
PyCell4Klayout

Release 0.1

IHP Authors

Apr 16, 2024

CONTENTS:

| | | |
|----------|-------------------------------------|-----------|
| 1 | Referenced documents | 2 |
| 2 | Introduction | 3 |
| 2.1 | Basic Structure | 3 |
| 3 | Implementation Details | 4 |
| 3.1 | Klayout | 4 |
| 3.2 | Synopsys PyCell API | 4 |
| 3.3 | Wrapper solution approach | 5 |
| 4 | API Reference | 6 |
| 4.1 | font | 6 |
| 4.2 | namemapper | 7 |
| 4.3 | box | 7 |
| 4.4 | grouping | 10 |
| 4.5 | shape | 11 |
| 4.6 | dlo | 11 |
| 4.7 | ulist | 12 |
| 4.8 | location | 13 |
| 4.9 | termtype | 13 |
| 4.10 | point | 14 |
| 4.11 | tech | 15 |
| 4.12 | orientation | 16 |
| 4.13 | constants | 16 |
| 4.14 | pathstyle | 17 |
| 4.15 | pointlist | 17 |
| 4.16 | dlogen | 18 |
| 4.17 | transform | 18 |
| 4.18 | rect | 19 |
| 4.19 | geo | 19 |
| 4.20 | polygon | 20 |
| 4.21 | numeric | 21 |
| 4.22 | signaltype | 22 |
| 4.23 | text | 23 |
| 4.24 | layer | 23 |
| 4.25 | physicalComponent | 24 |
| 5 | Indices and tables | 25 |
| | Python Module Index | 26 |

PyCell4Klayout is a Python library for supporting the PyCell API under the layout tool [Klayout](#).

Note: This project is under active development.

REFERENCED DOCUMENTS

| No. | Doc ID-Number | Title |
|-----|---------------|--|
| [1] | 2021.09 | Synopsys 'Python API Reference Manual' |
| | | |
| | | |
| | | |

INTRODUCTION

This reference manual documents the PyCell4Klayout API (Application Programming Interface) which is used to create parameterized cells within the [Klayout](#) design environment. This Klayout design environment makes use of the popular open-source Python programming language to provide a highly productive design environment for creating parameterized cells for analog layout design purposes. This PyCell4Klayout API provides a large number of classes and methods which are specialized for layout designs. By using these Python classes to provide powerful, high-level layout design abstractions, the Klayout design environment is extremely productive.

2.1 Basic Structure

The basic PyCell4Klayout system is built upon a set of base classes, from which the basic design and layout objects are generated through the PyCell Python API. These base classes are made accessible through the PyCell Python API, but do not have their own creation methods. Instead, other objects which are derived from these base classes can be constructed through the use of the PyCell Python API.

IMPLEMENTATION DETAILS

3.1 Klayout

Klayout is an open-source EDA layout tool with a rich set of functionalities like layout editing, DRC, LVS, PCells, scripting and so on. The feature set and the Klayout GUI are implemented in an object oriented C++ core using the Qt library. Klayout supports the programming languages Ruby and Python for scripting. The overall principle is that most of the C++ core classes have pendants with the same name in both the Ruby and Python scripting world. In the scripting world these pendants are just proxies which delegates an API-call to the C++ pendant which implements the API-call (language binding). This mechanism also covers the needed object lifetime management as well as the parameter type conversion back- and forward in the both directions between C++- and scripting-world (Marshalling).

3.1.1 PCell support

Klayout uses an own schema for supporting parametrized cells. The creation of PCells are covered in principle by the C++ core class PCellDeclaration. This class is also available in the Klayout Python namespace. The class PCellDeclaration as a PCell base class defines an API with some virtual function which must be implemented by a subclass to build a new PCell. This implemented functions are then called by the Klayout runtime engine to create/manage a new PCell. The most important of these functions are:

- *get_parameters*: Returns a list of parameter declarations for the PCell
- *coerce_parameters*: Modifies the parameters to match the requirements of the PCell
- *produce*: The production callback which creates the PCell layout

One method to create a PCell is subclassing a new class from PCellDeclaration and implement this set of specific function. This can be done in a Ruby- as well as Python-script.

3.2 Synopsys PyCell API

The PyCell API is build by a hierarchy of Python classes with a defined API [1]. The PyCell API classes can be roughly divided into the following groups:

- Basic geometry classes: Point, Box, Segement, Font, ...
- Physical component classes: Shape, Arc, Line, Dot,...
- Physical component related classes: Contact, AbutContact, Bar, DeviceContact, Via, ...
- Physical component reference classes: GroupingRef, InstanceRef, PolygonRef, ...
- Connectivity classes: SignalType, TermType, Net, Term, Pin, Layer, Tech, ...
- Parameter classes: ParamArray, ChoiceConstraint, RangeConstraint, ...

- PCell creation classes: Dlo, DloGen, Lib

3.2.1 PyCell creation principle

The DloGen class is the base class for all types of PCell generators. Any PCell generator would be derived from this base class. A PCell generator class which is derived from the DloGen base class must implement the following methods:

- *defineParamSpecs*: defines the parameters, including default values and constraints, for this PyCell
- *setupParams*: extracts the value for the parameters specified by the user for this PyCell
- *genLayout*: generates the actual physical layout for this PyCell

3.3 Wrapper solution approach

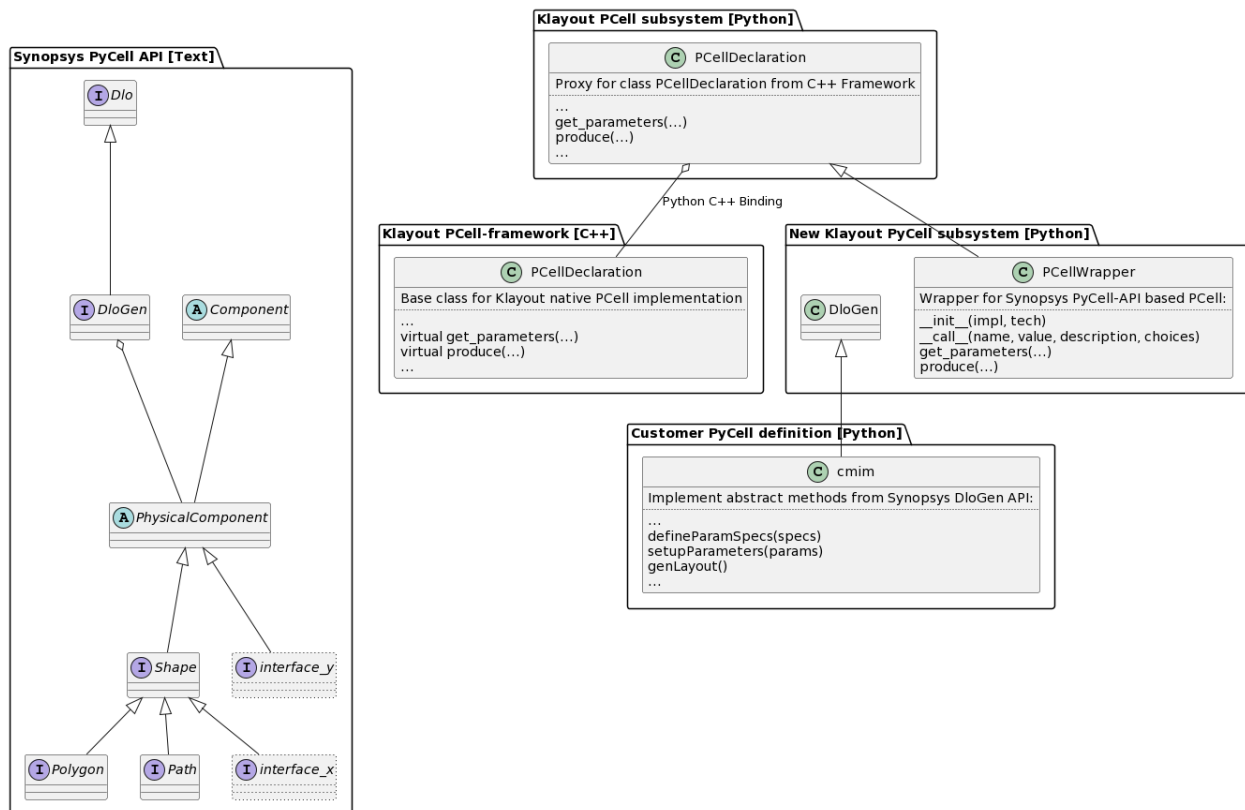


Fig. 1: PyCell wrapper structural architecture overview

API REFERENCE

This page contains auto-generated API reference documentation¹.

4.1 font

4.1.1 Module Contents

Classes

Font

```
class font.Font
    Bases: object
    EURO_STYLE = 1
    FIXED = 2
    GOTHIC = 3
    MATH = 4
    MIL_SPEC = 5
    ROMAN = 6
    SCRIPT = 7
    STICK = 8
    SWEDISH = 9
    classmethod getMembers()
    calcBBox(text, origin, height, location=Location.UPPER_LEFT, orient=Orientation.R0, overbar=False)
```

¹ Created with sphinx-autoapi

4.2 namemapper

4.2.1 Module Contents

Classes

NameMapper

```
class namemapper.NameMapper(obj: object = None)
    Bases: object
```

4.3 box

4.3.1 Module Contents

Classes

Box

```
class box.Box(l=INT_MAX, b=INT_MAX, r=INT_MIN, t=INT_MIN)
    Bases: object
    property bottom
    property left
    property right
    property top
    abut(refBox, align=True)
    alignEdge(refBox, refDir=None, offset=None)
    alignEdgeToCoord(coord)
    alignEdgeToPoint(point)
    alignLocation(refBox, refLoc=None, offset=None)
    alignLocationToPoint(pt)
    centerCenter()
    centerLeft()
    centerRight()
```

clone(*nameMap: cni.namemapper.NameMapper = NameMapper(), netMap: cni.namemapper.NameMapper = NameMapper()*)

contains(*incEdges=True*)

containsPoint(*incEdges=True*)

destroy()

expand()

expandDir(*coord*)

expandForMinArea(*minArea, grid=None*)

expandForMinWidth(*minWidth, grid=None*)

expandToGrid(*dir=None*)

fix()

getArea()

getCenter()

getCenterX()

getCenterY()

getCoord()

getDimension()

getHeight()

getLeft()

getLocationPoint()

getLocationPoint()

getPoints()

getRange()

getRangeX()

getRangeY()

getRight()

getSpacing(*refBox*)

getTop()

getWidth()

hasNoArea()

init()

intersect()
intersect(*dir*)
isInverted()
isNormal()
limit()
lowerCenter()
lowerLeft()
lowerRight()
merge(*dir*)
mergePoint()
mirrorX()
mirrorY()
moveBy(*dx: float, dy: float*) → None
moveTo(*loc=Location.CENTER_CENTER*)
moveTowards(*d*)
overlaps(*incEdges=True*)
place(*refBox, distance, align=True*)
removeRegion()
rotate90()
rotate180()
rotate270()
set()
set(*dir=None*)
set(*upperRight*)
set(*bottom, right, top*)
setBottom()
setCenter()
setCenterY()
setCoord(*coord*)
setDimension(*dir*)
setBottom()

```

setHeight()
setLocationPoint(pt)
setRange(range)
setRangeX()
setRangeY()
setRect(rect)
setRight()
setTop()
setWidth()
snap(snapType=None)
snapX(snapType=None)
snapY(snapType=None)
snapTowards(dir)
transform(transform: cni.transform.Transform) → None
upperCenter()
upperLeft()
upperRight()

```

4.4 grouping

4.4.1 Module Contents

Classes

| | |
|-----------------|--|
| <i>Grouping</i> | Helper class that provides a standard way to create an ABC using |
|-----------------|--|

```

class grouping.Grouping(name: str = "", components: cni.physicalComponent.PhysicalComponent = None)
    Bases: cni.physicalComponent.PhysicalComponent
    Helper class that provides a standard way to create an ABC using inheritance.
    add(components: cni.physicalComponent.PhysicalComponent) → None
    addToRegion(region: pya.Region)
    clone(nameMap: cni.physicalComponent.NameMapper = NameMapper(), netMap: cni.physicalComponent.NameMapper = NameMapper())

```

```

destroy()
getComps() → list
getComp(index: int) → cni.physicalComponent.PhysicalComponent
moveBy(dx: float, dy: float) → None
toString()
transform(transform: cni.physicalComponent.Transform) → None

```

4.5 shape

4.5.1 Module Contents

Classes

| | |
|--------------|--|
| <i>Shape</i> | Helper class that provides a standard way to create an ABC using |
|--------------|--|

```

class shape.Shape(bbox=None)
    Bases: cni.physicalComponent.PhysicalComponent
    Helper class that provides a standard way to create an ABC using inheritance.
    cell
    set_shape(shape: Shape)
    getShape()
    getBBox()

```

4.6 dlo

4.6.1 Module Contents

Classes

| | |
|-------------------------|----------------------------|
| <i>ChoiceConstraint</i> | Built-in mutable sequence. |
| <i>RangeConstraint</i> | |
| <i>PyCellContext</i> | |
| <i>PCellWrapper</i> | |

```
class dlo.ChoiceConstraint(choices, action=REJECT)
    Bases: list
    Built-in mutable sequence.
    If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.
class dlo.RangeConstraint(low, high, resolution=None, action=REJECT)
class dlo.PyCellContext(tech, cell)
    Bases: object
    __enter__()
    __exit__(*params)
class dlo.PCellWrapper(impl, tech)
    Bases: pya.PCellDeclaration
    __call__(name, value, description=None, constraint=None)
    get_parameters()
    params_as_hash(parameters)
    display_text(parameters)
    produce(layout, layers, parameters, cell)
```

4.7 ulist

4.7.1 Module Contents

Classes

| | |
|--------------|----------------------------|
| <i>ulist</i> | Built-in mutable sequence. |
|--------------|----------------------------|

Attributes

| |
|----------|
| <i>T</i> |
|----------|

ulist.T

```
class ulist.ulist(items=None)
    Bases: list[T]
    Built-in mutable sequence.
    If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.
    append(item) → None
        Append object to the end of the list.
```

4.8 location

4.8.1 Module Contents

Classes

Location

```
class location.Location
```

```
    Bases: object
```

```
    LOWER_LEFT = 1
```

```
    CENTER_LEFT = 2
```

```
    UPPER_LEFT = 3
```

```
    LOWER_CENTER = 4
```

```
    CENTER_CENTER = 5
```

```
    UPPER_CENTER = 6
```

```
    LOWER_RIGHT = 7
```

```
    CENTER_RIGHT = 8
```

```
    UPPER_RIGHT = 9
```

```
    mirrorX()
```

```
    mirrorY()
```

```
    rotate90()
```

```
    rotate180()
```

```
    rotate270()
```

```
    transform(transform)
```

4.9 termtree

4.9.1 Module Contents

Classes

TermType

```
class termtype.TermType
```

```
    Bases: object
```

```
    INPUT = 1
```

```
    OUTPUT = 2
```

```
    INPUT_OUTPUT = 3
```

```
    SWITCH = 4
```

```
    JUMPER = 5
```

```
    UNUSED = 6
```

```
    TRISTATE = 7
```

4.10 point

4.10.1 Module Contents

Classes

Point

```
class point.Point(x, y)
```

```
    Bases: object
```

```
    property x
```

```
        Returns the value of the x-coordinate for this point
```

```
    property y
```

```
        Returns the value of the y-coordinate for this point
```

```
    classmethod areColinearPoints(p1, p2, p3)
```

```
        Returns True if these three points are colinear or coincident, and returns False otherwise.
```

Parameters

- **p1** (*Point*) – first point.
- **p2** (*Point*) – second point.
- **p3** (*Point*) – third point.

Returns

```
    whether all three points are collinear or coincident
```

Return type

```
    boolean
```

```
    copy()
```

```
    getCoord(dir)
```

```

getSpacing(dir, refPoint)
getX()
getY()
invalid()
isBetween(a, b)
isValid()
place(dir, refPoint, distance, align=True)
set(p)
set(_x, _y)
setCoord(dir, coord)
setX(x)
setY(y)
snap(grid, snapType=None)
snapX(grid, snapType=None)
snapY(grid, snapType=None)
snapTowards(grid, dir)
toDiagAxes()
toOrthogAxes()
transform(trans)
__eq__(other)
    Return self==value.

```

4.11 tech

4.11.1 Module Contents

Classes

TechImpl

Tech

```
class tech.TechImpl
```

```
    Bases: object
```

```
class tech.Tech
    Bases: object
    techsByName
    techInUse =
    register()
    get()
```

4.12 orientation

4.12.1 Module Contents

Classes

Orientation

```
class orientation.Orientation
    Bases: object
    R0 = 0
    R90 = 1
    R180 = 2
    R270 = 3
    MY = 4
    MYR90 = 5
    MX = 6
    MXR90 = 7
    concat(other)
    getRelativeOrient(other)
```

4.13 constants

4.13.1 Module Contents

```
constants.REJECT = 1
constants.ACCEPT = 2
constants.USE_DEFAULT = 3
```

constants.INT_MAX

constants.INT_MIN

4.14 pathstyle

4.14.1 Module Contents

Classes

PathStyle

class pathstyle.PathStyle

Bases: object

TRUNCATE = 1

EXTEND = 2

ROUND = 3

VARIABLE = 4

4.15 pointlist

4.15.1 Module Contents

Classes

PointList

Built-in mutable sequence.

class pointlist.PointList(*items=None*)

Bases: `cni.ulist.ulist[cni.point.Point]`

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

compress(*isClose=True*) → *PointList*

Compresses this PointList, by removing any extra (coincident and/or collinear) points from this PointList. The optional `isClosed` parameter is used to indicate whether this set of points is meant to represent a closed shape or not. If all points are collinear, then the first and last points will be the result of compressing this PointList. If the first and last points are coincident, then only the first point is returned

Parameters

isClose – Whether represented shape is closed

Returns

see description above

Return type*PointList***containsPoint**(*point: cni.point.Point*) → bool

4.16 dlogen

4.16.1 Module Contents

Classes

Dlo

DloGen

class dlogen.**Dlo**(*libName, cellName, viewName='layout', viewType=None*)

Bases: object

classmethod exists(*dloName*) → bool**class** dlogen.**DloGen**Bases: *Dlo***set_tech**(*tech*)**addPin**(*name, label, box, layer*)

4.17 transform

4.17.1 Module Contents

Classes

Transform

class transform.**Transform**(*arg1, arg2, arg3, arg4=None*)

Bases: object

property transform

returns the internal transform representation

property xOffset

returns the x-coordinate value of the offset for this Transform

property yOffset

returns the y-coordinate value of the offset for this Transform

property mag

returns the magnification value for this Transform

property orientation

returns the orientation value for this Transform

4.18 rect

4.18.1 Module Contents

Classes

Rect

Helper class that provides a standard way to create an ABC using

```
class rect.Rect(layer: cni.layer.Layer, box: cni.box.Box)
```

Bases: cni.shape.Shape

Helper class that provides a standard way to create an ABC using inheritance.

property bottom**property left****property right****property top**

```
addToRegion(region: cni.shape.pya.Region)
```

```
clone(nameMap: cni.shape.NameMapper = NameMapper(), netMap: cni.shape.NameMapper =
      NameMapper())
```

```
destroy()
```

```
moveBy(dx: float, dy: float) → None
```

```
toString() → str
```

```
transform(transform: cni.shape.Transform) → None
```

4.19 geo

4.19.1 Module Contents

Functions

fgOr(\rightarrow `cni.grouping.Grouping`)

fgAnd()

fgXor()

fgNot()

fgMerge()

`geo.fgOr(components1: cni.grouping.ulist[cni.grouping.PhysicalComponent], components2: cni.grouping.ulist[cni.grouping.PhysicalComponent], resultLayer: Layer) \rightarrow cni.grouping.Grouping`

`geo.fgAnd()`

`geo.fgXor()`

`geo.fgNot()`

`geo.fgMerge()`

4.20 polygon

4.20.1 Module Contents

Classes

| | |
|----------------|--|
| <i>Polygon</i> | Helper class that provides a standard way to create an ABC using |
|----------------|--|

class `polygon.Polygon`(*arg1*, *arg2*=None)

Bases: `cni.shape.Shape`

Helper class that provides a standard way to create an ABC using inheritance.

addToRegion(*region*: `pya.Region`)

clone(*nameMap*: `cni.shape.NameMapper` = `NameMapper()`, *netMap*: `cni.shape.NameMapper` = `NameMapper()`)

destroy()

getPoints() \rightarrow `cni.pointlist.PointList`

moveBy(*dx*: `float`, *dy*: `float`) \rightarrow None

toString() \rightarrow str

transform(*transform*: `cni.shape.Transform`) \rightarrow None

4.21 numeric

4.21.1 Module Contents

Classes

Numeric

The Numeric class is used to create a floating point number from a string

class numeric.Numeric

Bases: float

The Numeric class is used to create a floating point number from a string representation, such as “10ns”. This string representation is composed of two parts: 1) a number part and 2) a scale factor part. Thus, this Numeric class can be used to represent a floating point number as a floating point number along with a scaling factor. Since this Numeric class is derived from the base Python float class, it can be used just like a regular floating point number in any numerical computation.

The number part of this Numeric class string representation can be any valid Python integer or floating point number; this Python floating point number can be represented using standard scientific notation, such as “1.23e-4”. The scaling factor part of this Numeric class string representation must be one of the following pre-defined scaling factor string values:

| Character | Name | Multiplier |
|-----------|-----------------|------------|
| Y | Yotta | 1e24 |
| Z | Zetta | 1e21 |
| E | Exa | 1e18 |
| P | Peta | 1e15 |
| T | Tera | 1e12 |
| G | Giga | 1e09 |
| M | Mega | 1e06 |
| K or k | Kilo | 1e03 |
| ‘ | no scale factor | 1.0 |
| % | percent | 1e-2 |
| c | centi | 1e-2 |
| m | milli | 1e-3 |
| u | micron | 1e-6 |
| n | nano | 1e-9 |
| p | pico | 1e-12 |
| f | femto | 1e-15 |
| a | atto | 1e-18 |
| z | zepto | 1e-21 |
| y | yocto | 1e-24 |

Note that any characters after the first character in the scaling factor are simply ignored. Thus, the scaling factor “mVolt” is the same as “m”. This capability can be used to create more descriptive scaling factors.

Numeric(int | float | string) – creates a Numeric object, based upon the specified number or string. The string must be a string of the form <number><scaleFactor>, where the <scaleFactor> is one of the pre-defined scaling factors in the above table of scaling factor strings. That is, this string representation must be composed of a number part and a scaling factor part, where the scaling factor is a pre-defined scaling factor string.

property scaleFactor

The default (original) scale factor

property scale_factors

List of all available scaling factors, along with their values

scaleFormat(*scaleFactor=None*)

Returns the floating point number formatted using the specified scaleFactor scaling value. If this scaleFactor parameter is not specified, then the floating point number is returned using the scale factor which was used when the Numeric class object was created.

Parameters

scaleFactor (*string or None*) – Optional scaling factor to use.

Returns

new scaled Numeric object

Return type

Numeric

4.22 signaltype

4.22.1 Module Contents

Classes

SignalType

class signaltype.SignalType

Bases: object

SIGNAL = 1

POWER = 2

GROUND = 3

CLOCK = 4

TIEOFF = 5

TIEHI = 6

TIELO = 7

ANALOG = 8

SCAN = 9

RESET = 10

4.23 text

4.23.1 Module Contents

Classes

| | |
|-------------|--|
| <i>Text</i> | Helper class that provides a standard way to create an ABC using |
|-------------|--|

```
class text.Text(layer, text, point, size)
    Bases: cni.rect.Shape
    Helper class that provides a standard way to create an ABC using inheritance.
    addToRegion(region: pya.Region)
    clone(nameMap: cni.rect.NameMapper = NameMapper(), netMap: cni.rect.NameMapper = NameMapper())
    destroy()
    moveBy(dx: float, dy: float) → None
    setAlignment(align)
    setOrientation(orient)
    setDrafting(drafting)
    transform(transform: cni.rect.Transform) → None
```

4.24 layer

4.24.1 Module Contents

Classes

| |
|--------------|
| <i>Layer</i> |
|--------------|

```
class layer.Layer(name, purpose=None)
    Bases: object
    property name
    property number
    property purposeName
    property purposeNumber
    tech
```

```

layout
getAttrs()
getGridResolution()
getLayerAbove()
getLayerAbove(layerMaterial)
getLayerBelow()
getLayerBelow(layerMaterial)
getLayerName()
getLayerNumber()
getMaterial()
getPurposeName()
getPurposeNumber()
getRoutingDir()
isAbove(layer)
isMaskLayer()

```

4.25 physicalComponent

4.25.1 Module Contents

Classes

| | |
|--------------------------|--|
| <i>PhysicalComponent</i> | Helper class that provides a standard way to create an ABC using |
|--------------------------|--|

class physicalComponent.**PhysicalComponent**

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

abstract **addToRegion**(region: *pya.Region*)

abstract **clone**(nameMap: *cni.namemapper.NameMapper = NameMapper()*, netMap: *cni.namemapper.NameMapper = NameMapper()*)

fgOr(component: *PhysicalComponent*, resultLayer: *Layer*) → *Grouping*

abstract **destroy**()

abstract **moveBy**(dx: *float*, dy: *float*) → *None*

abstract **transform**(transform: *cni.transform.Transform*) → *None*

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

b

box, 7

c

constants, 16

d

dlo, 11

dlogen, 18

f

font, 6

g

geo, 19

grouping, 10

l

layer, 23

location, 13

n

namemapper, 7

numeric, 21

o

orientation, 16

p

pathstyle, 17

physicalComponent, 24

point, 14

pointlist, 17

polygon, 20

r

rect, 19

s

shape, 11

signaltype, 22

t

tech, 15

termtype, 13

text, 23

transform, 18

u

ulist, 12

Symbols

`__call__()` (*dlo.PCellWrapper* method), 12
`__enter__()` (*dlo.PyCellContext* method), 12
`__eq__()` (*point.Point* method), 15
`__exit__()` (*dlo.PyCellContext* method), 12

A

`abut()` (*box.Box* method), 7
`ACCEPT` (*in module constants*), 16
`add()` (*grouping.Grouping* method), 10
`addPin()` (*dlogen.DloGen* method), 18
`addToRegion()` (*grouping.Grouping* method), 10
`addToRegion()` (*physicalComponent.PhysicalComponent* method), 24
`addToRegion()` (*polygon.Polygon* method), 20
`addToRegion()` (*rect.Rect* method), 19
`addToRegion()` (*text.Text* method), 23
`alignEdge()` (*box.Box* method), 7
`alignEdgeToCoord()` (*box.Box* method), 7
`alignEdgeToPoint()` (*box.Box* method), 7
`alignLocation()` (*box.Box* method), 7
`alignLocationToPoint()` (*box.Box* method), 7
`ANALOG` (*signaltype.SignalType* attribute), 22
`append()` (*ulist.ulist* method), 12
`areColinearPoints()` (*point.Point* class method), 14

B

`bottom` (*box.Box* property), 7
`bottom` (*rect.Rect* property), 19
`box`
 module, 7
`Box` (*class in box*), 7

C

`calcBBox()` (*font.Font* method), 6
`cell` (*shape.Shape* attribute), 11
`CENTER_CENTER` (*location.Location* attribute), 13
`CENTER_LEFT` (*location.Location* attribute), 13
`CENTER_RIGHT` (*location.Location* attribute), 13
`centerCenter()` (*box.Box* method), 7
`centerLeft()` (*box.Box* method), 7
`centerRight()` (*box.Box* method), 7

`ChoiceConstraint` (*class in dlo*), 11
`CLOCK` (*signaltype.SignalType* attribute), 22
`clone()` (*box.Box* method), 7
`clone()` (*grouping.Grouping* method), 10
`clone()` (*physicalComponent.PhysicalComponent* method), 24
`clone()` (*polygon.Polygon* method), 20
`clone()` (*rect.Rect* method), 19
`clone()` (*text.Text* method), 23
`compress()` (*pointlist.PointList* method), 17
`concat()` (*orientation.Orientation* method), 16
`constants`
 module, 16
`contains()` (*box.Box* method), 8
`containsPoint()` (*box.Box* method), 8
`containsPoint()` (*pointlist.PointList* method), 18
`copy()` (*point.Point* method), 14

D

`destroy()` (*box.Box* method), 8
`destroy()` (*grouping.Grouping* method), 10
`destroy()` (*physicalComponent.PhysicalComponent* method), 24
`destroy()` (*polygon.Polygon* method), 20
`destroy()` (*rect.Rect* method), 19
`destroy()` (*text.Text* method), 23
`display_text()` (*dlo.PCellWrapper* method), 12
`dlo`
 module, 11
`Dlo` (*class in dlogen*), 18
`dlogen`
 module, 18
`DloGen` (*class in dlogen*), 18

E

`EURO_STYLE` (*font.Font* attribute), 6
`exists()` (*dlogen.Dlo* class method), 18
`expand()` (*box.Box* method), 8
`expandDir()` (*box.Box* method), 8
`expandForMinArea()` (*box.Box* method), 8
`expandForMinWidth()` (*box.Box* method), 8
`expandToGrid()` (*box.Box* method), 8

EXTEND (*pathstyle.PathStyle* attribute), 17

F

fgAnd() (*in module geo*), 20
 fgMerge() (*in module geo*), 20
 fgNot() (*in module geo*), 20
 fgOr() (*in module geo*), 20
 fgOr() (*physicalComponent.PhysicalComponent* method), 24
 fgXor() (*in module geo*), 20
 fix() (*box.Box* method), 8
 FIXED (*font.Font* attribute), 6
 font
 module, 6
 Font (*class in font*), 6

G

geo
 module, 19
 get() (*tech.Tech* method), 16
 get_parameters() (*dlo.PCellWrapper* method), 12
 getArea() (*box.Box* method), 8
 getAttrs() (*layer.Layer* method), 24
 getBBox() (*shape.Shape* method), 11
 getCenter() (*box.Box* method), 8
 getCenterX() (*box.Box* method), 8
 getCenterY() (*box.Box* method), 8
 getComp() (*grouping.Grouping* method), 11
 getComps() (*grouping.Grouping* method), 11
 getCoord() (*box.Box* method), 8
 getCoord() (*point.Point* method), 14
 getDimension() (*box.Box* method), 8
 getGridResolution() (*layer.Layer* method), 24
 getHeight() (*box.Box* method), 8
 getLayerAbove() (*layer.Layer* method), 24
 getLayerBelow() (*layer.Layer* method), 24
 getLayerName() (*layer.Layer* method), 24
 getLayerNumber() (*layer.Layer* method), 24
 getLeft() (*box.Box* method), 8
 getLocationPoint() (*box.Box* method), 8
 getMaterial() (*layer.Layer* method), 24
 getMembers() (*font.Font* class method), 6
 getPoints() (*box.Box* method), 8
 getPoints() (*polygon.Polygon* method), 20
 getPurposeName() (*layer.Layer* method), 24
 getPurposeNumber() (*layer.Layer* method), 24
 getRange() (*box.Box* method), 8
 getRangeX() (*box.Box* method), 8
 getRangeY() (*box.Box* method), 8
 getRelativeOrient() (*orientation.Orientation* method), 16
 getRight() (*box.Box* method), 8
 getRoutingDir() (*layer.Layer* method), 24
 getShape() (*shape.Shape* method), 11

getSpacing() (*box.Box* method), 8
 getSpacing() (*point.Point* method), 14
 getTop() (*box.Box* method), 8
 getWidth() (*box.Box* method), 8
 getX() (*point.Point* method), 15
 getY() (*point.Point* method), 15
 GOTHIC (*font.Font* attribute), 6
 GROUND (*signaltype.SignalType* attribute), 22
 grouping
 module, 10
 Grouping (*class in grouping*), 10

H

hasNoArea() (*box.Box* method), 8

I

init() (*box.Box* method), 8
 INPUT (*termtype.TermType* attribute), 14
 INPUT_OUTPUT (*termtype.TermType* attribute), 14
 INT_MAX (*in module constants*), 16
 INT_MIN (*in module constants*), 17
 intersect() (*box.Box* method), 8, 9
 invalid() (*point.Point* method), 15
 isAbove() (*layer.Layer* method), 24
 isBetween() (*point.Point* method), 15
 isInverted() (*box.Box* method), 9
 isMaskLayer() (*layer.Layer* method), 24
 isNormal() (*box.Box* method), 9
 isValid() (*point.Point* method), 15

J

JUMPER (*termtype.TermType* attribute), 14

L

layer
 module, 23
 Layer (*class in layer*), 23
 layout (*layer.Layer* attribute), 23
 left (*box.Box* property), 7
 left (*rect.Rect* property), 19
 limit() (*box.Box* method), 9
 location
 module, 13
 Location (*class in location*), 13
 LOWER_CENTER (*location.Location* attribute), 13
 LOWER_LEFT (*location.Location* attribute), 13
 LOWER_RIGHT (*location.Location* attribute), 13
 lowerCenter() (*box.Box* method), 9
 lowerLeft() (*box.Box* method), 9
 lowerRight() (*box.Box* method), 9

M

mag (*transform.Transform* property), 18

MATH (*font.Font attribute*), 6
merge() (*box.Box method*), 9
mergePoint() (*box.Box method*), 9
MIL_SPEC (*font.Font attribute*), 6
mirrorX() (*box.Box method*), 9
mirrorX() (*location.Location method*), 13
mirrorY() (*box.Box method*), 9
mirrorY() (*location.Location method*), 13
module
 box, 7
 constants, 16
 dlo, 11
 dlogen, 18
 font, 6
 geo, 19
 grouping, 10
 layer, 23
 location, 13
 namemapper, 7
 numeric, 21
 orientation, 16
 pathstyle, 17
 physicalComponent, 24
 point, 14
 pointlist, 17
 polygon, 20
 rect, 19
 shape, 11
 signaltype, 22
 tech, 15
 termtype, 13
 text, 23
 transform, 18
 ulist, 12
moveBy() (*box.Box method*), 9
moveBy() (*grouping.Grouping method*), 11
moveBy() (*physicalComponent.PhysicalComponent method*), 24
moveBy() (*polygon.Polygon method*), 20
moveBy() (*rect.Rect method*), 19
moveBy() (*text.Text method*), 23
moveTo() (*box.Box method*), 9
moveTowards() (*box.Box method*), 9
MX (*orientation.Orientation attribute*), 16
MXR90 (*orientation.Orientation attribute*), 16
MY (*orientation.Orientation attribute*), 16
MYR90 (*orientation.Orientation attribute*), 16
N
name (*layer.Layer property*), 23
namemapper
 module, 7
NameMapper (*class in namemapper*), 7
number (*layer.Layer property*), 23

numeric
 module, 21
Numeric (*class in numeric*), 21
O
orientation
 module, 16
Orientation (*class in orientation*), 16
orientation (*transform.Transform property*), 19
OUTPUT (*termtype.TermType attribute*), 14
overlaps() (*box.Box method*), 9
P
params_as_hash() (*dlo.PCellWrapper method*), 12
pathstyle
 module, 17
PathStyle (*class in pathstyle*), 17
PCellWrapper (*class in dlo*), 12
physicalComponent
 module, 24
PhysicalComponent (*class in physicalComponent*), 24
place() (*box.Box method*), 9
place() (*point.Point method*), 15
point
 module, 14
Point (*class in point*), 14
pointlist
 module, 17
PointList (*class in pointlist*), 17
polygon
 module, 20
Polygon (*class in polygon*), 20
POWER (*signaltype.SignalType attribute*), 22
produce() (*dlo.PCellWrapper method*), 12
purposeName (*layer.Layer property*), 23
purposeNumber (*layer.Layer property*), 23
PyCellContext (*class in dlo*), 12
R
R0 (*orientation.Orientation attribute*), 16
R180 (*orientation.Orientation attribute*), 16
R270 (*orientation.Orientation attribute*), 16
R90 (*orientation.Orientation attribute*), 16
RangeConstraint (*class in dlo*), 12
rect
 module, 19
Rect (*class in rect*), 19
register() (*tech.Tech method*), 16
REJECT (*in module constants*), 16
removeRegion() (*box.Box method*), 9
RESET (*signaltype.SignalType attribute*), 22
right (*box.Box property*), 7
right (*rect.Rect property*), 19
ROMAN (*font.Font attribute*), 6

rotate180() (*box.Box* method), 9
 rotate180() (*location.Location* method), 13
 rotate270() (*box.Box* method), 9
 rotate270() (*location.Location* method), 13
 rotate90() (*box.Box* method), 9
 rotate90() (*location.Location* method), 13
 ROUND (*pathstyle.PathStyle* attribute), 17

S

scale_factors (*numeric.Numeric* property), 22
 scaleFactor (*numeric.Numeric* property), 21
 scaleFormat() (*numeric.Numeric* method), 22
 SCAN (*signaltype.SignalType* attribute), 22
 SCRIPT (*font.Font* attribute), 6
 set() (*box.Box* method), 9
 set() (*point.Point* method), 15
 set_shape() (*shape.Shape* method), 11
 set_tech() (*dlogen.DloGen* method), 18
 setAlignment() (*text.Text* method), 23
 setBottom() (*box.Box* method), 9
 setCenter() (*box.Box* method), 9
 setCenterY() (*box.Box* method), 9
 setCoord() (*box.Box* method), 9
 setCoord() (*point.Point* method), 15
 setDimension() (*box.Box* method), 9
 setDrafting() (*text.Text* method), 23
 setHeight() (*box.Box* method), 9
 setLocationPoint() (*box.Box* method), 10
 setOrientation() (*text.Text* method), 23
 setRange() (*box.Box* method), 10
 setRangeX() (*box.Box* method), 10
 setRangeY() (*box.Box* method), 10
 setRect() (*box.Box* method), 10
 setRight() (*box.Box* method), 10
 setTop() (*box.Box* method), 10
 setWidth() (*box.Box* method), 10
 setX() (*point.Point* method), 15
 setY() (*point.Point* method), 15
 shape
 module, 11
 Shape (*class in shape*), 11
 SIGNAL (*signaltype.SignalType* attribute), 22
 signaltype
 module, 22
 SignalType (*class in signaltype*), 22
 snap() (*box.Box* method), 10
 snap() (*point.Point* method), 15
 snapTowards() (*box.Box* method), 10
 snapTowards() (*point.Point* method), 15
 snapX() (*box.Box* method), 10
 snapX() (*point.Point* method), 15
 snapY() (*box.Box* method), 10
 snapY() (*point.Point* method), 15
 STICK (*font.Font* attribute), 6

SWEDISH (*font.Font* attribute), 6
 SWITCH (*termtype.TermType* attribute), 14

T

T (*in module ulist*), 12
 tech
 module, 15
 Tech (*class in tech*), 15
 tech (*layer.Layer* attribute), 23
 TechImpl (*class in tech*), 15
 techInUse (*tech.Tech* attribute), 16
 techsByName (*tech.Tech* attribute), 16
 termtype
 module, 13
 TermType (*class in termtype*), 13
 text
 module, 23
 Text (*class in text*), 23
 TIEHI (*signaltype.SignalType* attribute), 22
 TIELO (*signaltype.SignalType* attribute), 22
 TIEOFF (*signaltype.SignalType* attribute), 22
 toDiagAxes() (*point.Point* method), 15
 toOrthogAxes() (*point.Point* method), 15
 top (*box.Box* property), 7
 top (*rect.Rect* property), 19
 toString() (*grouping.Grouping* method), 11
 toString() (*polygon.Polygon* method), 20
 toString() (*rect.Rect* method), 19
 transform
 module, 18
 Transform (*class in transform*), 18
 transform (*transform.Transform* property), 18
 transform() (*box.Box* method), 10
 transform() (*grouping.Grouping* method), 11
 transform() (*location.Location* method), 13
 transform() (*physicalComponent.PhysicalComponent* method), 24
 transform() (*point.Point* method), 15
 transform() (*polygon.Polygon* method), 20
 transform() (*rect.Rect* method), 19
 transform() (*text.Text* method), 23
 TRISTATE (*termtype.TermType* attribute), 14
 TRUNCATE (*pathstyle.PathStyle* attribute), 17

U

ulist
 module, 12
 ulist (*class in ulist*), 12
 UNUSED (*termtype.TermType* attribute), 14
 UPPER_CENTER (*location.Location* attribute), 13
 UPPER_LEFT (*location.Location* attribute), 13
 UPPER_RIGHT (*location.Location* attribute), 13
 upperCenter() (*box.Box* method), 10
 upperLeft() (*box.Box* method), 10

`upperRight()` (*box.Box method*), [10](#)
`USE_DEFAULT` (*in module constants*), [16](#)

V

`VARIABLE` (*pathstyle.PathStyle attribute*), [17](#)

X

`x` (*point.Point property*), [14](#)
`xOffset` (*transform.Transform property*), [18](#)

Y

`y` (*point.Point property*), [14](#)
`yOffset` (*transform.Transform property*), [18](#)