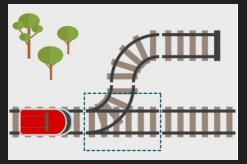
generate environment

import Flatland packages

create custom (or random) environment

```
return rail, rail_map, optionals
```



in this version, it makes sense to use references that mimic **Flatland track IDs** vs. (**track type, rotation**)

`cell((1,0), (21,270)).` → `cell((1,0), 3089).`

find a path

represent custom environment in a format readable by clingo

```
class clingo_map():

    def __init__(self, rail_map):
        self.clingo_str = ""
        self.mapping = {}
```

utilize Clingo API to process custom environment

```
ctl = clingo.Control()

# load the map into clingo
ctl.add("base", [], env.clingo_str)
ctl.add("base", [], types)
ctl.add("base", [], encoding)

# ground & solve
ctl.ground([("base", [])])
ctl.solve(on_model=on_model)
```

visualize results

translate Clingo solution into Flatland actions

```
def is_switch(loc, env):
    return env[loc] in switch_ids

def next_move(dirs, locs, env):
```

the logic of determining which action is necessary is done by considering a change in location, a change in orientation, and the cell itself:

- if the cell doesn't change, `STOP_MOVING`
- else if the orientation shifts by -90° and the cell is a switch, `MOVE_LEFT`
- else if the orientation shifts by +90° and the cell is a switch, `MOVE_RIGHT`
- else `MOVE_FORWARD`