laygo2 cheatsheet 0.1

laygo2

What is laygo2?

The laygo2 package is a comprehensive framework for automatic IC layout generation in Python

laygo2.github.io

Use the following import convention:

```
>>> import numpy as np
```

Create design hierarchy

```
l = laygo2.Library(name="mylib")  # library
d = laygo2.Design(name="mydsn") # design
```

Layout on physical grids

```
r = laygo2.Rect(xy=[[0, 0], [100, 100]], layer=["M1", "drawing"])
t = laygo2.Text(xy=[0, 200], layer=["text", "drawing"], text="test")
p = laygo2.Pin(xy=[[0, 0], [100, 100]], layer=["M1", "pin"], netname="net0"))
i = laygo2.Instance(xy=[0, 0], libname="mylib", cellname="mycell", pins={"net0":p})
d.append([r,t,p,i])
```

Load templates and abstract grids

```
import laygo2_tech as tech
templates = tech.load_templates() # load technology templates
grids = tech.load_grids(templates=templates) # load technology grids
tmpl = templates["nmos"]
i1 = tmpl.generate(params={"nf":4})  # generate instance
pg = grids["placement"]
rg = grids["route_M1_M2"]
```

Layout on abstract grids

```
i = d.place(grid=pg, inst=i, mn=[0, 0])  # place instance
i1 = d.place(grid=pg, inst=i1, mn=pg.mn.top_left(i)) # use pointer
d.route(grid=rg, mn=[[0, 1], rg.mn(i.pins["p"])[0])  # route wire
d.pin(name="X", grid=rg, mn=rg.mn(i.pins["p"])  # pin
```

Export design

```
laygo2.interface.gdspy.export(lib, filename, cellname, scale, layermapfile) # GDS
laygo2.interface.skill.export(lib, filename, cellname, scale) # skill
laygo2.interface.bag.export(lib, filename, cellname, scale) # bag
laygo2.interface.mpl.export(lib, cellname, colormap, order, filename) # matplotlib
```

Register template

```
tmpl1 = d.export_to_template() # convert instance to native template
laygo2.interface.yaml.export(tmpl1, filename, mode="append") # register template
```