PyPrimeMesh cheat sheet

Version: 0.9.5

Launch PyPrimeMesh client

Launch and exit PyPrimeMesh server from Python in gRPC mode:

```
# Launch PyPrimeMesh server
from ansys.meshing import prime

with prime.launch_prime(timeout=20) as
    prime_client:
    model = prime_client.model
    # Define script here
    prime_client.exit()
```

Launch an instance of PyPrimeMesh at IP address 127.0.0.1 and port 50055 with the number of processes set to 4:

```
with prime.launch_prime(
    ip="127.0.0.1", port=50055, n_procs=4
    ) as prime_client:
    model = prime_client.model
```

Read and write files

Read or write files of different formats based on file extensions:

```
from ansys.meshing.prime import lucid

# Define \texttt{mesher} object
mesher = lucid.Mesh(model)
# Read mesh (*.msh) file
mesh_file_name = r"sample1_mesh.msh"
mesher.read(mesh_file_name, append=False)
# Write mesh (*.cdb) file
cdb_file_name = r"sample3_case.cdb"
mesher.write(cdb_file_name)
```

Part summary

Query for the part summary:

```
part = model.get_part_by_name("sample_part")
summary = part.get_summary(prime.PartSummaryParams(
    model))
print("Total number of cells: ", summary.n_cells)
```

Define size controls

Set global sizing parameters:

```
model.set_global_sizing_params(
    prime.GlobalSizingParams(min=0.5, max=16.0,
        growth_rate=1.2)
)
```

Define the curvature size control:

```
curvature_control =
    model.control_data.create_size_control(
  prime.SizingType.CURVATURE
)
control_name = "Curvature_Size_Control"
curvature_control.set_suggested_name(control_name)
eval_type = prime.ScopeEvaluationType.LABELS
scope = prime.ScopeDefinition(
    model,
    evaluation_type=eval_type,
    label_expression="*",
)
curvature_control.set_scope(scope)
curvature_control.set_curvature_sizing_params(
    prime.CurvatureSizingParams(model,
        normal_angle=18)
)
```

Generate wrapper surface mesh

Generate the wrapper surface mesh:

```
mesher.wrap(
  min_size=0.5,
  max_size=16,
  input_parts="flange,pipe",
  use_existing_features=True,
)
```

Generate surface mesh

Generate the surface mesh based on specified minimum and maximum sizes:

```
mesher.surface_mesh(
    min_size=0.5,
    max_size=16,
    generate_quads=True,
)
```

Generate the surface mesh based on size controls:

```
control_name = "Curvature_Size_Control"
mesher.surface_mesh_with_size_controls(
    control_name,
)
```

Analyze surface mesh

Generate surface mesh diagnostics:

```
surface_scope = prime.ScopeDefinition(model,
    part_expression="*")
diag = prime.SurfaceSearch(model)
diag_prms = prime.SurfaceDiagnosticSummaryParams(
    model,
    scope=surface_scope,
    compute_free_edges=True,
)
diag_res = diag.get_surface_diagnostic_summary(
  diag_prms
)
print("Number of free edges : ",
    diag_res.n_free_edges)
```

Generate surface mesh quality metrics:

```
face_quality_measures =
    prime.FaceQualityMeasure.SKEWNESS
quality = prime.SurfaceSearch(model)
quality_params = prime.SurfaceQualitySummaryParams(
    model=model,
    scope=surface_scope,
    face_quality_measures=[face_quality_measures],
    quality_limit=[0.9],
)
qual_summary_res =
    quality.get_surface_quality_summary(
  quality_params
)
print(
    "Maximum surface skewness : ",
    qual_summary_res.quality_results[0].max_quality
)
```

Generate volume mesh

Generate a volume mesh:

```
volume_fill_type = prime.VolumeFillType.HEXCOREPOLY
mesher.volume_mesh(
    volume_fill_type=volume_fill_type,
)
```