

PyFluent cheat sheet



Version: 0.38.dev6

Quick Start

```
import ansys.fluent.core as pyfluent
from ansys.units import VariableCatalog as VC
from ansys.fluent.core import (
    ScalarFieldDataRequest,
    VectorFieldDataRequest,
    SurfaceFieldDataRequest,
    PathlinesFieldDataRequest,
    SurfaceDataType,
)
from ansys.fluent.core.solver import *
```

Session Management

Launch sessions

```
solver = pyfluent.Solver.from_install()
meshing = pyfluent.Meshing.from_install()

# Connect to existing session
session = pyfluent.Solver.from_connection(ip,
    port, password)
```

File I/O

```
ReadCase(settings_source=solver)(
    file_name=<case_file>)
ReadCaseData(settings_source=solver)(
    file_name=<data_file>)

WriteCase(settings_source=solver)(
    file_name=<output_case_file>)
WriteCaseData(settings_source=solver)(
    file_name=<output_case_file>)
```

Meshing Workflow

Watertight geometry workflow

```
watertight = meshing.watertight()

import_geom = watertight.import_geometry
import_geom.file_name = <geometry_file>
import_geom.length_unit = "mm"
import_geom()

surf_mesh = watertight.create_surface_mesh
surf_mesh.cfd_surface_mesh_controls.max_size = 0.3
surf_mesh()

watertight.describe_geometry.setup_type = "fluid"
watertight.describe_geometry()

volume_mesh = watertight.create_volume_mesh_wtm
```

```
volume_mesh.volume_fill = "poly-hexcore"
volume_mesh()

solver = meshing.switch_to_solver()
```

Physics Setup

Boundary conditions

```
inlet = VelocityInlet(solver, name="cold-inlet")
inlet.momentum.velocity_magnitude = 0.4
turbulence = inlet.turbulence
turbulence = (turbulence.turbulence_specification.
    INTENSITY_AND_HYDRAULIC_DIAMETER)
turbulence.turbulent_intensity = 0.05
turbulence.hydraulic_diameter = "4 [in]"
inlet.thermal.temperature = 293.15
```

Materials

```
materials = Materials(solver)
air_copy =
    materials.fluid.make_a_copy(from_= "air",
        to= "air-copied")
materials.fluid["air-copied"].viscosity = 1.81e-05

# Create new material
my_solid = materials.solid.create("my-solid")
my_solid.density.value = 2650
my_solid.thermal_conductivity.value = 7.6
```

Cell zone conditions

```
cell_zones = CellZoneConditions(solver)
cell_zones.fluid["elbow-fluid"].general.material =
    "air-copied"
```

Models

```
models = Models(solver)
models.energy.enabled = True
models.energy.viscous_dissipation = True

viscous = Viscous(solver)
viscous.model = viscous.model.K_EPSILON
k_epsilon_model = viscous.k_epsilon_model
k_epsilon_model = k_epsilon_model.REALIZABLE

rad = Radiation(solver)
rad.model = rad.model.MONTE_CARLO
rad.solve_frequency.number_of_histories = 1e7
```

```
species = Species(solver).model.option
species = species.SPECIES_TRANSPORT
```

Solution

```
soln = Solution(solver)
methods = soln.methods
methods.p_v_coupling.flow_scheme = "Coupled"
grad_scheme =
    methods.spatial_discretization.gradient_scheme
grad_scheme = grad_scheme.GREEN_GAUSS_NODE_BASED
```

Report definitions

```
rep_defs = ReportDefinitions(solver)
surf_rep = rep_defs.surface["outlet-temp-avg"] = {}
soln_report_type =
    rep_defs.surface["outlet-temp-avg"].report_type
soln_report_type = soln_report_type.SURFACE_AREA
rep_defs.surface["outlet-temp-avg"].field =
    VC.TEMPERATURE
```

Initialize and solve

```
initialization = Initialization(solver)
init_type = initialization.initialization_type
init_type = init_type.HYBRID
Initialize(solver)()

run_calc = RunCalculation(solver)
run_calc.parameters.iter_count = 100
Iterate(solver)()

# Check convergence
Monitor(solver).residual.plot()
```

Post-Processing

Field data access

```
# Ensure the session is initialized before
# requesting field data

field_data = solver.fields.field_data

abs_press_req = ScalarFieldDataRequest(
    field_name=VC.ABSOLUTE_PRESSURE,
    surfaces=[ "cold-inlet" ], )

abs_pressure =
    field_data.get_field_data(abs_press_req)

vel_req = VectorFieldDataRequest(
    field_name=VC.VELOCITY,
```

```

surfaces=["cold-inlet"],
)
velocity = field_data.get_field_data(vel_req)

surf_req = SurfaceFieldDataRequest(
    data_types=[SurfaceDataType.Vertices,
                SurfaceDataType.FacesCentroid],
    surfaces=["hot-inlet",
              "cold-inlet"])
surfaces = field_data.get_field_data(surf_req)

vertices = surfaces["cold-inlet"].vertices
centroids = surfaces["hot-inlet"].face_centroids

```

Field info (allowed values, active status, ranges)

```

# Access field data categories
scalar_fields = field_data.scalar_fields
vector_fields = field_data.vector_fields
surface_fields = field_data.surfaces

# Discover available field names
scalar_names = scalar_fields.allowed_values()
vector_names = vector_fields.allowed_values()
surface_names = surface_fields.allowed_values()

# Check variable availabilities
temp_active =
    scalar_fields.is_active(VC.TEMPERATURE)

```

```

vel_active = vector_fields.is_active(VC.VELOCITY)

# Query numeric ranges using VariableCatalog
    entries
temp_range = scalar_fields.range(VC.TEMPERATURE)
vmag_range =
    scalar_fields.range(VC.VELOCITY_MAGNITUDE)

```

Reduction functions

```

reduction = solver.fields.reduction

velocity_inlet = VelocityInlet(solver,
                               name="cold-inlet")

pressure_outlet = PressureOutlet(solver,
                                 name="outlet")

wall = WallBoundary(solver, name="wall-elbow")

area = reduction.area(locations=[velocity_inlet])

volume =
    reduction.volume(locations=[velocity_inlet])

avg_pressure =
    reduction.area_average(VC.ABSOLUTE_PRESSURE,
                           locations=[pressure_outlet])
total_mass =
    reduction.mass_integral(VC.ABSOLUTE_PRESSURE,
                           locations=[pressure_outlet])

```

```

locations=[pressure_outlet])

force = reduction.force(locations=[wall])
max_vel = reduction.maximum(VC.VELOCITY_MAGNITUDE,
                            locations=[wall])

```

Solution variables

```

sv_info = solver.fields.solution_variable_info
sv_data = solver.fields.solution_variable_data

zones_info = sv_info.get_zones_info()

temp_data = sv_data.get_data(
    variable_name=VC.TEMPERATURE,
    zone_names=["elbow-fluid"],
    domain_name="mixture")

temp_array =
    sv_data.create_empty_array(VC.TEMPERATURE,
                              "elbow-fluid", "mixture")

temp_array[:] = 500

sv_data.set_data(
    variable_name=VC.TEMPERATURE,
    zone_names_to_data={'elbow-fluid': temp_array},
    domain_name="mixture")

```