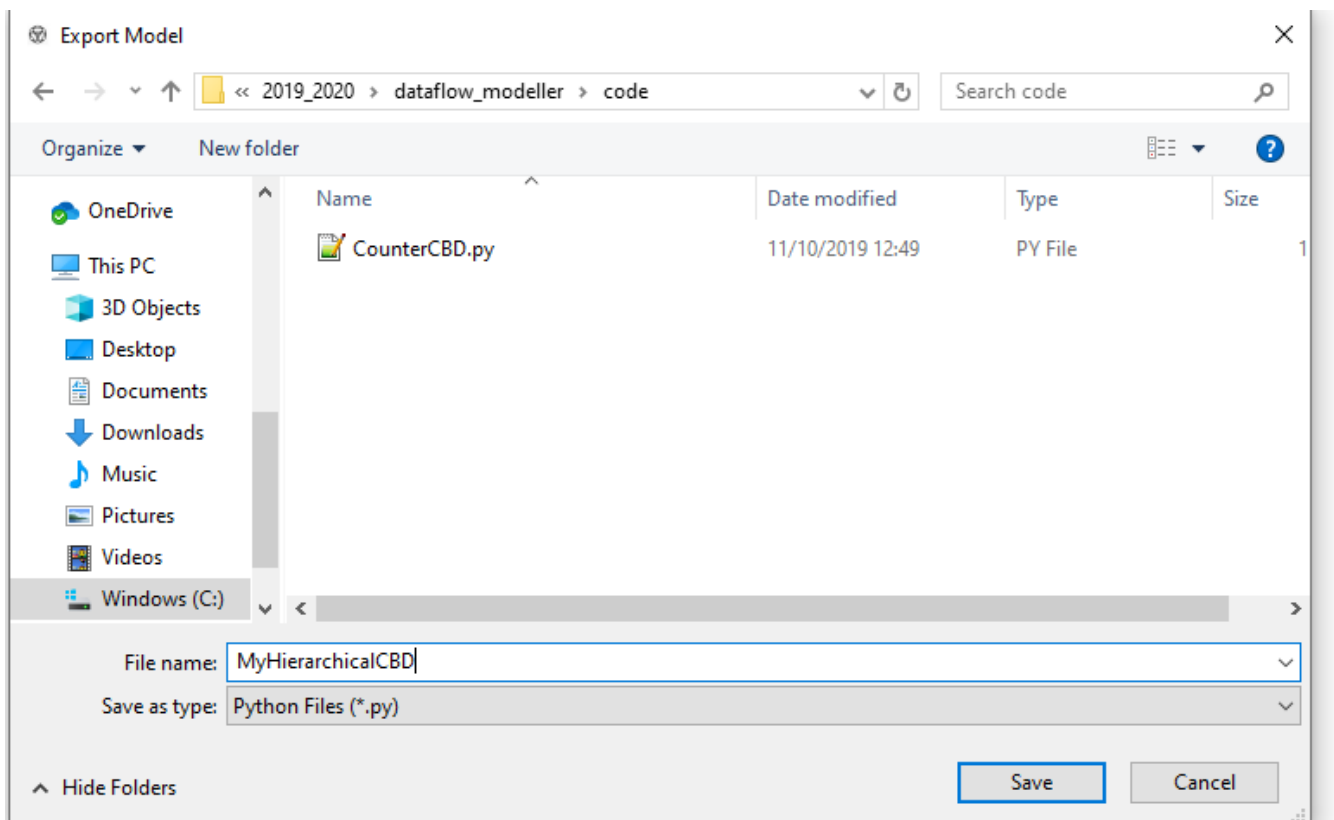


### Step 8. Export the CBD.

Go to File, Export, Causal Block Diagrams, and define the following constant values:



Select an adequate name for the generate python file:



And enjoy:

```

1  #!/usr/bin/env python
2  from CBDLibrary import *
3  from CBDDraw import *
4
5  class Counter(CBD):
6      def __init__(self, block_name):
7          CBD.__init__(self, block_name, input_ports=[], output_ports=["OutCounterD"])
8          self.addBlock(DelayBlock(block_name="delay"))
9          self.addBlock(ConstantBlock(block_name="1", value=1))
10         self.addBlock(ConstantBlock(block_name="0", value=0))
11         self.addBlock(AdderBlock(block_name="sum"))
12         self.addConnection("1", "sum", input_port_name="IN2", output_port_name="OUT1")
13         self.addConnection("sum", "delay", input_port_name="IN1", output_port_name="OUT1")
14         self.addConnection("0", "delay", input_port_name="IC", output_port_name="OUT1")
15         self.addConnection("delay", "sum", input_port_name="IN1", output_port_name="OUT1")
16         self.addConnection("delay", "OutCounterD", output_port_name="OUT1")
17
18  class Graph(CBD):
19      def __init__(self, block_name):
20          CBD.__init__(self, block_name)
21          self.addBlock(Counter(block_name="counter"))
22          self.addBlock(ProductBlock(block_name="multiply"))
23          self.addBlock(ConstantBlock(block_name="2", value=2))
24          self.addConnection("counter", "multiply", input_port_name="IN1", output_port_name="OutCounterD")
25          self.addConnection("2", "multiply", input_port_name="IN2", output_port_name="OUT1")
26
27  CBD = Graph("CBD")
28

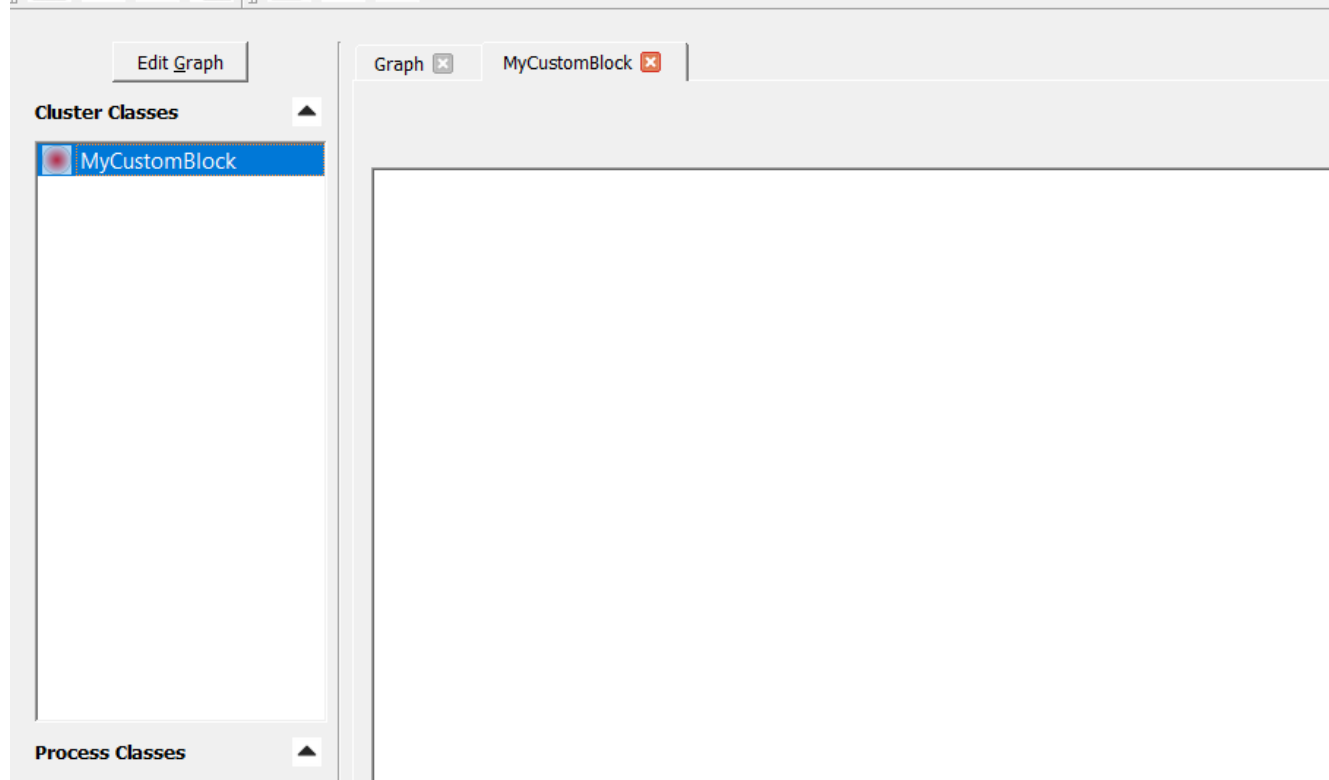
```

## Tutorial: Working with Custom Blocks

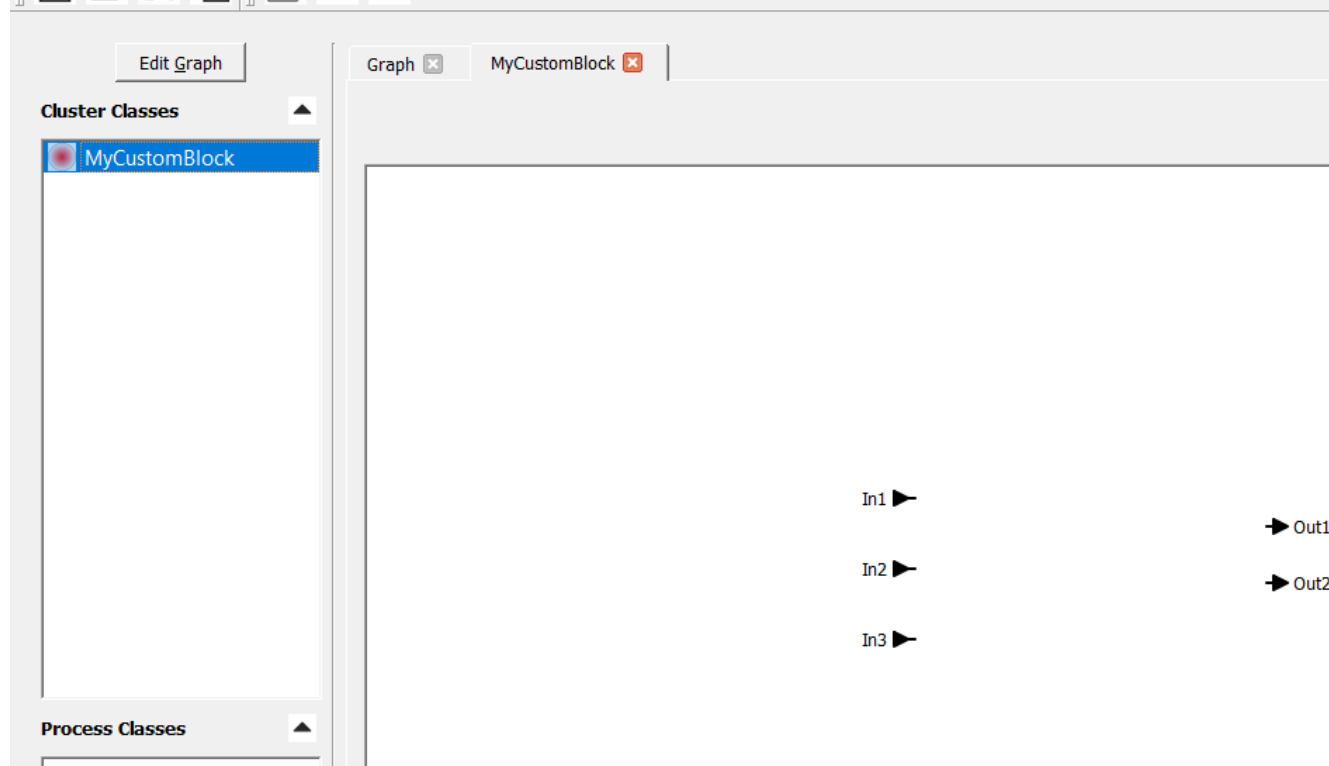
Follow this tutorial if you want to code your own block, and still use Dataflow Modeler to link that block with other blocks.

**Step 0. Follow the previous tutorials if you haven't yet.**

**Step 1. Open Dataflow Modeler and create a new cluster. Name it `MyCustomBlock`:**




Step 2. Add the required data inputs and outputs to the `MyCustomBlock` cluster.

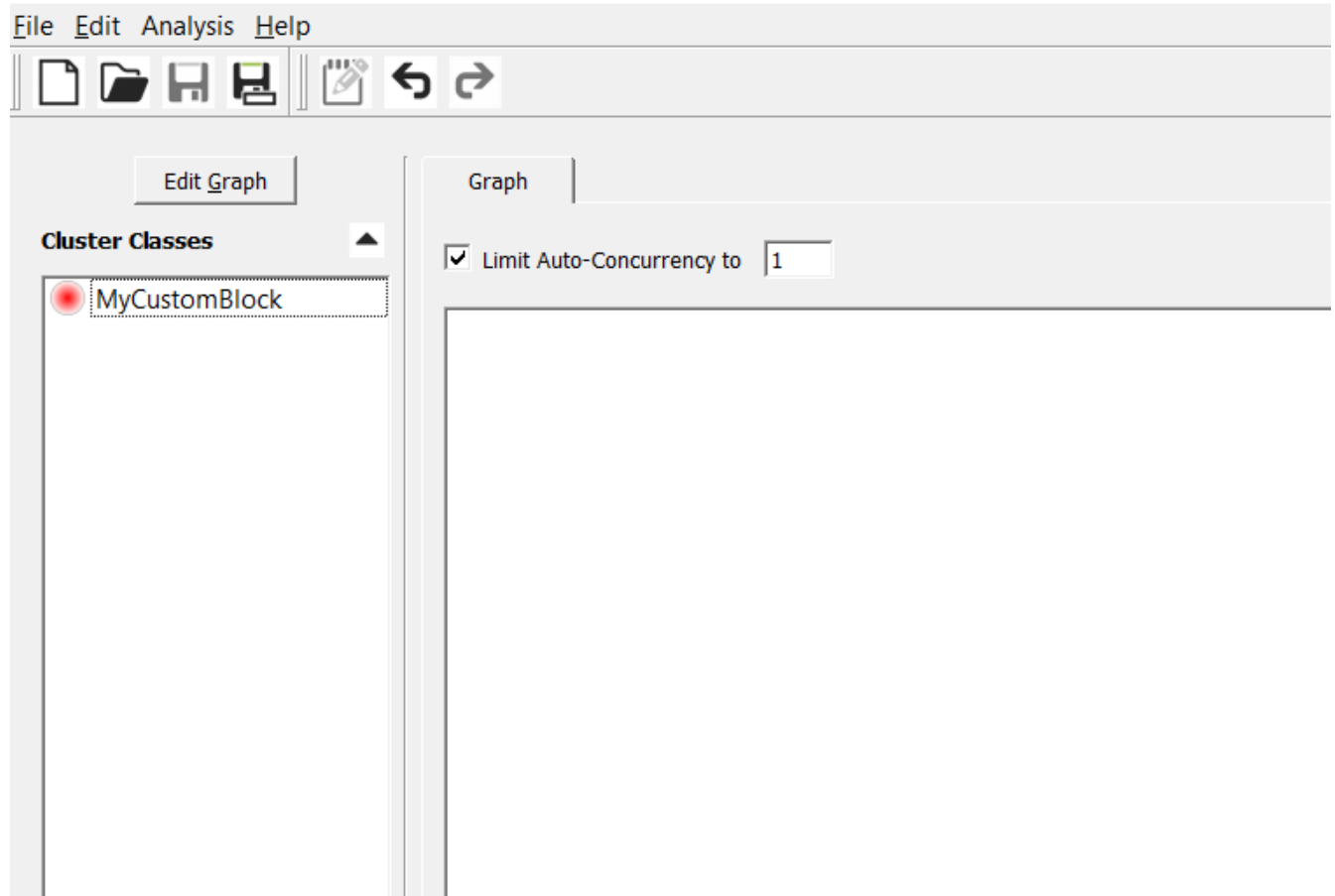


**Step 3. Save your CBD as `MyCustomBlock.dfm`.**

**Step 4. Create a new CBD, choose `File -> Import -> Classes`, and select `MyCustomBlock.dfm`.**

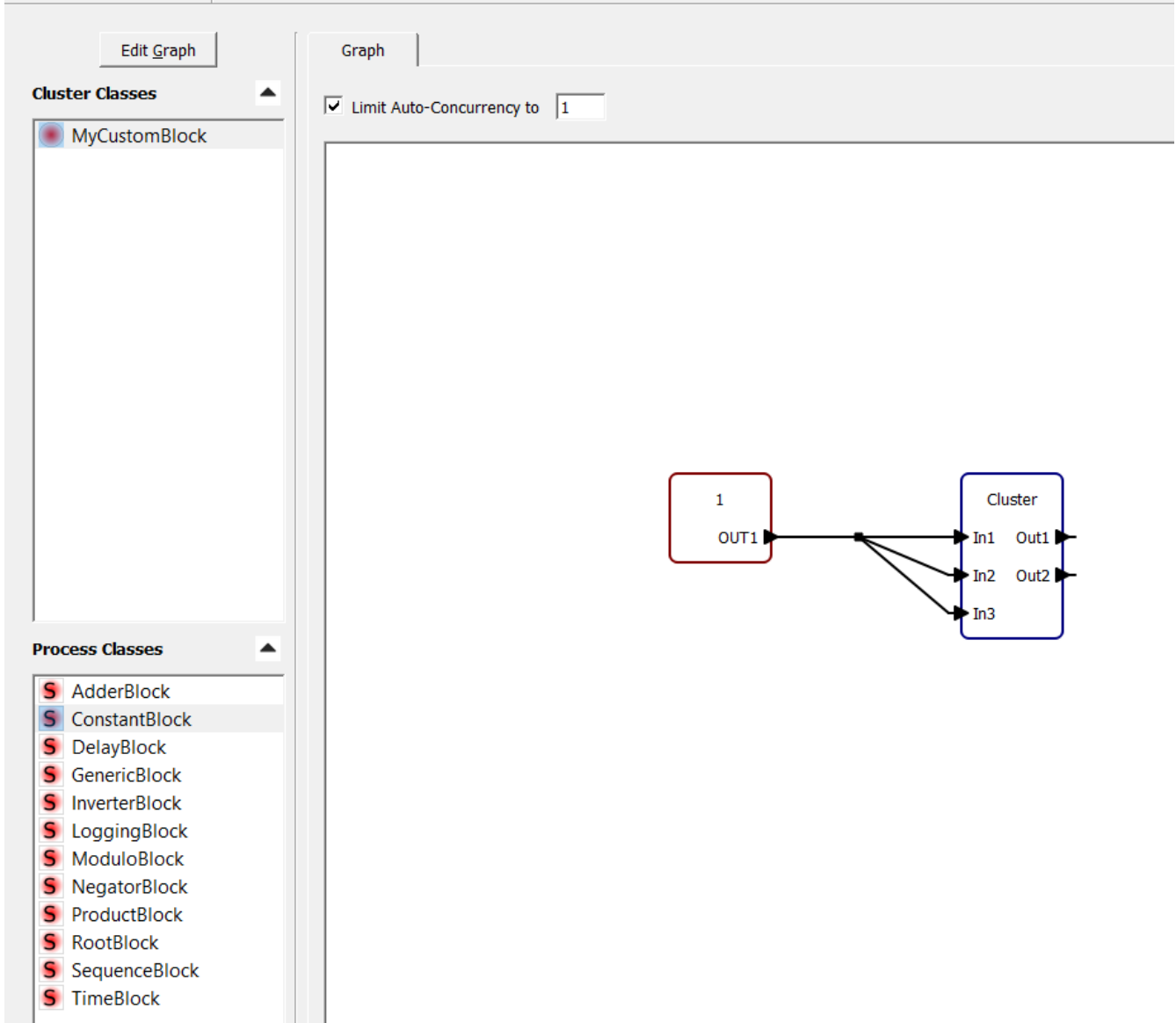
The `MyCustomBlock` cluster should now be available to use.

 Dataflow Modeller - SADF - MyCBDWithCustomBlock.dfm

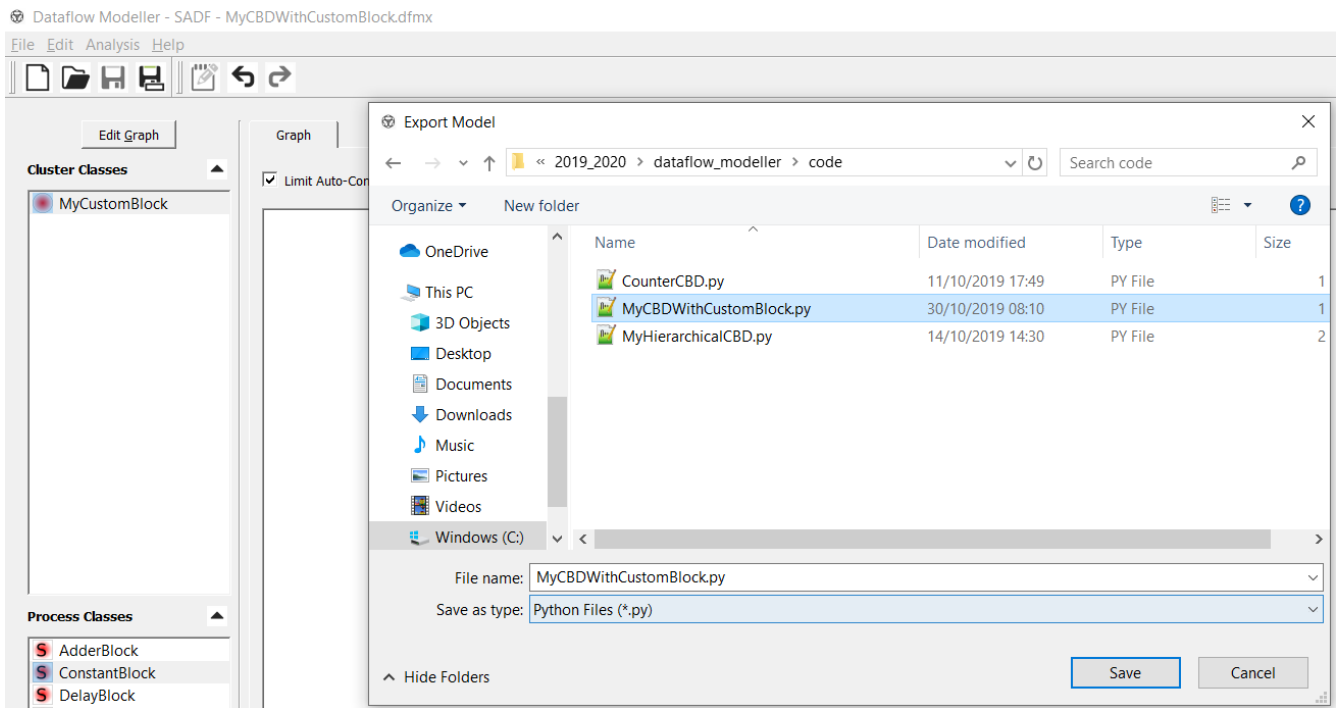


**Step 5. Import the CBDLibrary classes and create your cbd.**





**Step 6. Generate cbd code. Choose File -> Export -> Causal Block Diagrams, and generate the MyCBDWithCustomBlock.py file:**



## Step 7. Replace the generated `MyCustomBlock` class with your own code:

```
MyCBDWithCustomBlock.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
MyCBDWithCustomBlock.py
1  #!/usr/bin/env python
2  from CBDLibrary import *
3  from CBDDraw import *
4
5  class MyCustomBlock(CBD):
6      def __init__(self, block_name):
7          CBD.__init__(self, block_name, input_ports=["In1", "In2", "In3"], output_ports=["Out1", "Out2"])
8
9  class Graph(CBD):
10     def __init__(self, block_name):
11         CBD.__init__(self, block_name)
12         self.addBlock(ConstantBlock(block_name="1", value=float(1)))
13         self.addBlock(MyCustomBlock(block_name="Cluster"))
14         self.addConnection("1", "Cluster", input_port_name="In1", output_port_name="OUT1")
15         self.addConnection("1", "Cluster", input_port_name="In2", output_port_name="OUT1")
16         self.addConnection("1", "Cluster", input_port_name="In3", output_port_name="OUT1")
17
18     CBD = Graph("CBD")
19
```