# PyWorkbench cheat sheet

**Version: main**

## Connect PyWorkbench to Ansys Workbench from Python

### Connect to a local Workbench server

Perform these steps to link PyWorkbench with a local session:

- Initiate Ansys Workbench.
- Enter the `StartServer()` method in the Workbench command window.
- Use the given port number to link PyWorkbench with the server.

```
from ansys.workbench.core import connect_workbench
workbench = connect_workbench(port=port)
```

### Connect to a remote Workbench server

Execute the following steps to link PyWorkbench with a remote session:

```
from ansys.workbench.core import connect_workbench
host = "server_machine_name_or_IP"
port = server_port_number
workbench = connect_workbench(host=host, port=port)
```

## Launch a Workbench server and start a client

You can launch a Workbench server and start a client through a Python script on the client side.

This script initiates a server on a local system:

```
from ansys.workbench.core import launch_workbench
workbench = launch_workbench()
```

This script initiates a server on a remote Windows device using appropriate user authentication:

```
host = "server_machine_name_or_ip"
username = "your_username_on_server_machine"
password = "your_password_on_server_machine"
workbench = launch_workbench(
    host=host, username=username, password=password
)
```

## Execute scripts on the Workbench server

These methods can be utilized to execute IronPython-based Workbench scripts containing commands or queries, with the help of PyWorkbench:

- `run_script_string()`: Executes a script included within a string
- `run_script_file()`: Executes a script file in the client's working directory

### Use the `run_script_string()` method:

```
wbjn_template = """
import os
import json
import string
import os.path
work_dir = GetServerWorkingDirectory()
projectArchiveToLoad = os.path.join(work_dir,
    "MatDesigner.wbpz")
projectSaveLocation = os.path.join(work_dir,
    "MatDesigner.wbpj")
Unarchive(ArchivePath=projectArchiveToLoad,
    ProjectPath=projectSaveLocation,
    Overwrite=True)
"""
workbench.run_script_string(wbjn_template)
```

### Use the `run_script_file()` method:

```
workbench.run_script_file("project_workflow.wbjn")
```

### Assign output to global variable:

Assign necessary output from these methods to the global variable `wb_script_result` as a JSON string. This script returns all message summaries from the Workbench session:

```
import json
messages = [m.Summary for m in GetMessages()]
wb_script_result = json.dumps(messages)
```

While executing, the following script displays `info`, `warning`, and `error` levels in the logger.

```
workbench.run_script_file("project_workflow.wbjn",
    log_level="info")
```

## Upload and download files

Use the `upload_file()` and `download_file()` methods to transfer data files to and from the server.

Use the `GetServerWorkingDirectory()` query in server-side scripts to obtain the server's operating directory.

This script uploads all PRT and AGDB files matching the wildcard patterns in the working directory from the client to the server:

```
workbench.upload_file("model?.prt", "*.agdb")
```

This server-side script loads a geometry file into a new Workbench system from the server's directory:

```
workbench.run_script_string(
    r"""import os
work_dir = GetServerWorkingDirectory()
geometry_file = os.path.join(work_dir,
    "my_geometry.agdb")
template = GetTemplate(TemplateName="Static
    Structural", Solver="ANSYS")
system = template.CreateSystem()
system.GetContainer(ComponentName="Geometry").SetFile(F
"""
)
```

This server-side script transfers a Mechanical solver output file to the server's directory from Workbench:

```
workbench.run_script_string(
    r"""import os
import shutil
work_dir = GetServerWorkingDirectory()
mechanical_dir = mechanical.project_directory
out_file_src = os.path.join(mechanical_dir,
    "solve.out")
out_file_des = os.path.join(work_dir, "solve.out")
shutil.copyfile(out_file_src, out_file_des)
"""
)
```

This client script retrieves all .out files from the server's working directory:

```
workbench.download_file("*.out")
```

Use the `download_project_archive()` function to save, archive, and download the current Workbench project from the server to the client:

```
workbench.download_project_archive(archive_name="my_pro
```

## Initiate additional PyAnsys services for systems within a Workbench project

### Link PyMechanical and operate

In a Workbench project, you can operate and link the PyMechanical service from the same client machine.

This script creates a Mechanical system server side and then starts the PyMechanical service and client:

```
from ansys.mechanical.core import
    connect_to_mechanical

sys_name = workbench.run_script_string(
    r"""import json
wb_script_result =
```

```
    json.dumps(GetTemplate(
    TemplateName="Static Structural (ANSYS)"
    ).CreateSystem().Name)
"""
)
server_port = workbench.start_mechanical_server(
    system_name=sys_name
)
mechanical = connect_to_mechanical(
    ip="localhost", port=server_port
)
```

### Initiate the PyFluent service

This script initiates the PyFluent service along with the client for a Fluent system that was developed in Workbench:

```
import ansys.fluent.core as pyfluent

sys_name = workbench.run_script_string(
    r"""import json
wb_script_result =
    json.dumps(GetTemplate(
    TemplateName="FLUENT").CreateSystem().Name)
"""
)
server_info_file = workbench.start_fluent_server(
    system_name=sys_name
)
fluent = pyfluent.connect_to_fluent(
    server_info_file_name=server_info_file
)
```

### Initiate the PySherlock service

This script initiates the PySherlock service and its client for a Sherlock system set up in Workbench:

```
from ansys.sherlock.core import launcher as
    pysherlock

sys_name = workbench.run_script_string(
    r"""import json
wb_script_result =
    json.dumps(GetTemplate(
    TemplateName="SherlockPre").CreateSystem().Name
    )
"""
)
server_port = workbench.start_sherlock_server(
    system_name=sys_name
)
sherlock = pysherlock.connect_grpc_channel(
    port=server_port
)
```

- Getting started
- User guide
- Examples
- API reference