Cheat sheet for PyFluent

Solver Settings Object Interface

/ Launch Fluent locally

The following method is used to start Fluent from Python in gRPC mode.
This code starts Fluent in the background so that commands can be sent to Fluent from the Python interpreter.

```
import ansys.fluent.core as pyfluent
solver = pyfluent.launch_fluent(mode = "solver",
    show_gui = True)
```

The solver object contains attributes such as **file**, **setup**, **solution**, and **results**, which are also instances of settings objects.

/ Import mesh in launched session

The following examples show how to read the available mesh file in Fluent Session.

```
import_filename = 'example_file.msh.h5'
solver.file.read(file_type="case", file_name=
    import_filename)
```

There are other specific APIs available for reading case files and reading case-data files.

```
# e.g., read_case(), read_case_data()

import_filename = 'example_file.cas.h5'
solver.file.read_case(file_type="case", file_name=
    import_filename)
```

/ Enable heat transfer physics

The following examples show how to enable heat transfer by activating the energy equation.

```
solver.setup.models.energy.enabled = True
```

/ Accessing the **object state** with using **pprint**

```
>>> from pprint import pprint
>>> pprint(solver.setup.models.energy())
  {'enabled': True,
   'inlet_diffusion': True,
   'kinetic_energy': False,
   'pressure_work': False,
   'viscous_dissipation': False}
```

/ Define materials

This example shows how you use Solver settings objects to define materials.

```
solver.setup.materials.
    copy_database_material_by_name(type="fluid",
    name="water-liquid")
solver.setup.cell_zone_conditions.fluid["elbow-
    fluid"].material = "water-liquid"
```

/ Define boundary conditions

The examples in this section show how you use Solver settings objects to define boundary conditions.

```
solver.setup.boundary_conditions.velocity_inlet["
    cold-inlet"].vmag = {
    "option": "constant or expression",
    "constant": 0.4,
}
solver.setup.boundary_conditions.velocity_inlet[
    "cold-inlet"
].ke_spec = "Intensity and Hydraulic Diameter"
solver.setup.boundary_conditions.velocity_inlet["
    cold-inlet"].turb_intensity = 5
solver.setup.boundary_conditions.velocity_inlet[
    "cold-inlet"
].turb_hydraulic_diam = "4 [in]"
solver.setup.boundary_conditions.velocity_inlet["
    cold-inlet"].t = {
    "option": "constant or expression",
    "constant": 293.15,
}
```

/ Modify **cell zone** conditions
The examples in this section show how you use Solver settings objects to modify **cell zone** conditions.

```
# Enabling Laminar Zone

solver.setup.cell_zone_conditions.fluid["elbow-
    fluid"] = {"laminar": True}
```

/ Apply solution settings

PyFluent allows you to use Solver settings objects to apply solution settings, initialize, and solve.

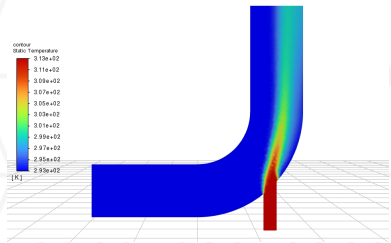
```
solver.solution.initialization.hybrid_initialize()
solver.solution.run_calculation.iterate(
    number_of_iterations=150)
```

/ Post-Processing

PyFluent allows you to post-process data with **results** object. The following example shows you how to create and display contours on a plane.

```
solver.results.graphics.contour["contour"] = {}
solver.results.graphics.contour["contour"].
    print_state()
solver.results.graphics.contour["contour"].field
    = "temperature"
solver.results.graphics.contour["contour"].
    surfaces_list = [
    "symmetry-xyplane",
]
```

/ Temperature Contour



References from PyAnsys Documentation

- Getting Started
- PyFluent Solver Settings Objects
- PyFluent Examples