Reynolds
OpenFoam GUI using Blender

Designed and Implemented by:
Deepak Surti

Project Owners
FOSSEE, IIT Bombay
Prabhu R, Shiva S

QUICK BLENDER INTRO

Games and Graphics domains

3D Modeling, Animation

Python API for customization with add-ons

SOLVING CAVITY TUTORIAL

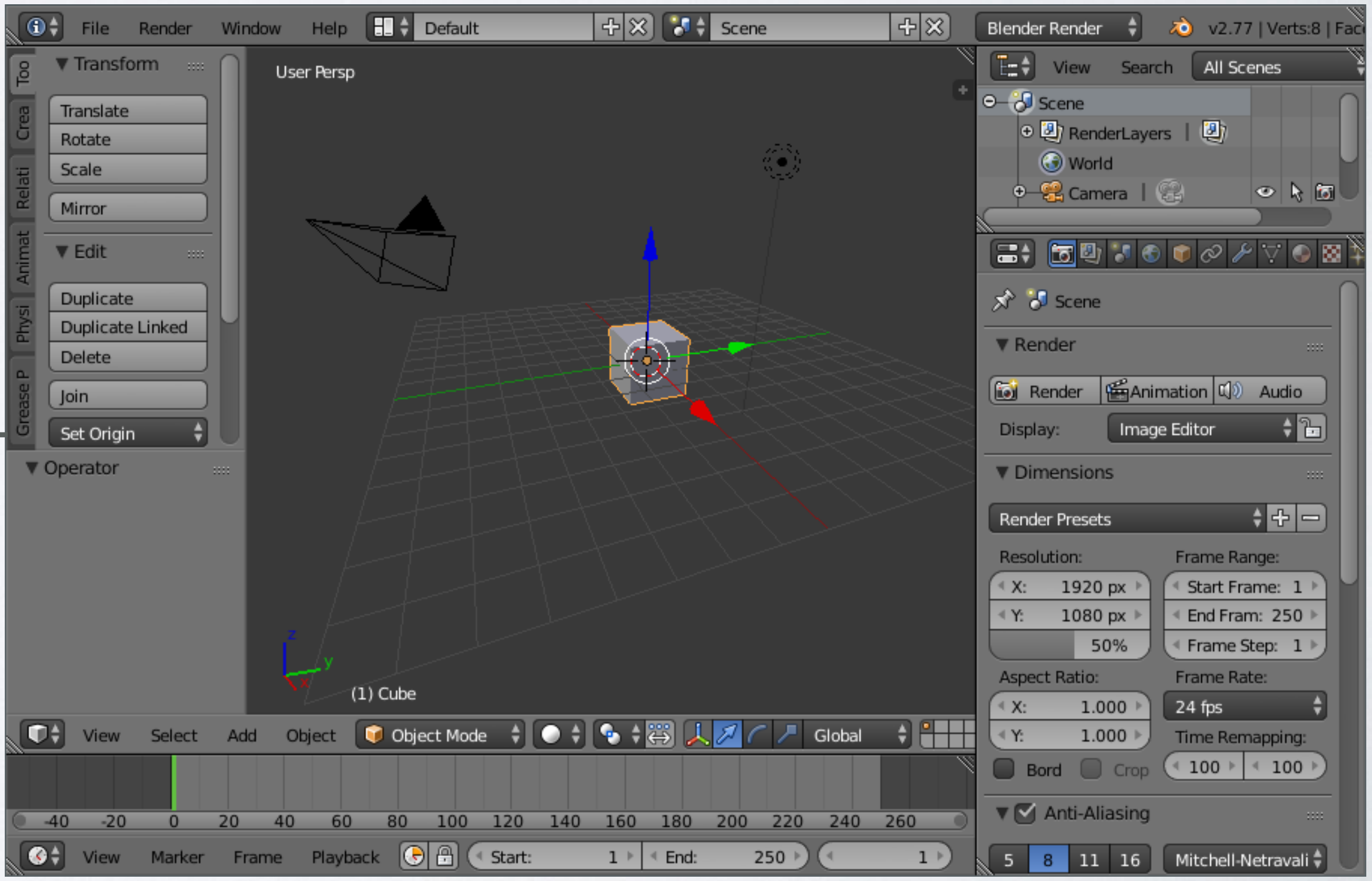
ICOFOAM SOLVER

WHY BLENDER?

PYTHON API
PYTHON Package for OpenFoam
3D Graphics - No Reinventing the Wheel
Advanced 3D Modeling Support

REYNOLDS DESIGN

$$\begin{tikzcd}
\text{reynolds <<openfoam api>>} \arrow[r] & \text{OpenFoam <<Commands>>} \\
\text{reynolds-blender <<add-on>>} \arrow[u] & \text{Blender} \arrow[l]
\end{tikzcd}$$



Display: Image Editor

Resolution: X: 1920 px, Y: 1080 px, 50%

Frame Range: Start Frame: 1, End Fram: 250, Frame Step: 1

Aspect Ratio: X: 1.000, Y: 1.000

Frame Rate: 24 fps

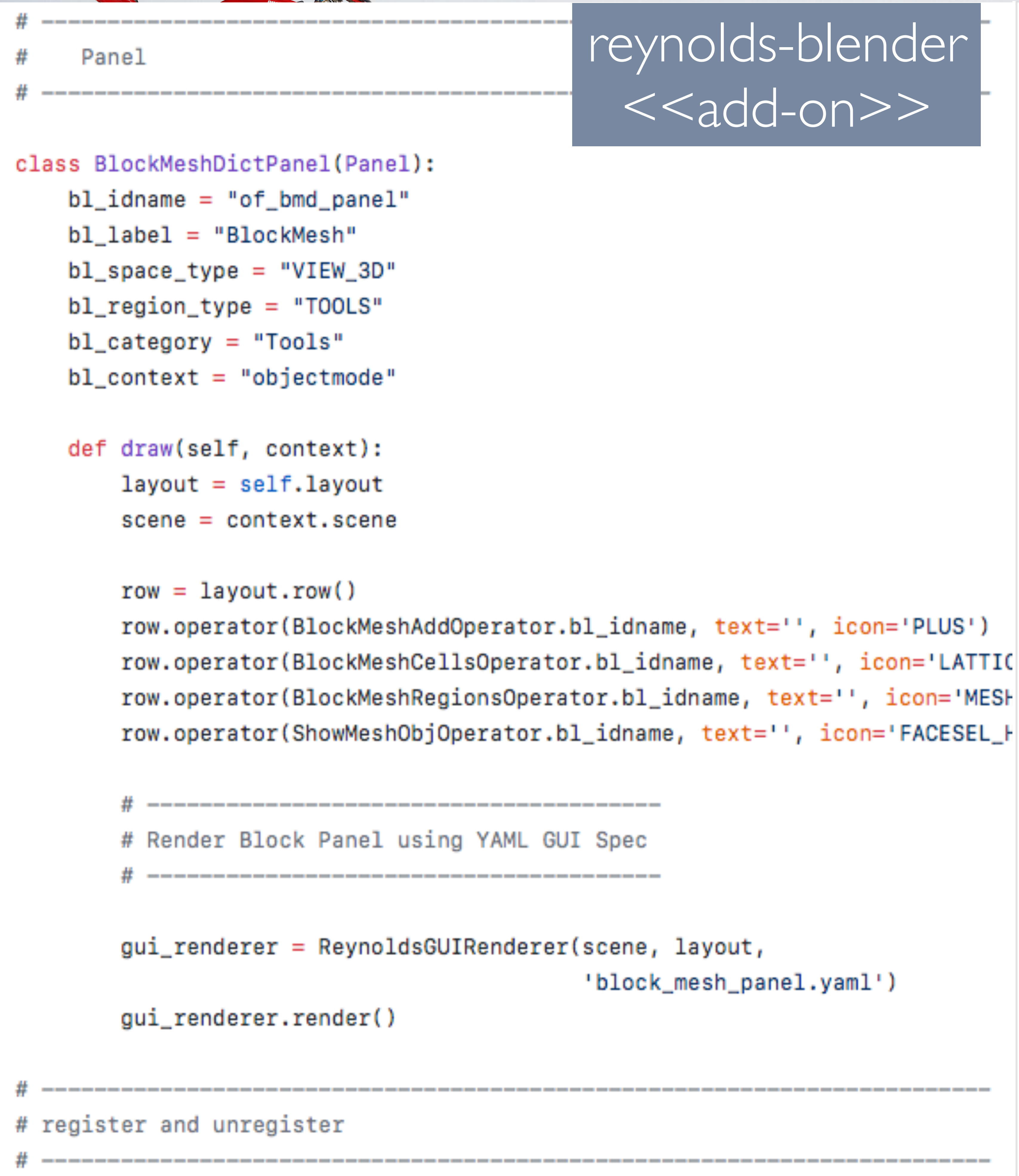
Time Remapping: 100, 100

Bord, Crop

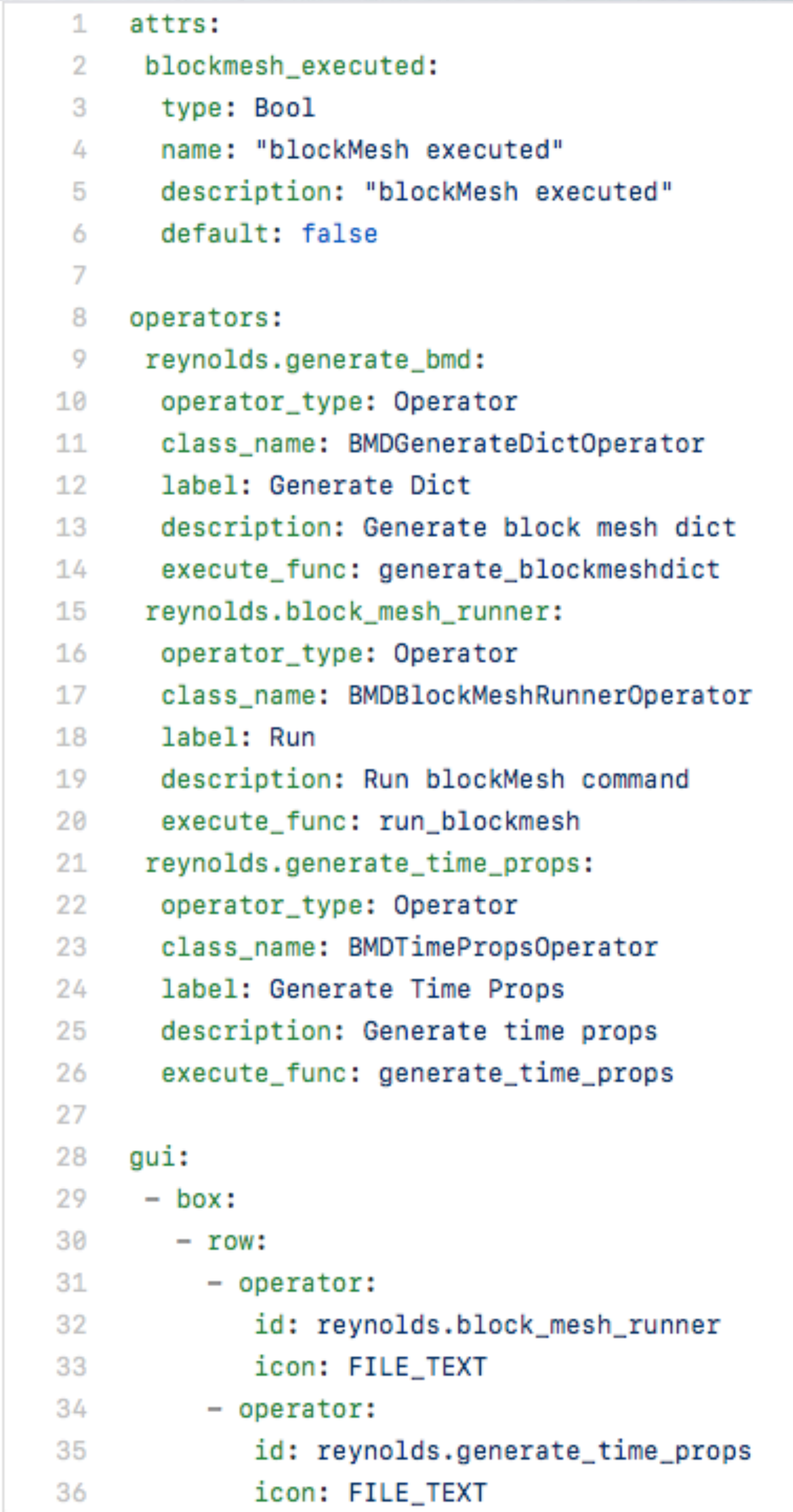
Anti-Aliasing: 5, 8, 11, 16

Start: 1, End: 250, 1

reynolds-blender <<add-on>>

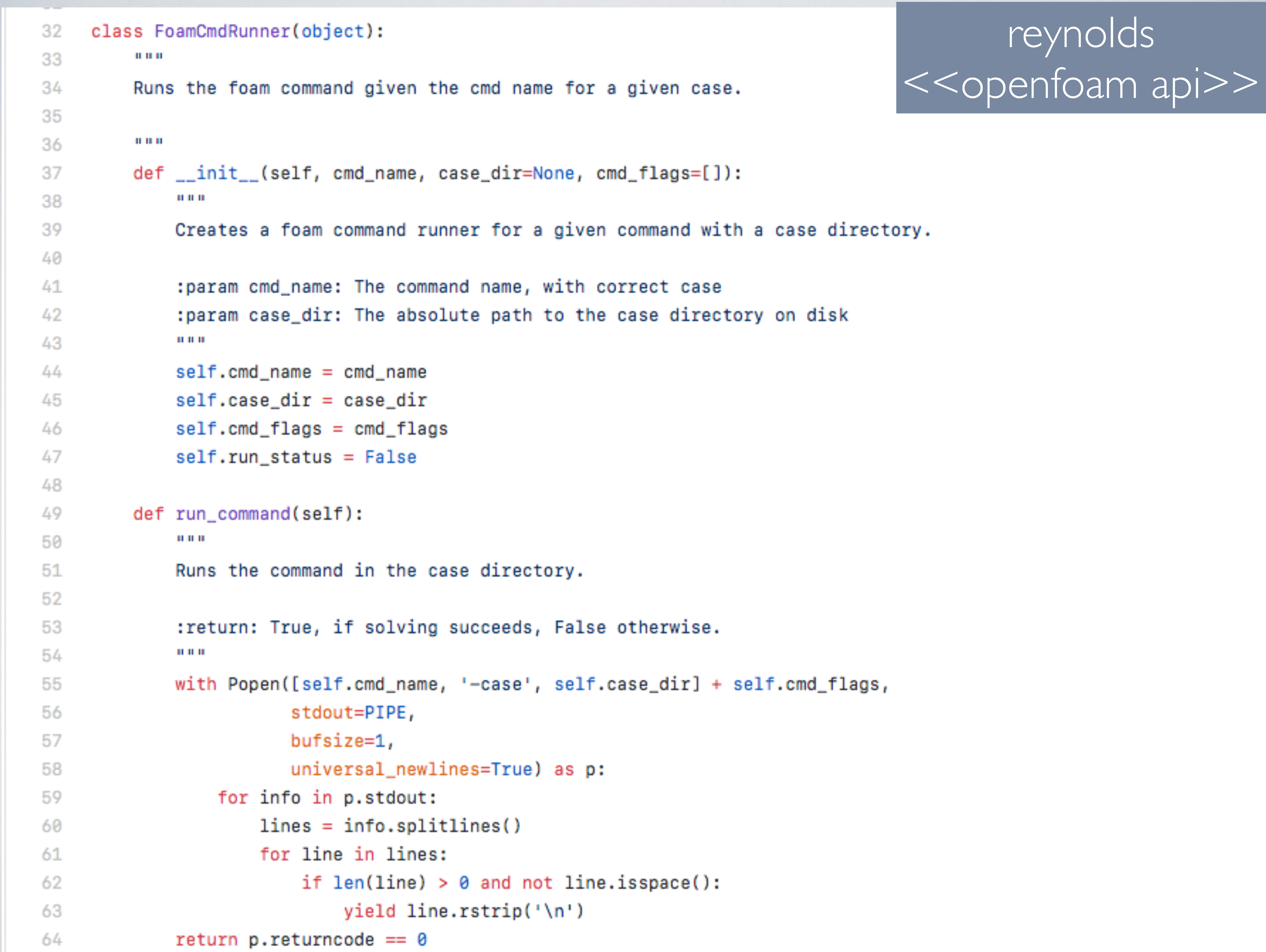


```python
# ------------------------------------------------------------------------------
#    Panel
# ------------------------------------------------------------------------------

class BlockMeshDictPanel(Panel):
    bl_idname = "of_bmd_panel"
    bl_label = "BlockMesh"
    bl_space_type = "VIEW_3D"
    bl_region_type = "TOOLS"
    bl_category = "Tools"
    bl_context = "objectmode"

    def draw(self, context):
        layout = self.layout
        scene = context.scene

        row = layout.row()
        row.operator(BlockMeshAddOperator.bl_idname, text='', icon='PLUS')
        row.operator(BlockMeshCellsOperator.bl_idname, text='', icon='LATTIC
        row.operator(BlockMeshRegionsOperator.bl_idname, text='', icon='MESH
        row.operator(ShowMeshObjOperator.bl_idname, text='', icon='FACESEL_H

        # ---------------------------------------
        # Render Block Panel using YAML GUI Spec
        # ---------------------------------------

        gui_renderer = ReynoldsGUIRenderer(scene, layout,
                                           'block_mesh_panel.yaml')
        gui_renderer.render()

# ------------------------------------------------------------------------------
# register and unregister
# ------------------------------------------------------------------------------
```



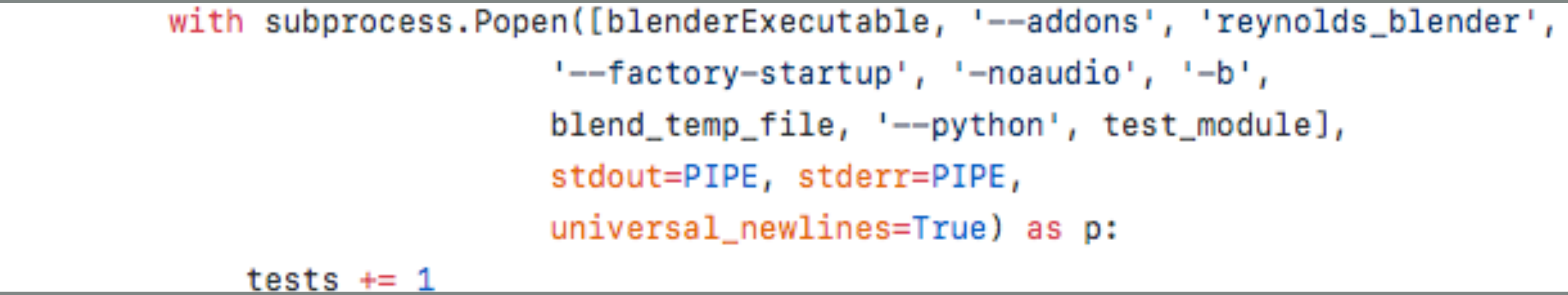
```yaml
attrs:
 blockmesh_executed:
  type: Bool
  name: "blockMesh executed"
  description: "blockMesh executed"
  default: false

operators:
 reynolds.generate_bmd:
  operator_type: Operator
  class_name: BMDGenerateDictOperator
  label: Generate Dict
  description: Generate block mesh dict
  execute_func: generate_blockmeshdict
 reynolds.block_mesh_runner:
  operator_type: Operator
  class_name: BMDBlockMeshRunnerOperator
  label: Run
  description: Run blockMesh command
  execute_func: run_blockmesh
 reynolds.generate_time_props:
  operator_type: Operator
  class_name: BMDTimePropsOperator
  label: Generate Time Props
  description: Generate time props
  execute_func: generate_time_props

gui:
 - box:
   - row:
     - operator:
        id: reynolds.block_mesh_runner
        icon: FILE_TEXT
     - operator:
        id: reynolds.generate_time_props
        icon: FILE_TEXT
```

reynolds <<openfoam api>>

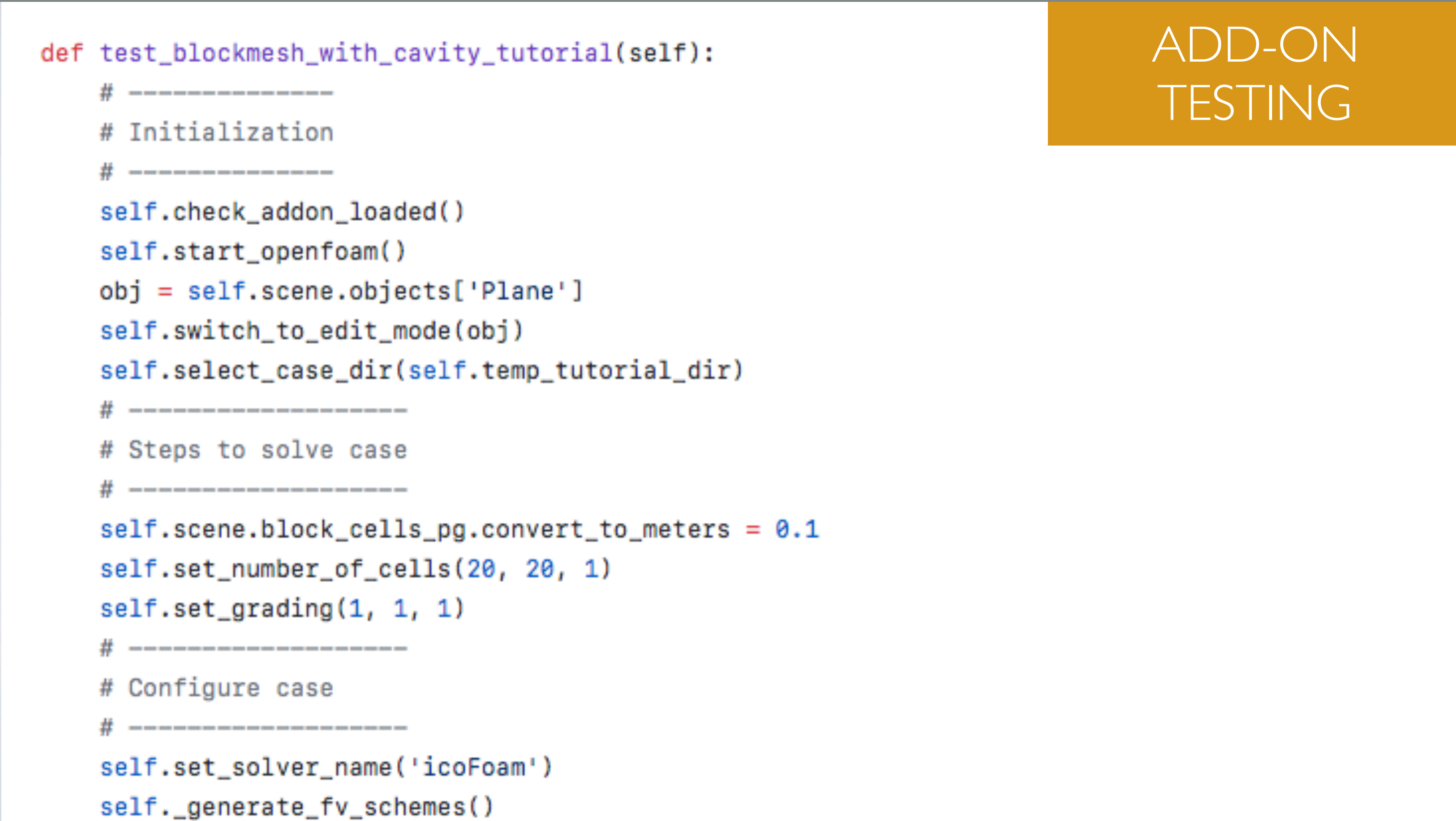


```python
class FoamCmdRunner(object):
    """
    Runs the foam command given the cmd name for a given case.

    """
    def __init__(self, cmd_name, case_dir=None, cmd_flags=[]):
        """
        Creates a foam command runner for a given command with a case directory.

        :param cmd_name: The command name, with correct case
        :param case_dir: The absolute path to the case directory on disk
        """
        self.cmd_name = cmd_name
        self.case_dir = case_dir
        self.cmd_flags = cmd_flags
        self.run_status = False

    def run_command(self):
        """
        Runs the command in the case directory.

        :return: True, if solving succeeds, False otherwise.
        """
        with Popen([self.cmd_name, '-case', self.case_dir] + self.cmd_flags,
                   stdout=PIPE,
                   bufsize=1,
                   universal_newlines=True) as p:
            for info in p.stdout:
                lines = info.splitlines()
                for line in lines:
                    if len(line) > 0 and not line.isspace():
                        yield line.rstrip('\n')
        return p.returncode == 0
```

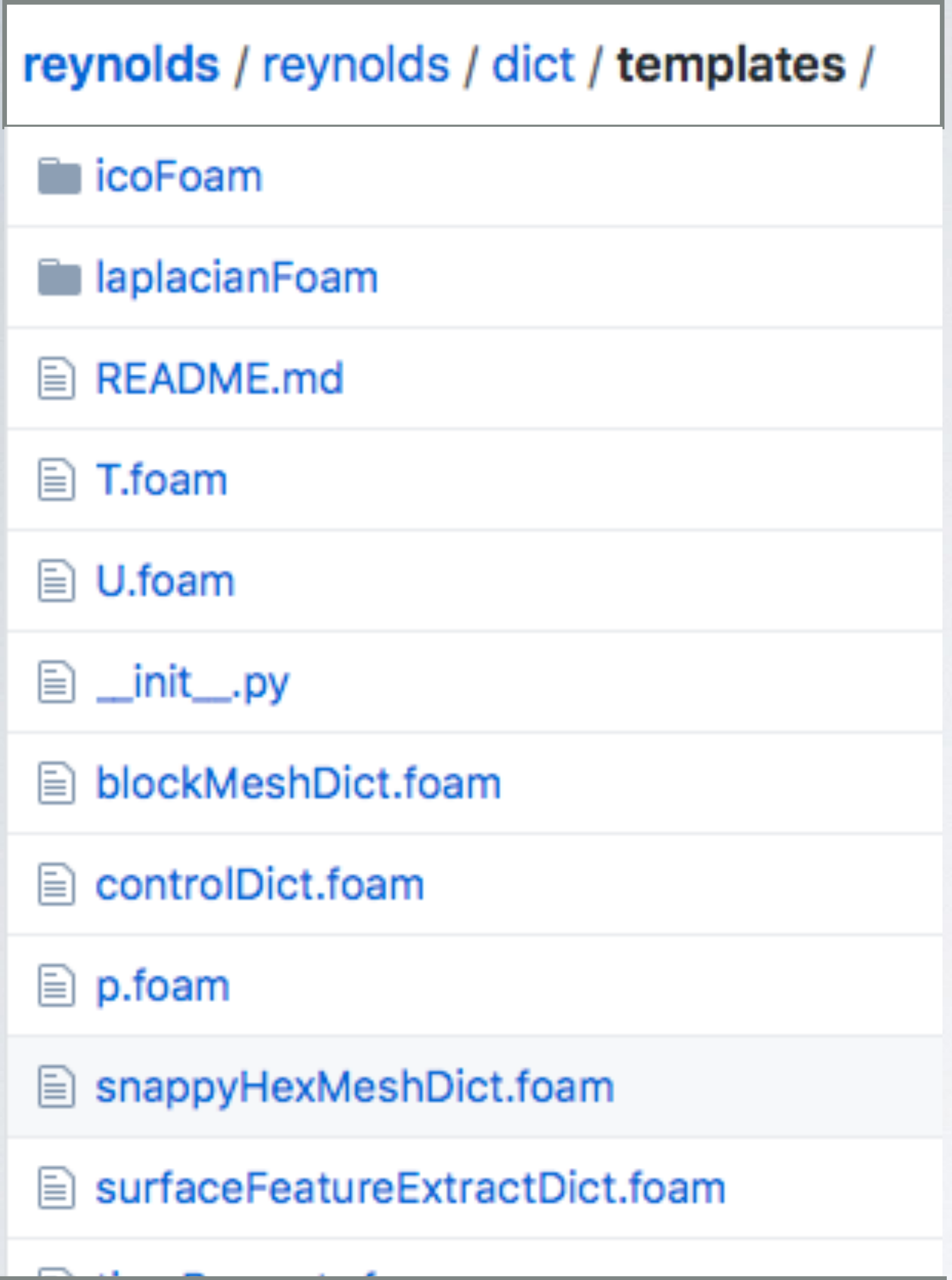
ADD-ON TESTING



```
    with subprocess.Popen([blenderExecutable, '--addons', 'reynolds_blender',
                           '--factory-startup', '-noaudio', '-b',
                           blend_temp_file, '--python', test_module],
                           stdout=PIPE, stderr=PIPE,
                           universal_newlines=True) as p:
        tests += 1
```

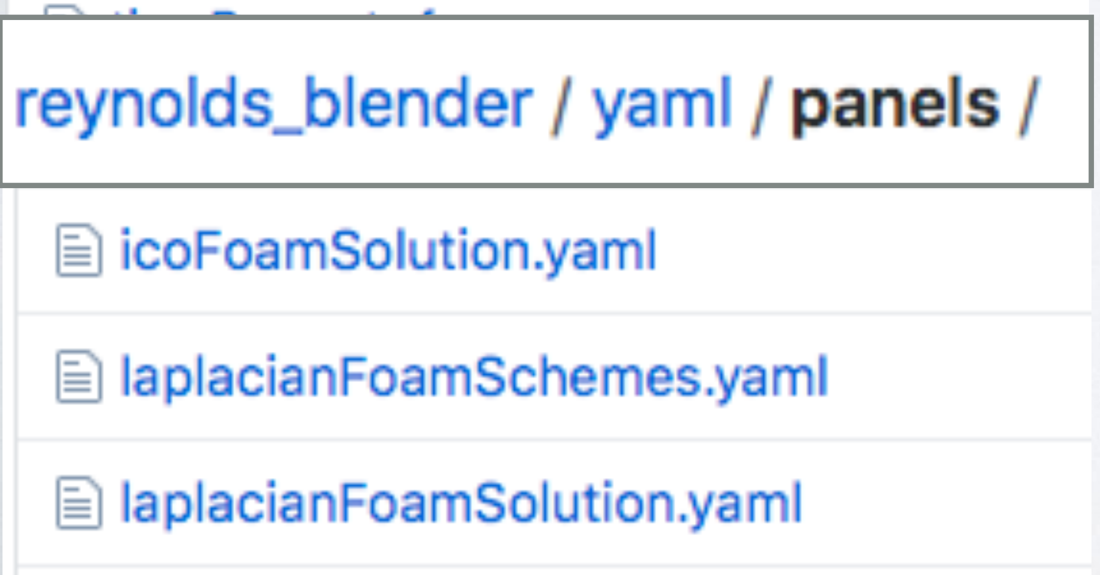


```
def test_blockmesh_with_cavity_tutorial(self):
    # -------------
    # Initialization
    # -------------
    self.check_addon_loaded()
    self.start_openfoam()
    obj = self.scene.objects['Plane']
    self.switch_to_edit_mode(obj)
    self.select_case_dir(self.temp_tutorial_dir)
    # ------------------
    # Steps to solve case
    # ------------------
    self.scene.block_cells_pg.convert_to_meters = 0.1
    self.set_number_of_cells(20, 20, 1)
    self.set_grading(1, 1, 1)
    # ------------------
    # Configure case
    # ------------------
    self.set_solver_name('icoFoam')
    self._generate_fv_schemes()
```



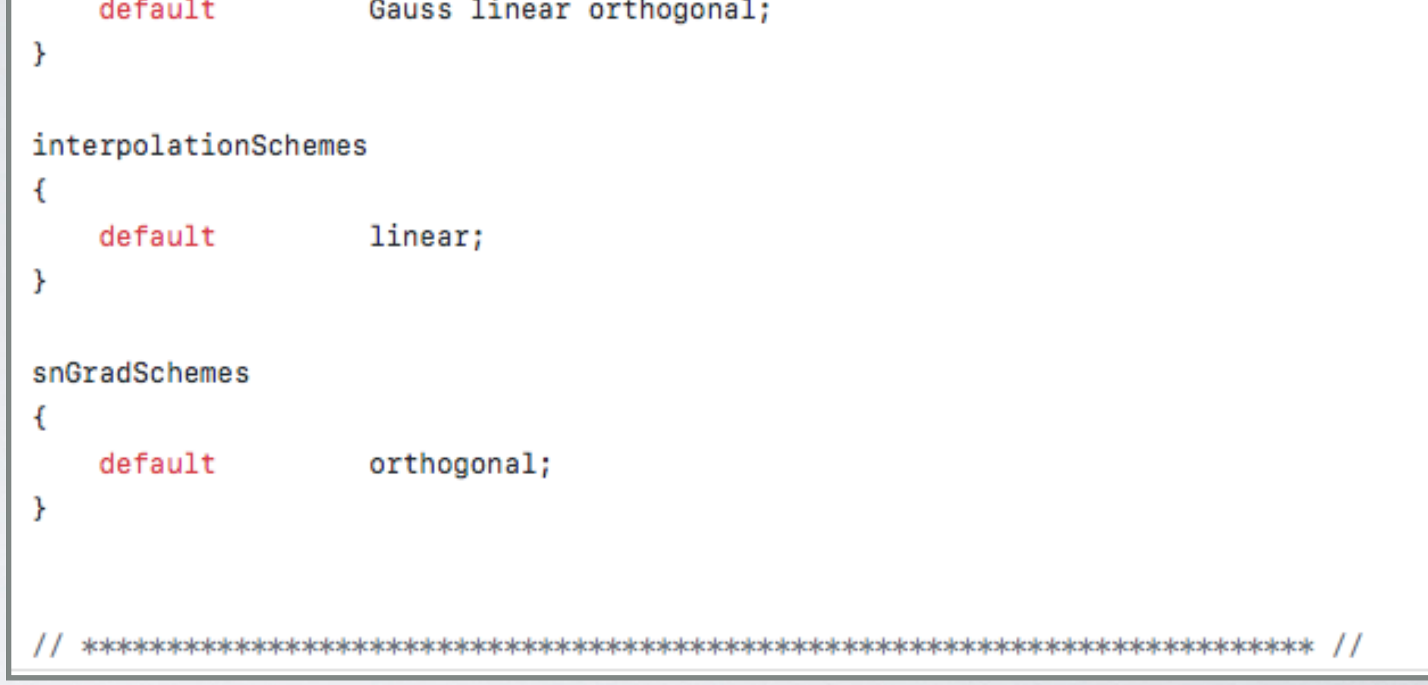
reynolds / reynolds / dict / templates /

- icoFoam
- laplacianFoam
- README.md
- T.foam
- U.foam
- __init__.py
- blockMeshDict.foam
- controlDict.foam
- p.foam
- snappyHexMeshDict.foam
- surfaceFeatureExtractDict.foam

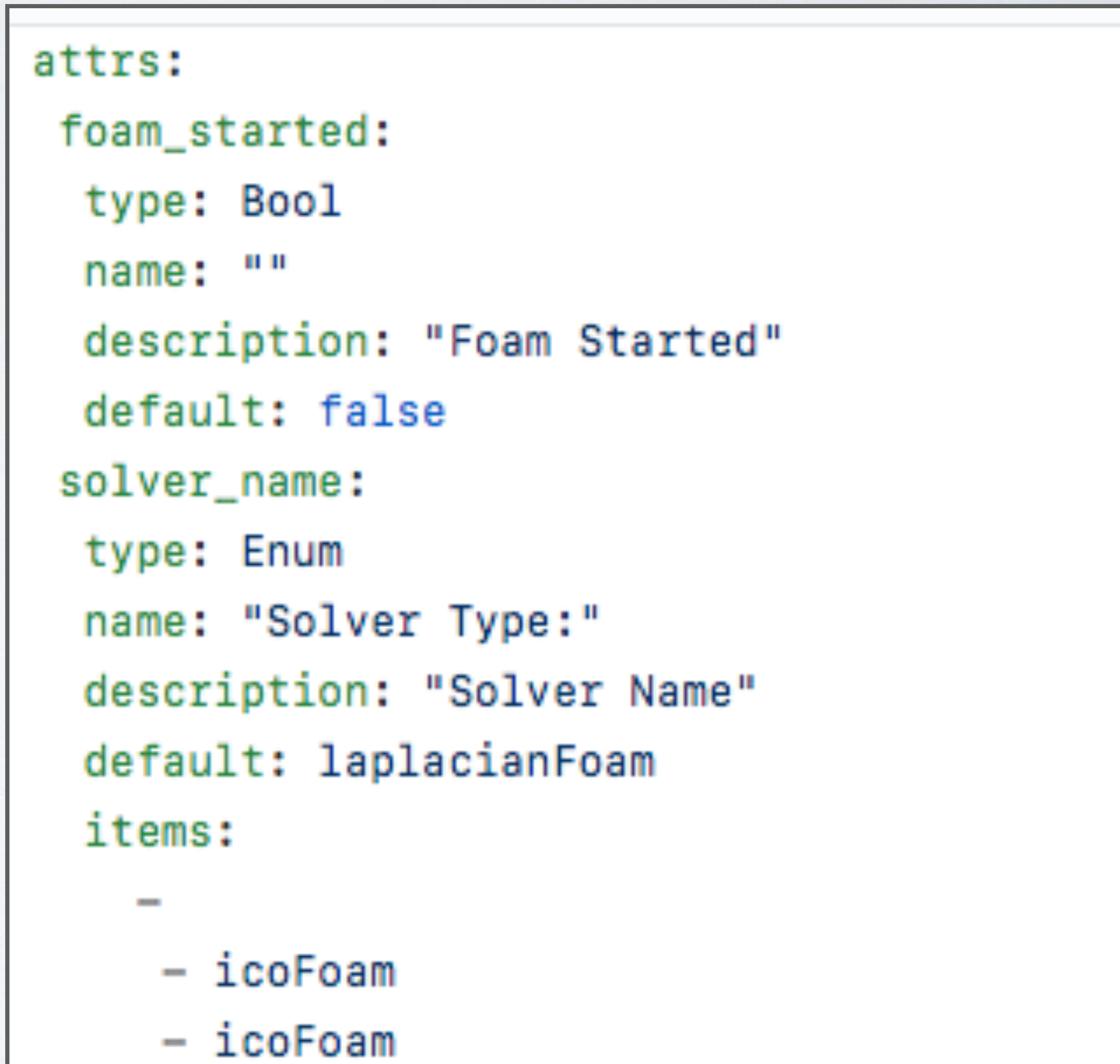


reynolds_blender / yaml / panels /

- icoFoamSolution.yaml
- laplacianFoamSchemes.yaml
- laplacianFoamSolution.yaml



```
    default         Gauss linear orthogonal;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         orthogonal;
}


// ************************************************************************* //
```



```
attrs:
 foam_started:
  type: Bool
  name: ""
  description: "Foam Started"
  default: false
 solver_name:
  type: Enum
  name: "Solver Type:"
  description: "Solver Name"
  default: laplacianFoam
  items:
    -
     - icoFoam
     - icoFoam
```

USING BLENDER? CONS …

Users need on-boarding due to Blender interface learning curve

Blender has it’s own GUI ToolKit

Not integrable in your own app as Python 3D interface

THANK YOU

SPECIAL THANKS TO:

FOSSEE, IIT BOMBAY

Shiva S, Prabhu R

GITHUB:
dmsurti/reynolds,
dmsurti/reynolds-blender