Analysis Workbook. Quick Tour

For impatient users and quick reference here is the typical sequence of the steps through which the typical analysis should proceed. Some of these steps should be done only once, some on every login, and others more frequently. Every step is marked with the frequency at which it should be executed. More details can be found in Pyana User Manual and Pyana Reference Manual.

1. **Setup an account** (Freq: once)

This step is described in details in Account Setup section of the workbook. You need to be able to login to analysis host and source the environment setup script at logon time.

2. **Environment setup** (Freq: every logon)

Either source the the environment setup script on every logon: `. /reg/g/psdm/etc/ana_env.sh` or add this command to the shell login script as explained in Account Setup section.

3. **Kerberos ticket** (Freq: every day or as necessary)

If you work with code repository and use commands like `svn`, `addpkg`, etc. then for authorization you will need to obtain a Kerberos ticket. Ticket can be obtained at any time by running command `kinit` which will prompt your for your regular password.

4. **Create a test release** (Freq: once)

Releases are explained in Packages and Releases section. Create new test release with the command like this:

```
newrel ana-current analisys-rel
```

This will create a new directory named `analisys-rel` and populate it with few files:

```
         ./analisys-rel/
                        SConstruct
                        .sit_release
```

Other directories and hidden system files may appear later at compilation stage or when you run `addpkg` or `newpkg` commands.

Below we refer to this directory as a "test release". Usually separate analyses use separate test releases, and each test release should bear unique name if they are located in the same directory.

5. **Change to the test release** (Freq: every logon or whenever needed)

```
cd analisys-rel
sit_setup
```

*The last sit_setup command is very important and must be executed from the release directory **every time you change the directory**.*

6. **Create new analysis package** (Freq: once)

Choose sufficiently unique name for your analysis package and create the package.

⚠ It's better to choose name which contains letters, digits, and underscores only because the package name will be used for Python package names.

```
newpkg my_ana_pkg
```

This will create directory `my_ana_pkg` and populate it with few files and sub-directories:

```
        .../analisys-rel/
                       ./my_ana_pkg/
                                    SConscript
                                    doc/
                                        README
                                        ChangeLog
```

7. **Add analysis module** (Freq: once)

This step depends on which specific analysis framework is used:

- For Python-based analysis framework you need to execute following commands:

  ```
  mkdir my_ana_pkg/src   # only if src directory does not exist yet
  codegen -l pyana-module my_ana_pkg my_ana_mod
  ```

  This will create file `my_ana_pkg/src/my_ana_mod.py` which contains boilerplate code for analysis module. The codegen command can also be used to generate other template modules, as explained here.

- For C++-based analysis framework you need to execute following commands:

  ```
  mkdir my_ana_pkg/src my_ana_pkg/include   # only if src and include directories do not exist yet
  codegen -l psana-module my_ana_pkg my_ana_mod
  ```

  This will create files `my_ana_pkg/include/my_ana_mod.h` and `my_ana_pkg/src/my_ana_mod.cpp` which contain skeleton code for analysis module.

Like with the package name it's better to use names containing letters, digits, and underscores only.

8. **Modify analysis code** (Freq: as much as you need)

Use your favorite editor to add or modify code the analysis module. For Python modules see section Writing User Modules in Pyana User Manual for examples. For C++ modules see Psana User Manual - Old for examples.

9. **Build** (Freq: after every edit)

Run the command which builds all necessary software in your test release:

```
scons
```

10. **Create configuration file** (Freq: once)

In addition to analysis module the job usually needs a separate configuration file. The name and location of the file could be arbitrary. By default Python-based analysis uses file named `pyana.cfg` in current directory so the simplest would be to create it inside the release directory:

```
vi pyana.cfg
```

C++-based analysis uses file named `psana.cfg` in current directory by default, create it with your favorite editor:

```
vi psana.cfg
```

The file will contain both the names of the modules you want to load and parameters you want to pass to those modules, exact content of the configuration file is explained in Pyana User Manual.

11. **Run the analysis** (Freq: as much as you need)

Chose the data you want to process and start analysis job. For Pyana it would be something like (assuming properly prepared `pyana.cfg` in current directory):

```
pyana /reg/d/psdm/AMO/amo14410/xtc/e23-r0406-*.xtc
```

For C++ framework the command may look like (assuming properly prepared `psana.cfg` in current directory):

```
psana /reg/d/psdm/AMO/amo14410/xtc/e23-r0406-*.xtc
```