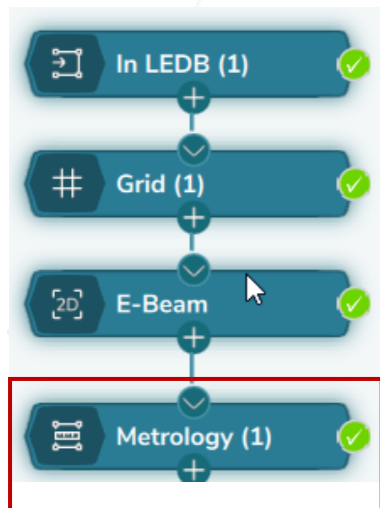


# BEAMER

Loops, Variables, Functions

Variable handling is a key factor for effective and flexible working inside of flows for pattern processing.



**Metrology Results**

Measurement Line  
R1, 0.500, CD Measurement [um] - Line R1, Layer 0.500, x...

x1 = -0.1000, y1 = 0.0000, x2 = 0.1000, y2 = 0.0000  
R1, 0.500, CD Measurement [um] - Line R1, Layer 0.500  
0.199800

Close

**Simulation - E-Beam**

Model Settings

Result Settings  
Archive   Gaussian Approximat...   Numerical PSF

Advanced

Alpha [um]   Beta [um]   Eta  
0.005000   30.000000   0.600000

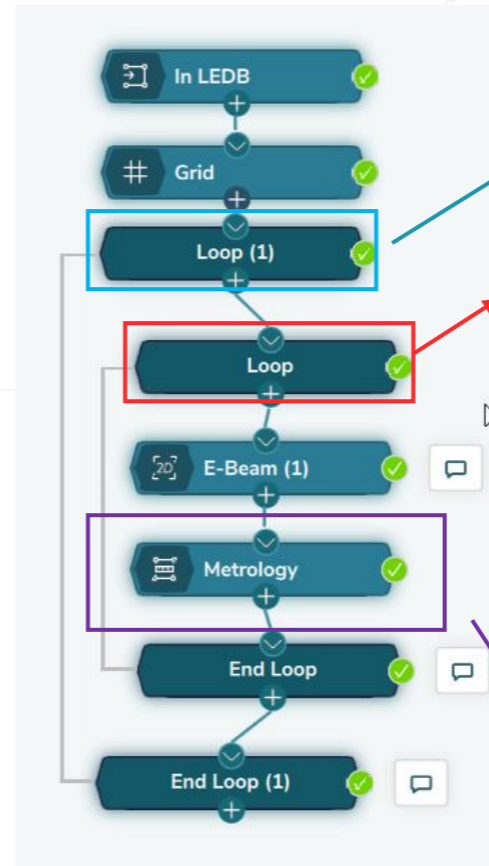
Exposure Dose: 1   Effective Blur FWHM [um]: 0.010000

Periodic X Simulation   Periodic Y Simulation

Include Shot Simulation

**%dose%**   **%blur%**

OK   Cancel

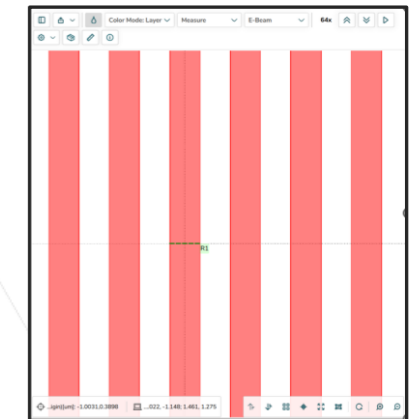


**%blur%**

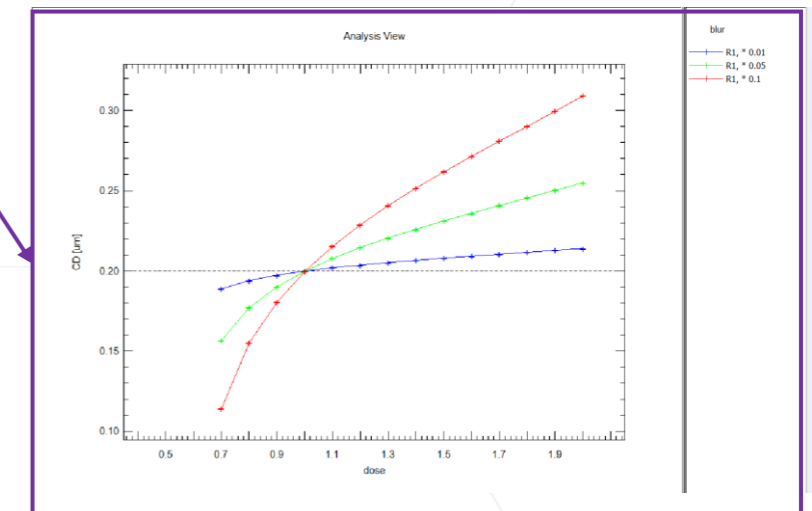
<input checked="" type="checkbox"/>	0.01
<input checked="" type="checkbox"/>	0.05
<input checked="" type="checkbox"/>	0.1

**%dose%**

<input checked="" type="checkbox"/>	0.5
<input checked="" type="checkbox"/>	0.6
<input checked="" type="checkbox"/>	0.7
<input checked="" type="checkbox"/>	0.8
<input checked="" type="checkbox"/>	0.9
<input checked="" type="checkbox"/>	1
<input checked="" type="checkbox"/>	1.1
<input checked="" type="checkbox"/>	1.2
<input checked="" type="checkbox"/>	1.3
<input checked="" type="checkbox"/>	1.4
<input checked="" type="checkbox"/>	1.5
<input checked="" type="checkbox"/>	1.6
<input checked="" type="checkbox"/>	1.7
<input checked="" type="checkbox"/>	1.8
<input checked="" type="checkbox"/>	1.9
<input checked="" type="checkbox"/>	2



50% L/S pattern



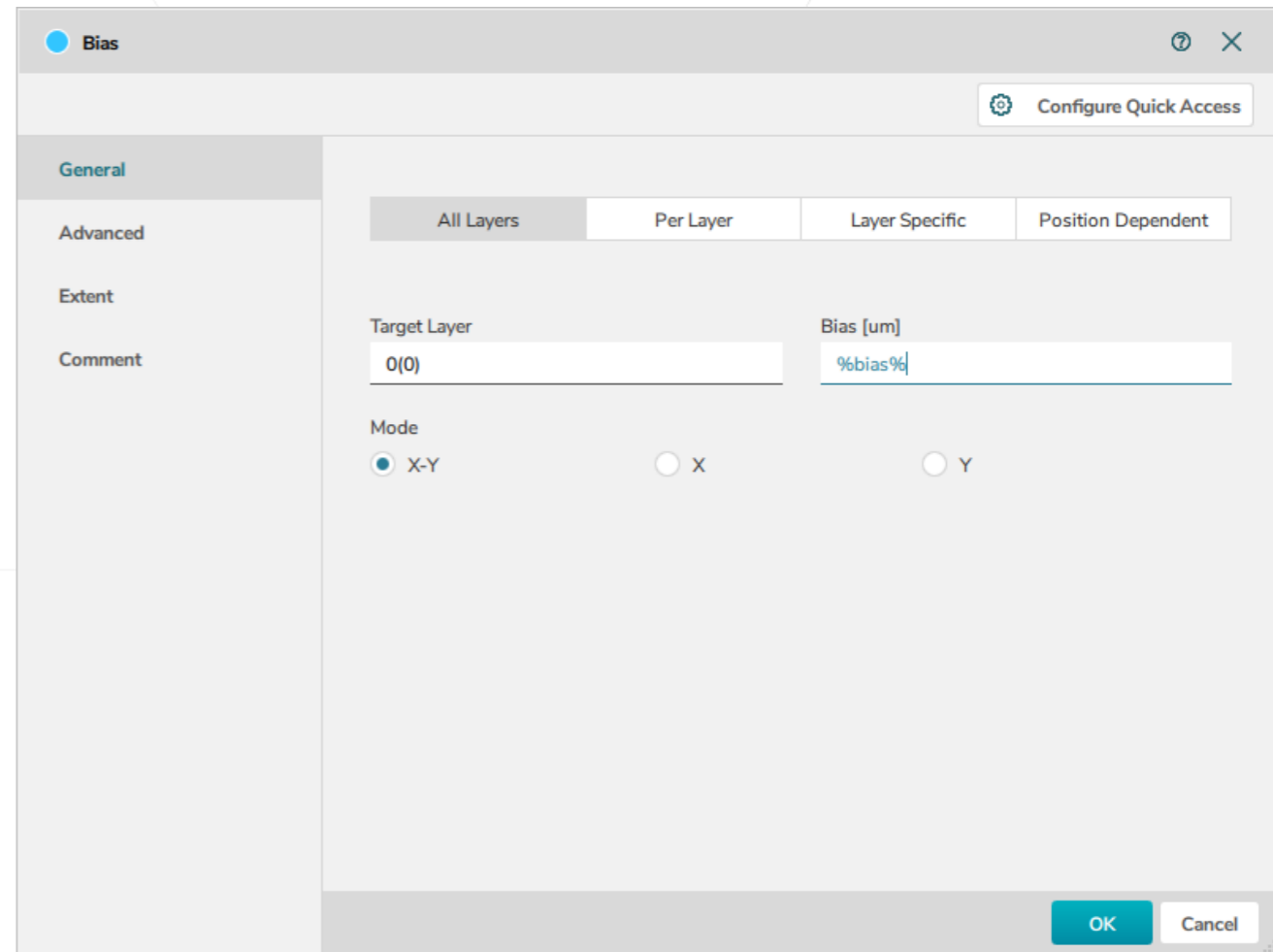
E-Beam simulation with metrology

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Variables can be used in the GUI at any location that allows text input.

Examples:

- Exposure dose
- Effective blur
- Bias
- Target layer
- Field size
- Path for the IMPORT module or EXPORT module
- ...



- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

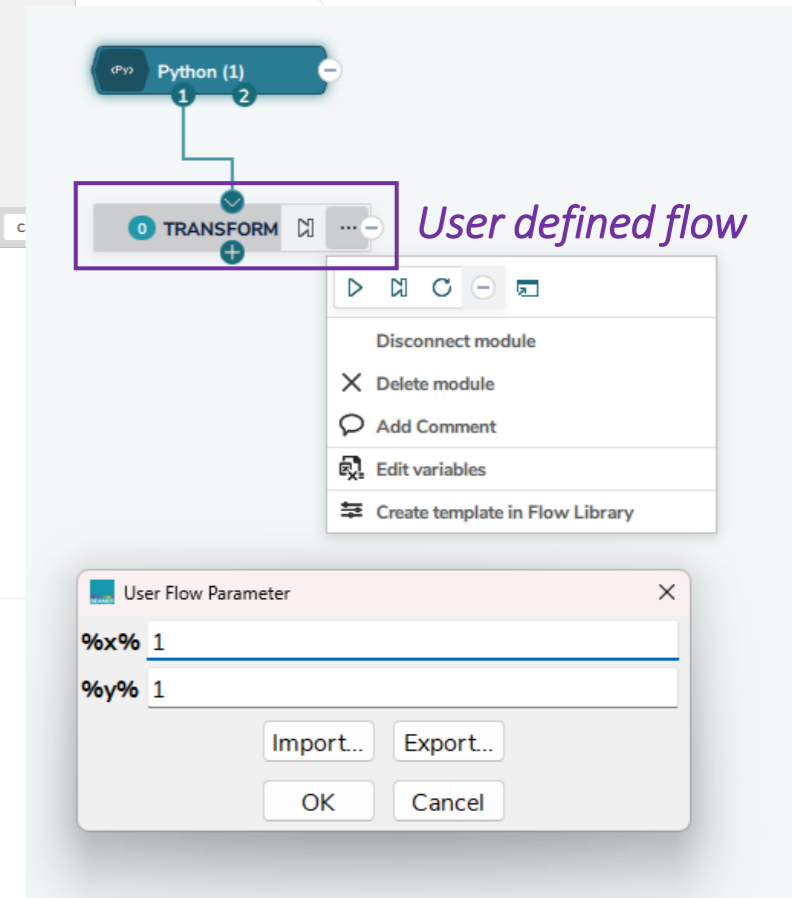
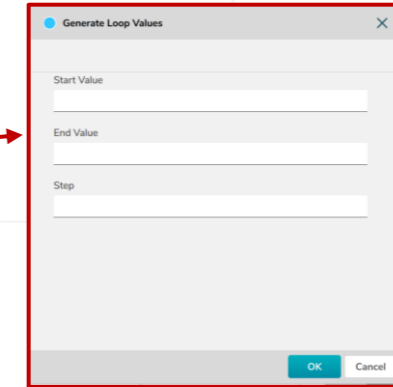
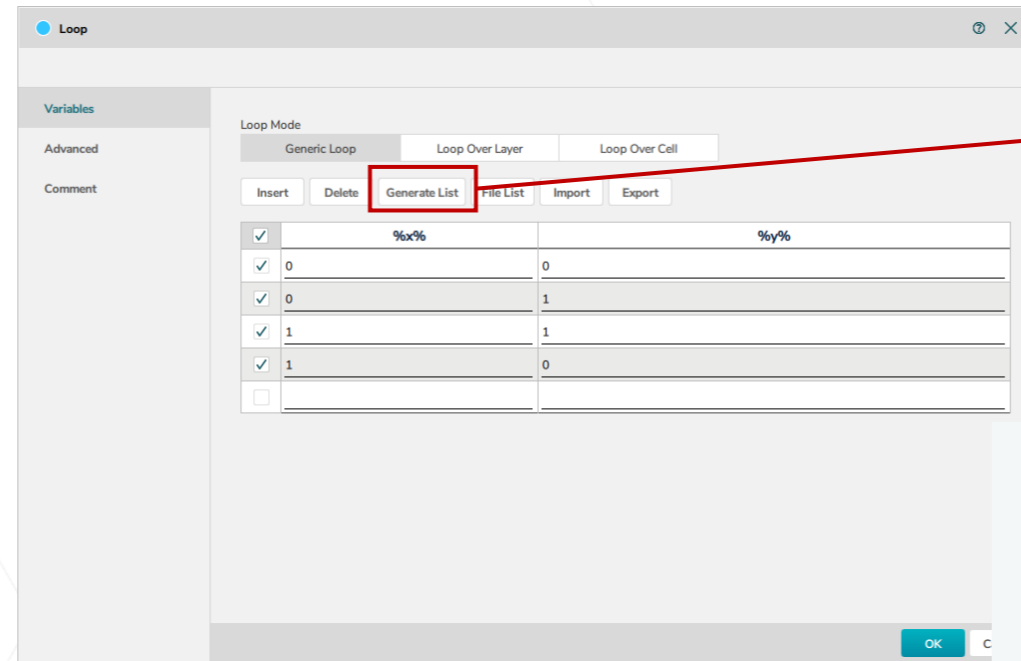
- Variables are defined by a name enclosed within ,%' symbols at at the start and end of the name. (%dose%, %x%...)
- Restricted names are:
  - %IMPORT%  
This holds the full path plus file name of the last IMPORT module in the flow.
  - %EXPORT%  
This holds the full path plus file name of the last EXPORT module in the flow.
  - %EnvVar\_<name>%  
Allows the reading of a system environmental variable by the name of <name>.

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

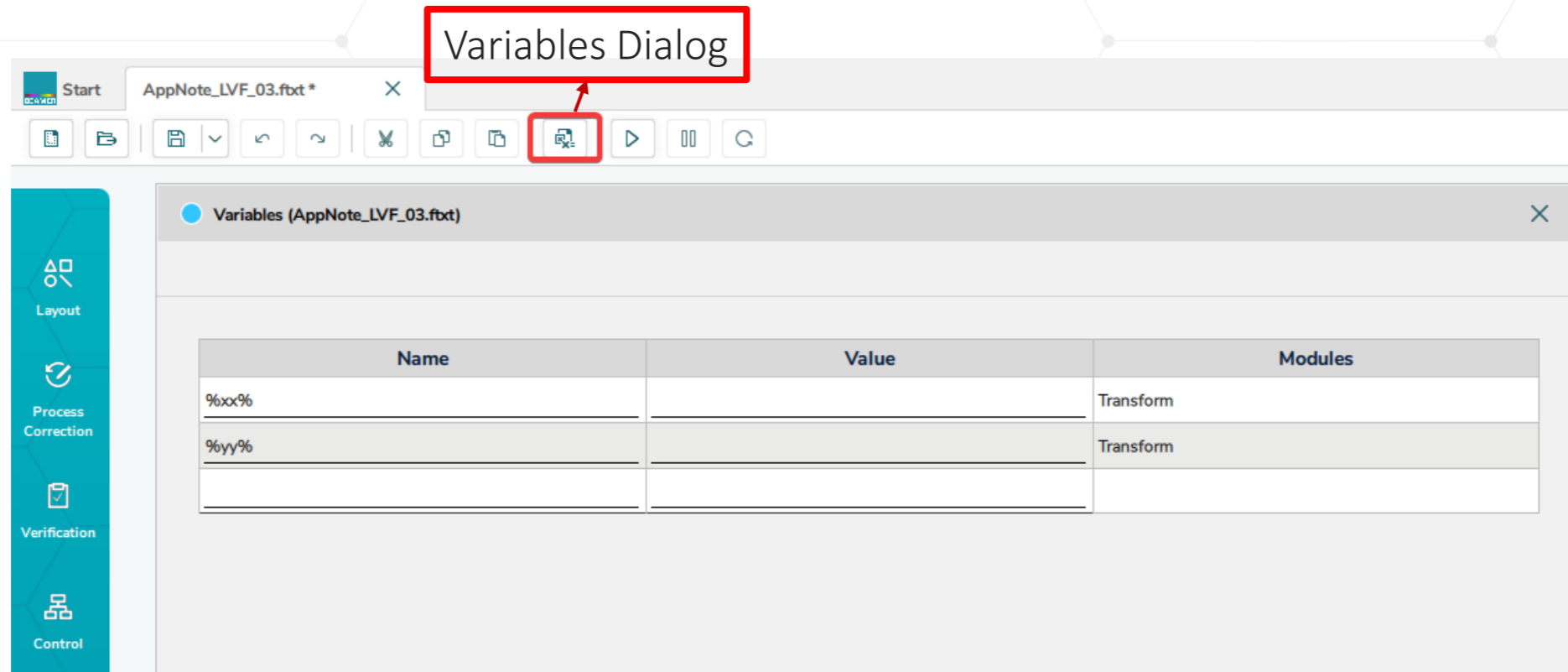
- Local variables
  - Variables defined **within loops or a *User defined flow***.
  - Valid only within the modules of the loops or the User defined flow, and can't be used outside.
  - Local variable takes dominance over the globally defined one, meaning you can overwrite global variables with local variables.
- Global variables
  - Variables defined **outside of loops and *User defined flows***.
  - Can be used in the flow as often as desired.
  - Only change the value once and be applied in multiple places.



- Defined in loop:
  - A Loop module setup allows the definition of variables as single or coupled setup.
  - Each row will be executed in sequence.
- Defined in *User defined flow*
  - Get access to the variables via the `,...'` and the `,Edit variables'` menu.



- A global variable interface displays all global variables used within the flow.



The values are easily set in the *Variables Dialog* which also displays the module name where the variable is used.

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Global variables contain by default no value. An input value is needed for execution of the flow, otherwise an error is shown.
- Defaults for variables allow to preload a variable with a specific value and therefore avoid error in the execution of the flow.
- Default definition:  
Simply insert a set of parentheses ‘()’ containing the desired value after the variable's name, but before the final '%’.

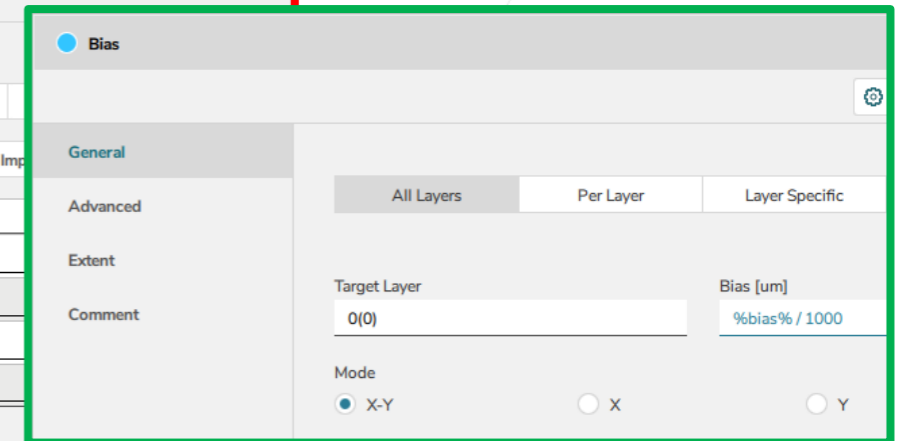
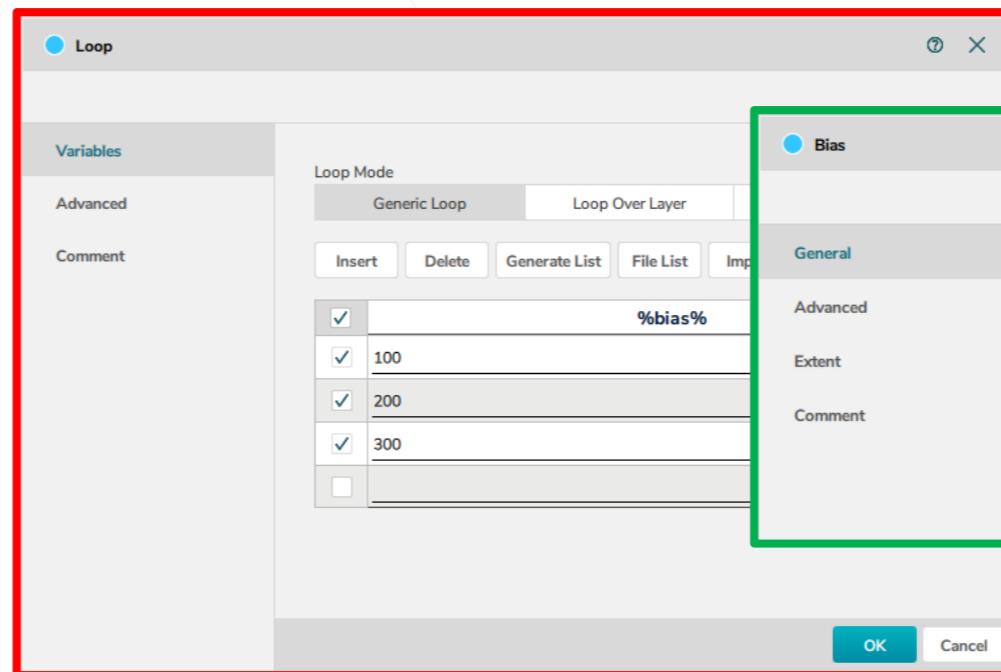
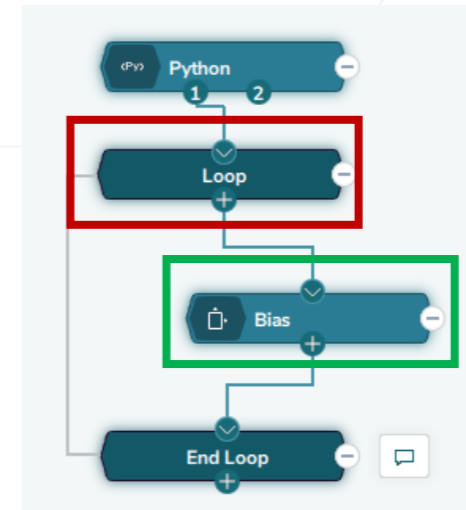
Example: `%name(20)%`

Variable `%name%` is established with a default value of 20

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Math operations with single variable are possible by including the desired formula in the field.
- For instance, if I would like to set my variable in units of nm but the field expects an input of  $\mu\text{m}$ , one can apply the conversion operation (division or multiplication).
- Example:

$\%bias\% / 1000$



- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- String creation can be a simple chain of text and variables:

Example: %x% and %y% indexing a matrix position

Export it as a file with matrix position information:

Samplefile\_posX%x%\_posY%y%.gds

Result: Samplefile\_posX0\_posY0.gds,

Samplefile\_posX0\_posY1.gds,

...



- Example: %x% and %y% indexing a matrix position  
Adding an offset to the position (100, 200...)
- In this case some math operation would be needed, the syntax is **%( )%** wrapping the formula.

Export it as a file with matrix position information:

Samplefile\_posX**%(bias%+%x%)**\_posY**%(bias%+%y%)**.gds

Result: Samplefile\_posX**100**\_posY**100**.gds,  
Samplefile\_posX**100**\_posY**101**.gds,

...

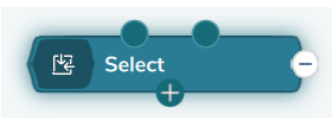
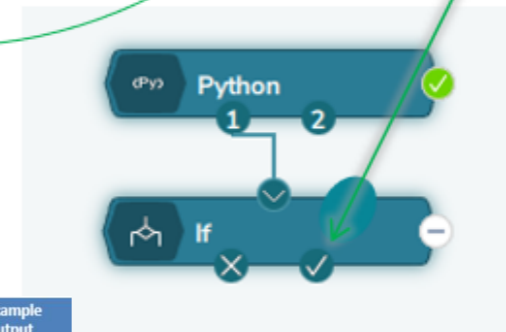
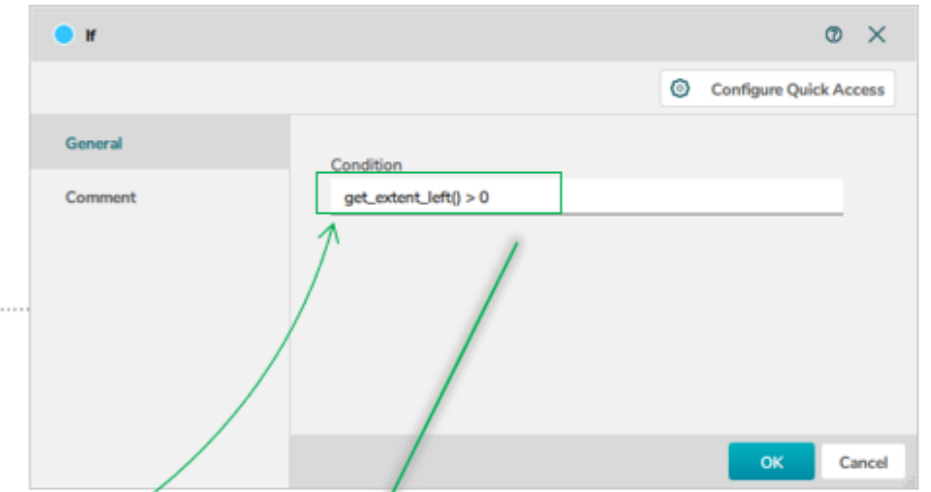
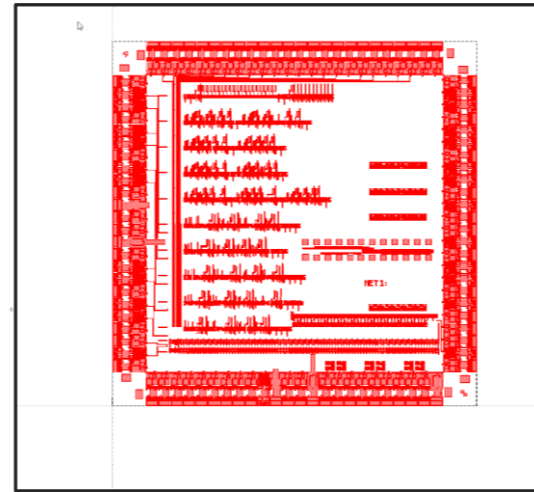
- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- In some cases, getting information from the environmental variables of the operating system can be helpful.
- Application case: Store a specific user or project name in a variable and append this to file names.
- **%EnvVar\_<name>%**: Allows the reading of a system environmental variable by the name of <name>.
- Preset system variables:  
ProjectName: VariableDemo  
UserName: Ritter
- Sample:  
**%EnvVar\_ProjectName%\_%EnvVar\_UserName%\_myproject\_01.gds**

VariableDemo\_Ritter\_myproject\_01.gds

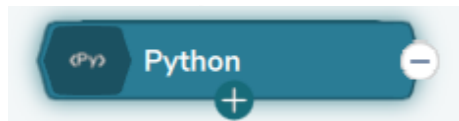
- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- The IF and SELECT modules support functions for decision-based flow.
- These functions use a Python-orientated syntax to read information from layout.
- Following function calls can be integrated into the formula.



Information	Visual Flow Query (IF module)	Python	Example Output
Pattern area (um <sup>2</sup> )	<code>get_area()</code>	<code>b.get_area(imp1)</code>	23423.5
Figure count	<code>get_number_figures()</code>	<code>b.get_...</code>	
Vertex count	<code>get_number_vertices()</code>	<code>b.get_...</code>	
Dose Range	<code>get_min_dose()</code> <code>get_max_dose()</code>	<code>b.get_...</code> <code>b.get_...</code>	Dose(s) assigned at coordinate (x,y)
Assigned dose count	<code>get_number_doseclasses()</code>	<code>b.get_...</code>	Element (overlap) count at (x,y)
Data Layer Count	<code>get_number_layer()</code>	<code>beamer...</code>	
Cell Count	<code>get_number_cells()</code>	<code>beamer...</code>	
Data Extents	<code>get_extent_left()</code> <code>get_extent_right()</code> <code>get_extent_top()</code> <code>get_extent_bottom()</code>	<code>b.get_...</code> <code>b.get_...</code> <code>b.get_...</code> <code>b.get_...</code>	Layer Existence Layout Existence
Top Cell Count	<code>get_number_topcells()</code>	<code>b.get_...</code> <code>n &lt; get_number_cells()</code>	Name of cell n <code>get_cell(n)</code>
Database Unit (um)	<code>get_database_units()</code>	<code>b.get_...</code> <code>n &lt; get_number_layer()</code>	Layer Specification n <code>get_layer(n)</code>

- *Python* module provides the tools for the perfect symbiosis with **BEAMER**



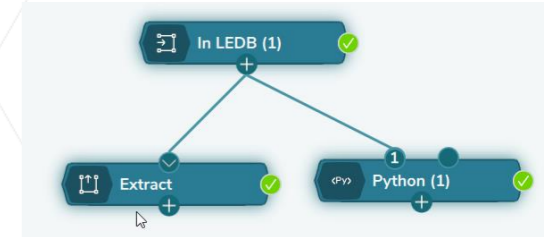
- ▷ Modules
- ▷ **Evaluations**
- ▷ Interface
- ▷ LayoutPy
- ▷ Math
- ▷ Examples
- ▷ Snippets
- ▷ History

- ▽ **Evaluations**
  - Check Layout
  - Check Layer
  - Check Cell
  - Get Area
  - Get number of figures
  - Get number of vertices
  - Get minimum dose
  - Get maximum dose
  - Get number of dose classes
  - Get number of layers
  - Get name of layer n
  - Get number of cells
  - Get name of cell n
  - Get left extent position
  - Get bottom extent position
  - Get right extent position
  - Get top extent position
  - Get number of top cells
  - Get database unit
  - Get dose at position
  - Get number of elements at position

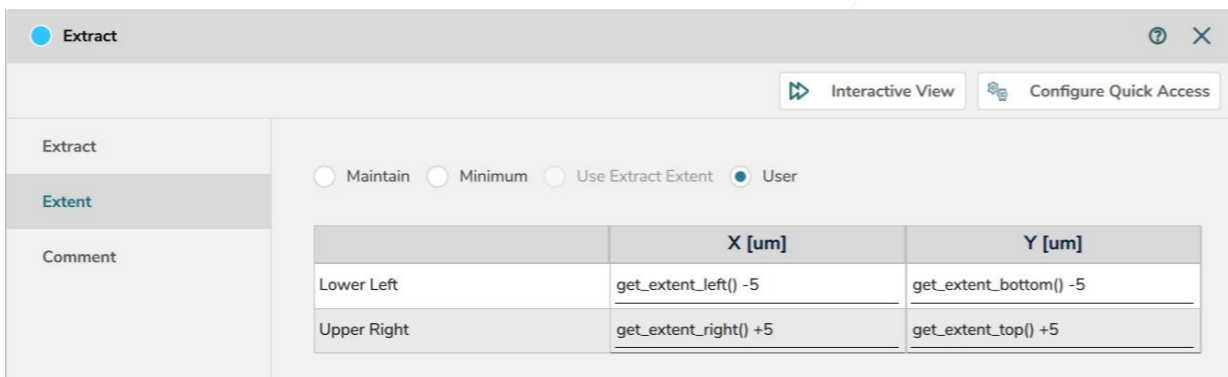
Commands that allow interaction  
with the layout

## • Comparison Python with GUI

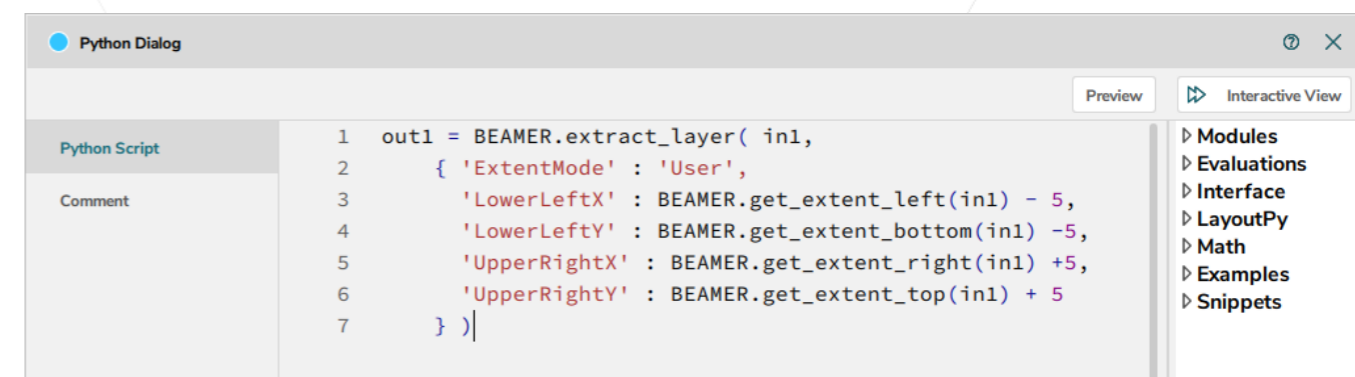
- Grow the extent of the current pattern bounding box by 5 $\mu$ m in each direction. This should be independent of the pattern coming in and therefore should work for all layouts without the user needing to inspect the pattern extent and adjust.



### GUI



### Python



- Python module provides the tools for the perfect symbiosis with **BEAMER**



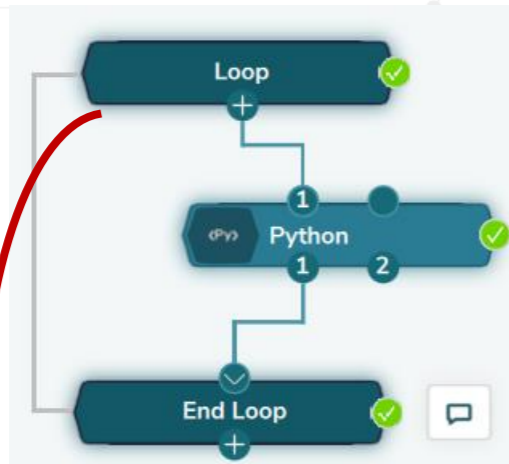
- ▷ Modules
- ▷ Evaluations
- ▷ **Interface**
- ▷ LayoutPy
- ▷ Math
- ▷ Examples
- ▷ Snippets
- ▷ History

- ▽ **Interface**
  - Variables: Read
  - Variables: Set existing
  - Variables: Set single
  - Variables: Set multiple

Commands that allow interaction with the layout and using variables as input in the Python environment. They allow you to read, create or change values of the variables which is defined in Beamer GUI.



- Variables can be used within the Python module environment



```

Python Dialog
Python Script
Comment
1 from LAYOUTpy import *
2 import os
3 import sys
4
5 variables = BEAMER.get_variable('%VarN%')
6 layer     = BEAMER.get_variable('%layer%')
7
8 x = 500
9 y = 500
10
11 radius = float(variables)
12 lay    = int(layer)
13 circle = Circle( (x, y), radius, layer=lay)
14
15 db = Database()
16 with Transaction(db) as txn:
17     txn.insert(circle)
18
19 db.close()
20 out1 = db.togobject()
    
```

Loop

Variables

Advanced

Comment

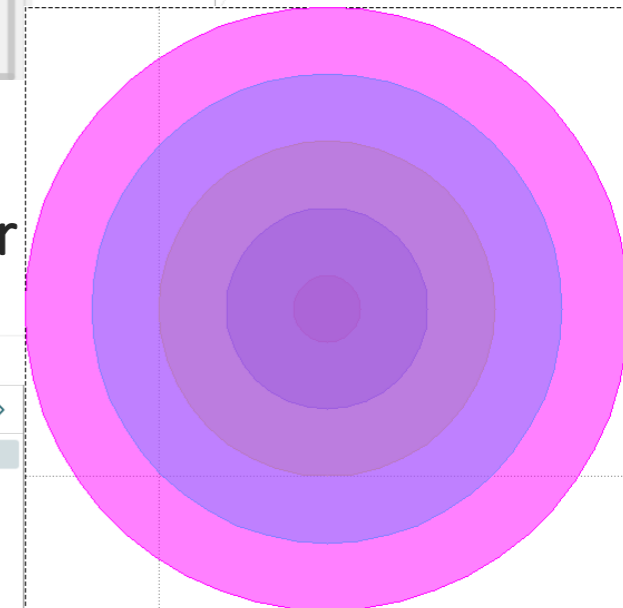
Loop Mode

Generic Loop | Loop Over Layer | Loop Over Cell

Insert | Delete | Generate List | File List | Import | Export

<input checked="" type="checkbox"/>	%VarN%	%layer%
<input checked="" type="checkbox"/>	100	1
<input checked="" type="checkbox"/>	300	3
<input checked="" type="checkbox"/>	500	5
<input checked="" type="checkbox"/>	700	7
<input checked="" type="checkbox"/>	900	9

Variables are created in a *Loop* module and used as **radius** and **layer** values for a set of concentric circles



- You can find some useful information in the AppNote under:  
<https://www.genisys-gmbh.com/applications/loops-variables-and-functions.html>



The screenshot shows a web browser displaying the GenISys website. The page title is "Loops, Variables and Functions". The content includes a "Motivation" section explaining that processing layouts or a large number of BEAMER flows often requires the modification of parameters in specific locations of the flow. A "Solution" section states that BEAMER provides various options for the use of variables within flows, which can be defined as local or global variables. A table of contents is provided on the left side of the page. On the right side, there is a screenshot of the BEAMER software interface showing a "User Flow Parameter" dialog box with fields for "Name" and "Value".

**Loops, Variables & Functions**

Variable handling is a key factor for effective and flexible working inside of flows and doing pattern processing. With this note we will explore the various options inside BEAMER for variables and how they can help in a day-to-day work environment.

**Content**

- Loops, Variables & Functions ..... 1
- Places for variables ..... 2
- Definitions of variables ..... 2
- Global and Local variables ..... 2
- Default values ..... 4
- Math with a single variable ..... 4
- Math with multiple variables ..... 6
- Reading environmental variables ..... 7
- Functions as Inputs ..... 7

**Loops, Variables and Functions**

**Motivation:**

The processing of layouts or a large number of BEAMER flows often requires the modification of parameters in specific locations of the flow. Variable handling is a critical factor for efficient and flexible work within flows and pattern processing.

**Solution:**

BEAMER provides various options for the use of variables within flows. Variables can be defined as local or global variables, depending on their location. Additionally, variables can be used in mathematical formulas or to read environmental parameters. They are beneficial in a daily work environment.

By using variables, BEAMER users can create more flexible and reusable flows. This can save time and effort, and it can also improve the accuracy and reliability of the results.

**User Flow Parameter Dialog:**

Name: %x% 1  
Value: %y% 1  
Buttons: Import, Export, OK, Cancel

# Thank You!

support@genisys-gmbh.com

## Headquarters

GenISys GmbH  
Eschenstr. 66  
D-82024 Taufkirchen (Munich)  
GERMANY

📞 +49-(0)89-3309197-60

📠 +49-(0)89-3309197-61

✉ info@genisys-gmbh.com

## USA Office

GenISys Inc.  
P.O. Box 410956  
San Francisco, CA  
94141-0956  
USA

📞 +1 (408) 353-3951

✉ usa@genisys-gmbh.com

## Japan / Asia Pacific Office

GenISys K.K.  
German Industry Park  
1-18-2 Hakusan Midori-ku  
Yokohama 226-0006  
JAPAN

📞 +81 (0)45-530-3306

📠 +81 (0)45-532-6933

✉ apsales@genisys-gmbh.com

