Python API
Code Conversion Guidelines
L-2016.06

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

CONTENTS

Python API Version 4.1 Code Conversion Guidelines

Introduction

Enhancements to the Python API include both new additions that provide improved functionality and also changes to existing classes and methods. These latter changes may require minor updates to existing PyCell Python code before it will work with the 4.1 release.

This document contains a brief description of these incompatible changes between the PyCell Studio 4.0 and 4.1 releases, along with suggestions for how to update existing code written using the 4.0 release. Although no attempt is being made to encode this information in the form of conversion scripts for existing PyCell Python source code, this listing of class method changes should still be used as a guideline for manual conversion.

Existing Python API Class Changes

MultiPath class

The MultiPath class has been substantially extended, so that it now provides a much richer set of capabilities than in previous releases. The MultiPath object now provides a use model where any optional subpath is defined by means of both the justification from the master path, along with the separation between the point list for the master path and the point list for the subpath. This is in contrast to the model used in previous PyCell Studio releases, where the optional offset subpath was defined by a single offset value, and the optional enclosure subpath was defined by a single expansion value.

In addition to providing more methods for this MultiPath class, existing methods now have several additional parameters. Specifically, the creation method as well as the createOffsetSubpath() and createEnclosureSubpath() methods now have several additional parameters, which were not available in the previous PyCell Studio 4.0 release. Thus, if any of these three methods were used without named keyword parameters, then it will be required to modify the parameters for these methods. In addition, the set of parameter values should be examined to convert the subpath definition from a single offset or expansion value to the justification and separation parameter values now used by the MultiPath class.

ContactRing class

The ContactRing class has been enhanced, so that multiple fill layers can be specified when a contact ring is created, instead of just a single fill layer. For example, this capability can be used to create a contact ring with both Nwell and thick oxide fill layers. Thus, the keyword "fillLayer" has been changed to "fillLayers", and this keyword now specifies a list of fill layers, instead of only a single fill layer. In order to change existing code, this keyword parameter value should be changed, and the single fill layer value should be specified as a list containing a single fill layer.

RangeConstraint class

The RangeConstraint class has been enhanced, so that an optional resolution parameter can now be specified. This enhancement makes the StepConstraint and RangeConstraint classes consistent in their functionality as well as in their parameter calling sequence. In some cases, it may be necessary to change existing code. This will be the case, if the third parameter for the RangeConstraint creation call specifies the optional action type, without the use of a named keyword parameter. In such cases, either use keyword parameters to specify values, or include this new resolution parameter as the third parameter value, prior to the specification of the action type.