NATIONAL CENTER FOR SUPERCOMPUTING APPLICATIONS

AMERIFLUX PROJECT

ILLINOIS
NCSA | National Center for
Supercomputing Applications

MINU MATHEW, YONG WOOK KIM

Motivation

- Center for Advanced Bioenergy and Bioproduct Innovation (CABBI)
  - Six crop sites
  - Flux and meteorological data collected from 2008 onwards.
  - Data can help explain the complex interactions between plant cover and climate.
- Ameriflux
  - Largest scientific network dedicated to sharing of flux observations.
  - Data shared must adhere to Ameriflux Flux Processing (FP) format.
  - Micro-meteorological and flux data collected at field sites must undergo various processes

Campbell data logger at a site

Before

- Micro-meteorological and flux data collected at field sites must undergo various processes to adhere to Ameriflux Flux Processing (FP) format
- Before :
  - Technicians manually collected, processed and formatted data.
  - Followed a word document that listed the step by step manual processing that needs to be done for Ameriflux submission.
  - Precluded near-real-time flux availability.
  - Prone to human error, degrades data quality
  - Not automated.

After

- Goal :
  - Automated data pipeline to convert raw data files to Ameriflux Flux Processing(FP) format
  - Minimal user interaction.
  - Open source github repository with high-quality open-source tools and libraries.
- After :
  - Automation of data processing for Eddypro and Pyfluxpro.
  - Seamless run of EddyPro software within the pipeline.
  - Highly modularized, robust and documented code
  - Validations done before and after each process
  - Gives users the ability to configure the settings
  - GUI and command line interfaces
  - All processes are logged
  - Immense reduction in processing time

Ameriflux Pipeline

Steps to run the pipeline :

1. Clone the code repository from github (one time)

```
$ git clone https://github.com/ncsa/ameriflux-pipeline.git
```

2. Install required python libraries (one time)

```
$ pip install -r requirements.txt
```

3. Execute the modules (GUI) (arbitrary number of times)

```
$ python metprocessor.py
```

```
$ python enveditor.py
```

```
$ python pipeline.py
```

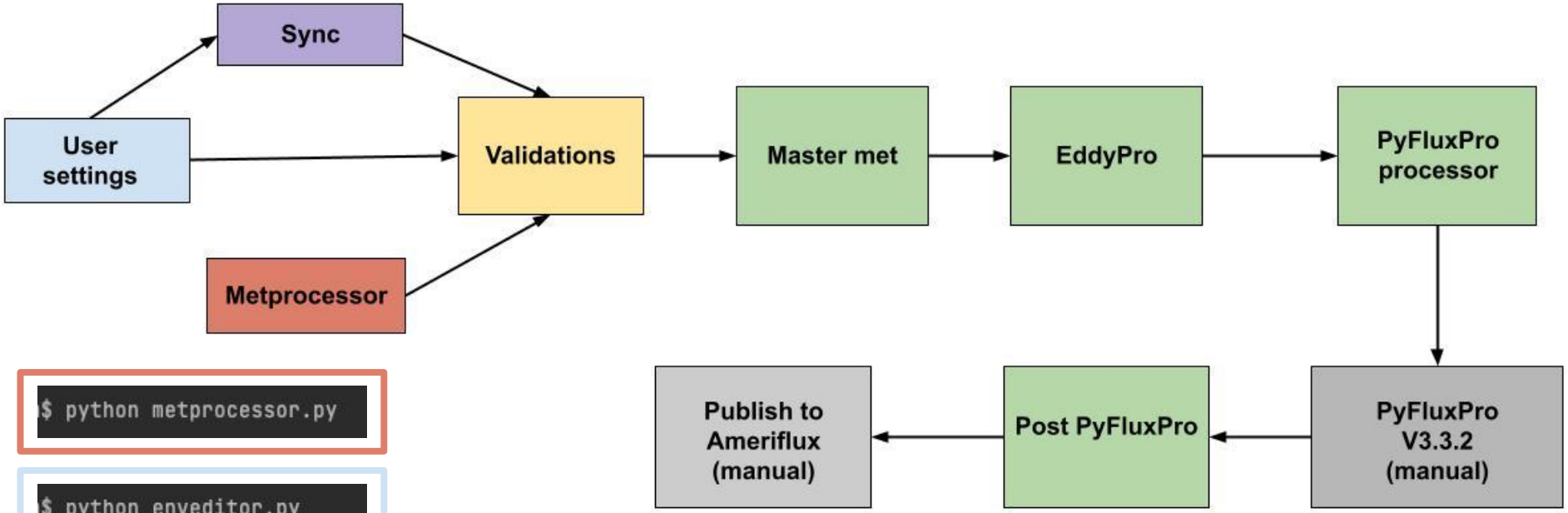
4. Run PyFluxPro V3.3.2

```
$ ./pfp
```

5. Execute the modules

```
$ python pipeline.py
```

Ameriflux Pipeline

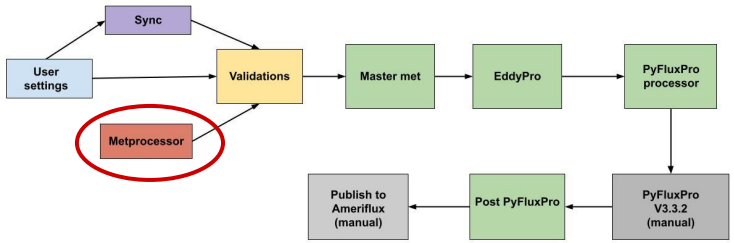



```
$ python metprocessor.py
```

```
$ python enveditor.py
```

```
$ python pipeline.py
```

Metprocessor Module

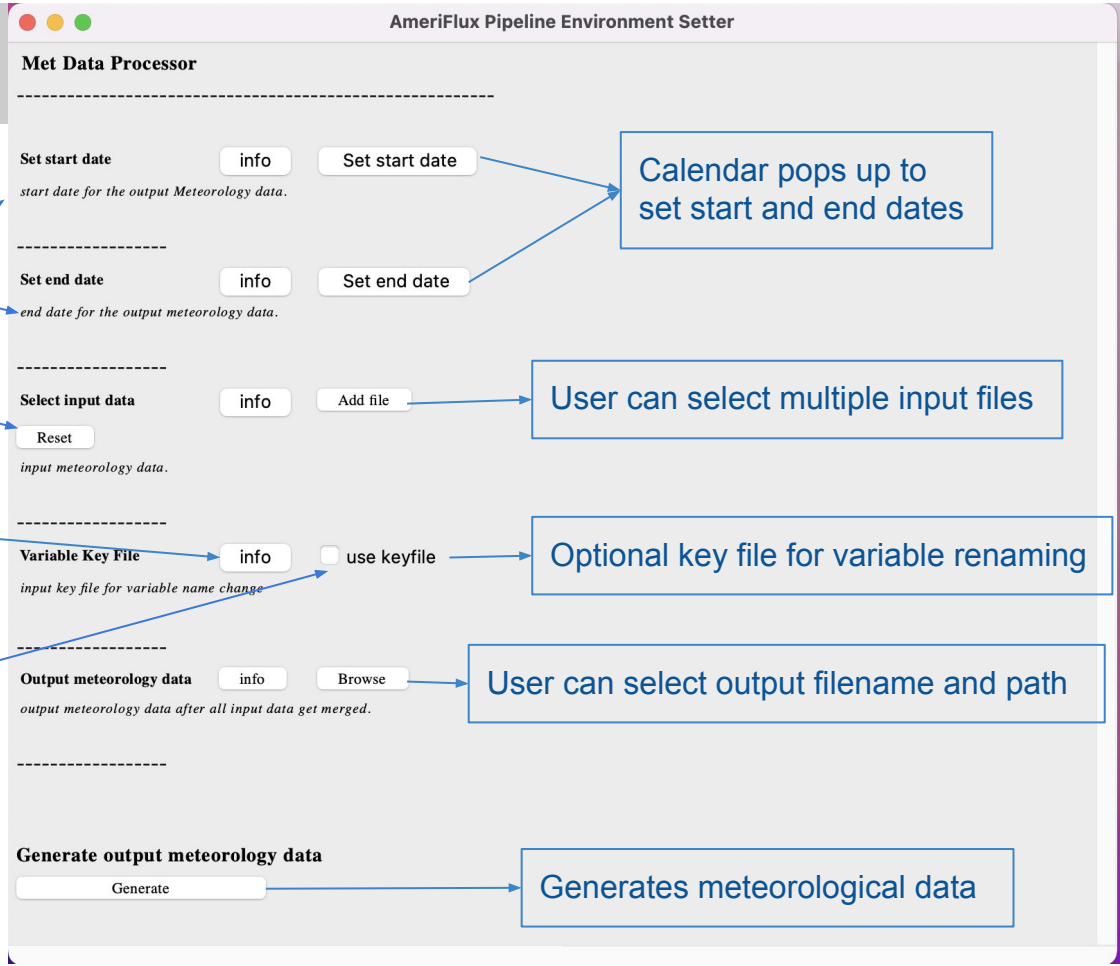


- First step of the pipeline
- Creates a met data that spans the required time period
- User can choose multiple files, a start date and end date from GUI.
- User also has the option to rename certain variable names.
  - Some variables do not use standard naming conventions used from 2021 onwards.
  - User can provide a mapping file for this.
- Demo
  - GUI : `$ python metprocessor.py`
  - Command line :

```
python met_data_processor.py --data
/Users/xx/data/master_met/input/FluxSB_EC.dat,/Users/xx/data/master_met/input/FluxSB_EC.dat.9.backup,/Users/x
x/data/master_met/input/FluxSB_EC.dat.10.backup --start 2021-01-01 --end 2021-12-31 --key
/Users/xx/master_met/input/metmerger_key.xlsx --output /Users/xx/data/master_met/input/Flux.csv
```

Metprocessor Module

- GUI : `$ python metprocessor.py`



One line description

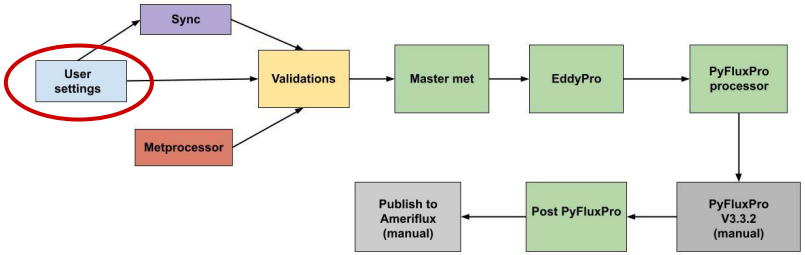
Reset button

Info button

If selected, user can browse a mapping file

Enveditor Module

- Pipeline can be configured as per user requirements
- Launches a GUI that takes in user inputs and settings to run the pipeline.
  - Settings are divided into 3 parts :
    i. Eddypro input data configurations
    ii. EddyPro software run configurations
    iii. PyFluxPro input configurations
  - Each setting has a one line description
  - Each setting has an “Info” button which displays more information
  - All settings has a default set value (including input file paths)
  - Generates a .env file with user settings.
- Users can directly modify .env file. An example file is given in github.
- The .env file is read by the python modules
- Demo
  - GUI : `$ python enveditor.py`



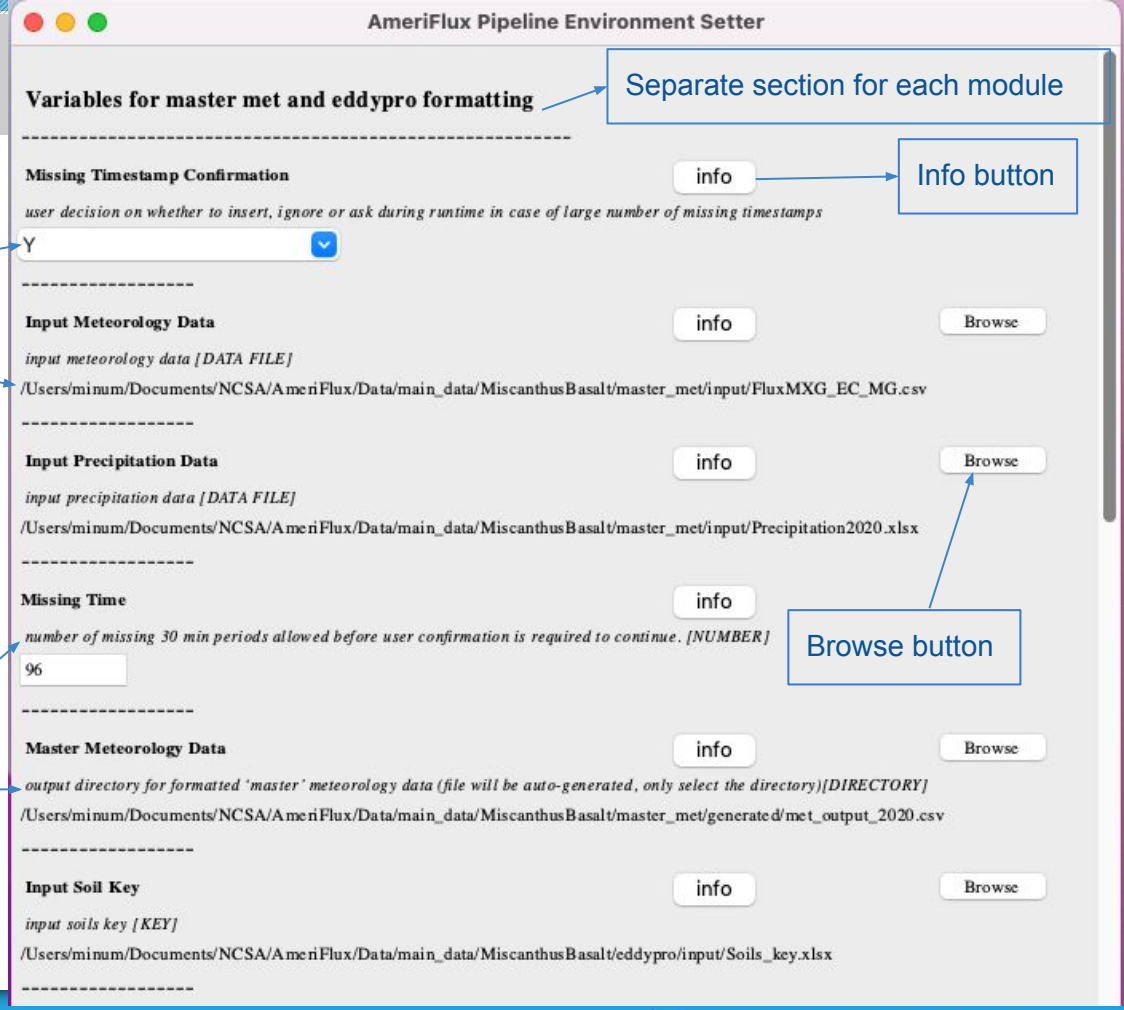
Enveditor Module

- GUI : `$ python enveditor.py`

Dropdown menu

Default values

One line description with expected input type



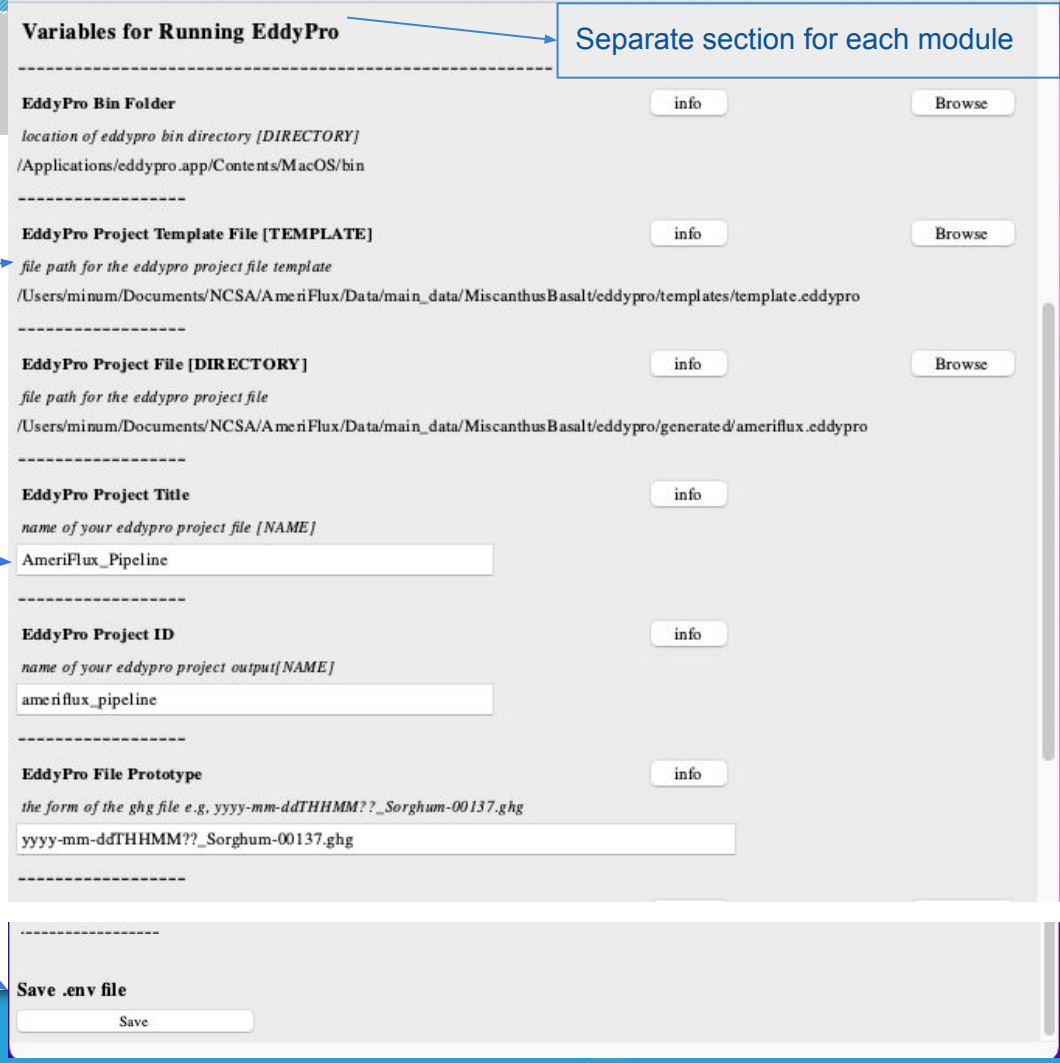
Enveditor Module

- GUI : `$ python enveditor.py`

One line description with expected input type

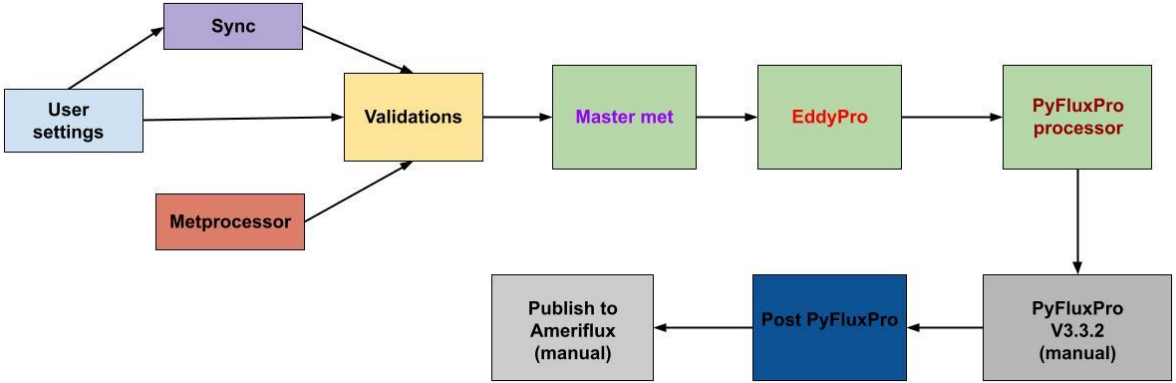
Text input

Save user settings



Pipeline Modules

- Pre-Pyfluxpro : All the processing till the generation of Pyfluxpro L1 and L2 files for ameriflux
  - Master met : Creates master met file and formats it for eddypro input
  - EddyPro : Headless run of EddyPro software
  - Pyfluxpro processor : Creates all input files for PyFluxPro software
- Post-Pyfluxpro : Creates Ameriflux submission-ready csv file, from Pyfluxpro nc output file.
- Utils
  - Sync : Syncs a folder in a remote server with that of a local machine. Can be used to download data
  - Validations : validates user inputs and checks if each process is executed as expected.



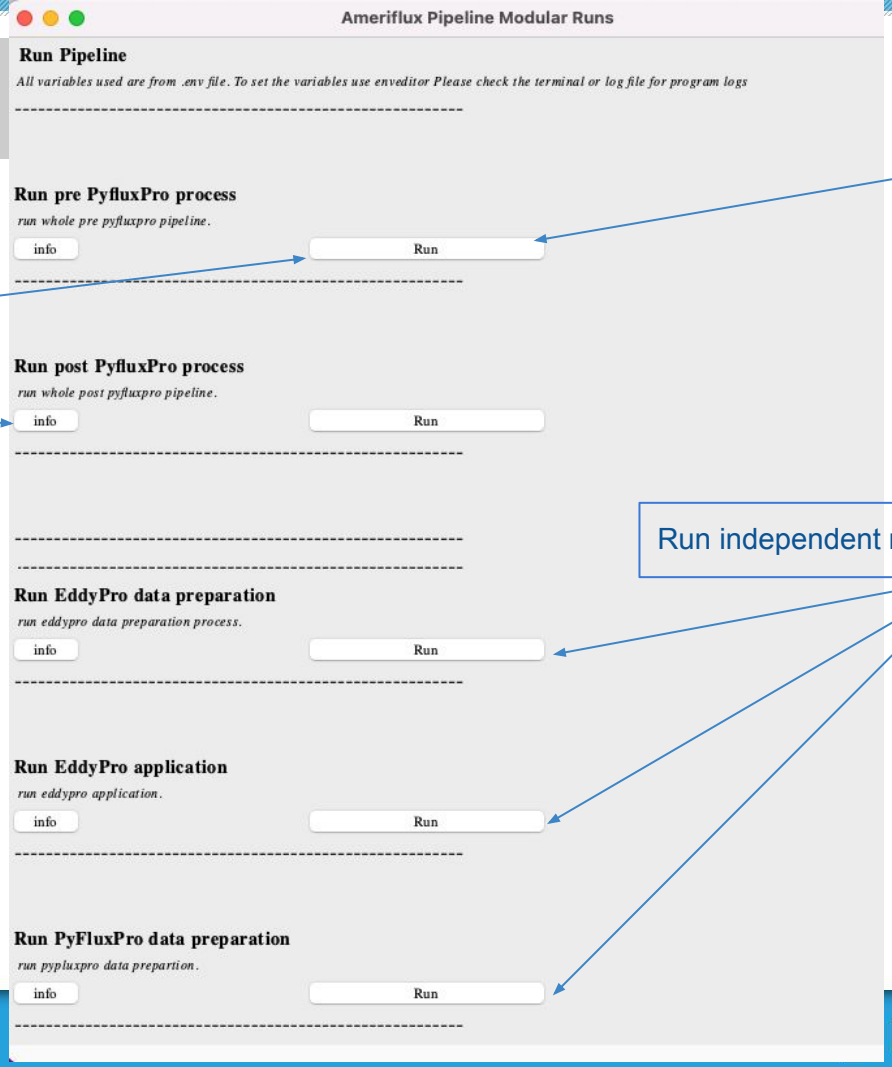
Pipeline Module

- Entire pipeline is highly modularized
- Users can run each modules separately.
- Demo
  - GUI

```
$ python pipeline.py
```

Pipeline Module

- GUI : `$ python pipeline.py`



Run Pre-Pyfluxpro module

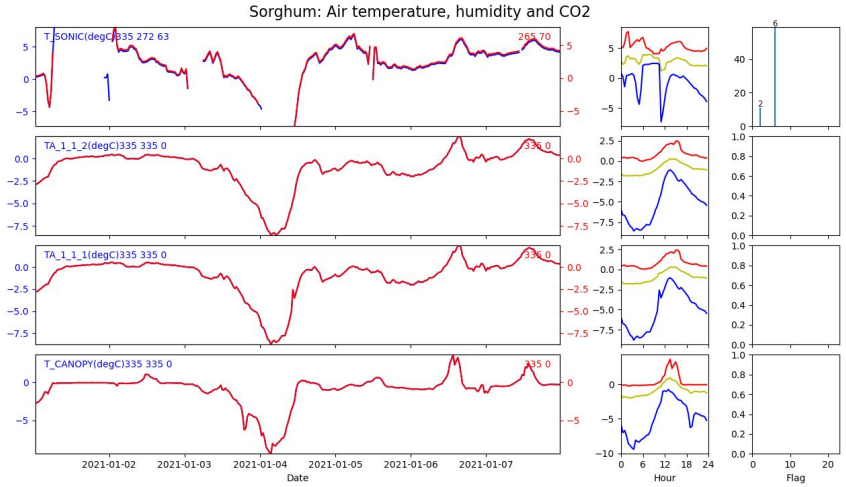
Run button

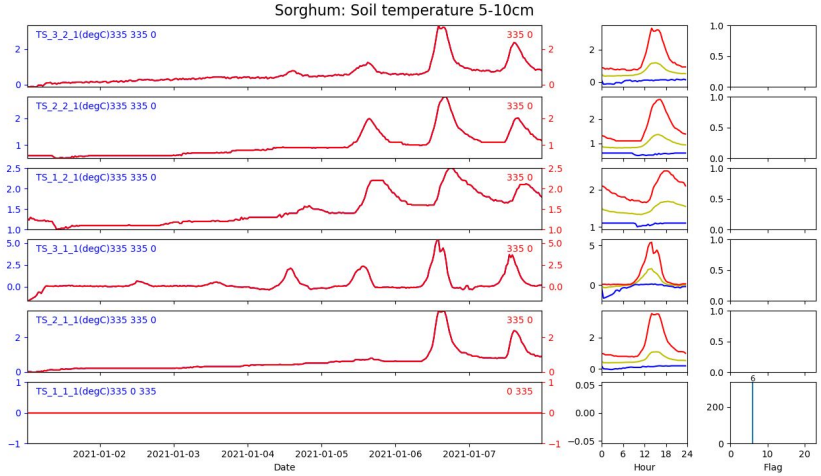
Info button

Run independent modules within Pre-Pyfluxpro

PyFluxPro Software

- After executing Pre-Pyfluxpro module, all inputs for PyFluxPro is generated.
- Run PyFluxPro V3.3.2 with the generated inputs. `n$ ./pfp`
- Analyse the plots for data quality.
- This is a manual step.

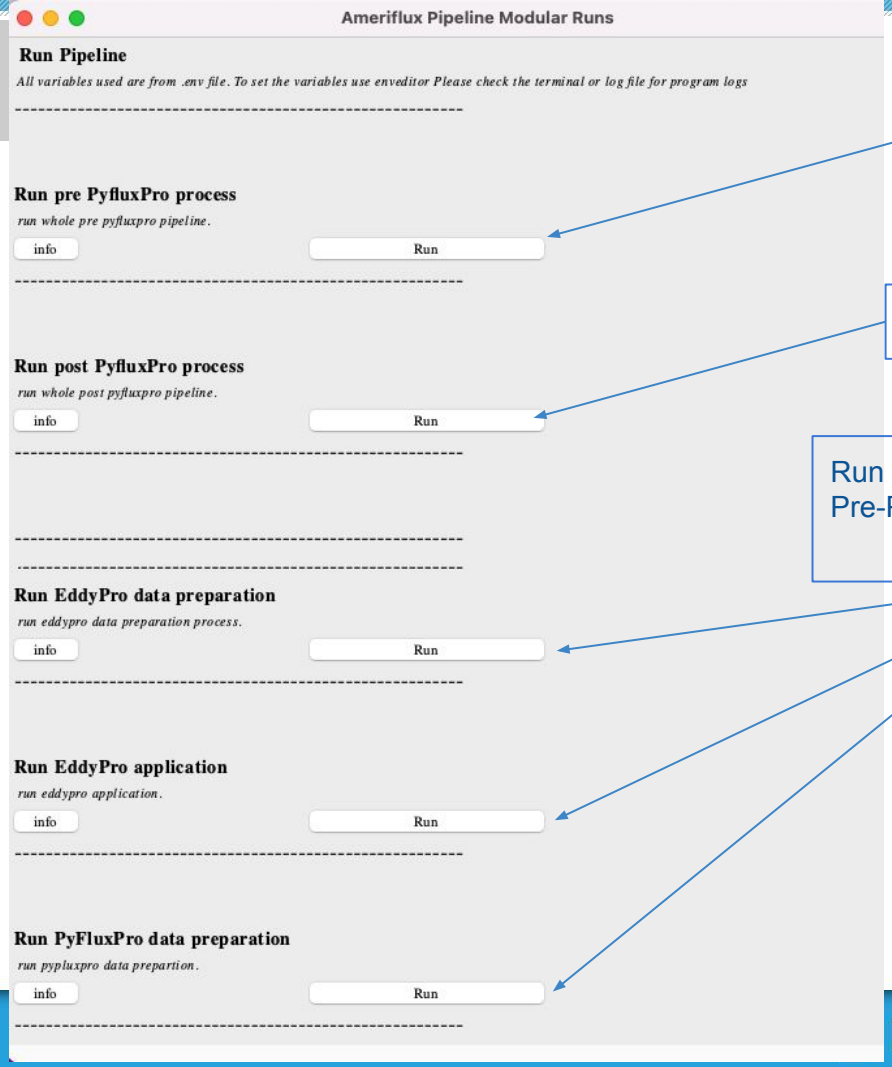




Pipeline Module

- GUI : `$ python pipeline.py`

Post-Pyfluxpro module generates Ameriflux-ready csv file



Run Pre-Pyfluxpro module, if necessary

Run Post-Pyfluxpro module

Run independent modules within Pre-Pyfluxpro, if necessary

Logging

- All processes are logged and captured in log files and in terminal.
- Logs are separated into INFO, WARNING and ERROR
- Logs mention which process is running
- Metprocessor module creates met_processor.log
- Pre-pyfluxpro module creates pre_pyfluxpro.log
- Post-pyfluxpro module create post_pyfluxpro.log
- EddyPro logs for each eddypro run is captured in time-stamped log files in output path

```
2022-10-18T15:26:16.032Z INFO    : __main__ - ------------------------------------------------
2022-10-18T15:26:16.032Z INFO    : __main__ - ############# Process Started #############
2022-10-18T15:26:16.036Z INFO    : __main__ - User input validations complete
2022-10-18T15:26:16.036Z INFO    : __main__ - Pre-processing of PyFluxPro run output has been started
2022-10-18T15:26:16.036Z INFO    : utils.data_util - Read csv file /Users/minum/Documents/NCSA/AmeriFlux/Data/main_data/MiscanthusBasalt/master_met/input/FluxMXG_EC_MG.csv
2022-10-18T15:26:16.129Z INFO    : master_met.mastermetprocessor - Input meteorological data contains 2623 rows and 71 columns
2022-10-18T15:26:16.134Z INFO    : utils.data_util - Read excel file /Users/minum/Documents/NCSA/AmeriFlux/Data/main_data/MiscanthusBasalt/master_met/input/Precipitation2020.xlsx
2022-10-18T15:26:18.737Z INFO    : master_met.mastermetprocessor - Checking for missing timestamps in precip data
2022-10-18T15:26:18.740Z INFO    : master_met.mastermetprocessor - 2 missing timeslot(s) found between 2020-10-30 10:30:00 and 2020-10-30 10:45:00
2022-10-18T15:26:18.740Z INFO    : master_met.mastermetprocessor - inserting 2 row(s) between 2020-10-30 10:30:00 and 2020-10-30 10:45:00
2022-10-18T15:26:18.742Z INFO    : master_met.mastermetprocessor - 1 missing timeslot(s) found between 2020-10-09 10:20:00 and 2020-10-09 10:30:00
2022-10-18T15:26:18.742Z INFO    : master_met.mastermetprocessor - inserting 1 row(s) between 2020-10-09 10:20:00 and 2020-10-09 10:30:00
2022-10-18T15:26:18.744Z INFO    : master_met.mastermetprocessor - 1 missing timeslot(s) found between 2020-03-17 14:30:00 and 2020-03-17 14:40:00
2022-10-18T15:26:18.744Z INFO    : master_met.mastermetprocessor - inserting 1 row(s) between 2020-03-17 14:30:00 and 2020-03-17 14:40:00
2022-10-18T15:26:18.745Z INFO    : master_met.mastermetprocessor - 1 missing timeslot(s) found between 2020-03-17 13:25:00 and 2020-03-17 13:35:00
```

Code Structure and Documentation

```
├── data
│   ├── eddypro
│   │   ├── input
│   │   │   └── README.md
│   │   ├── output
│   │   │   └── README.md
│   │   └── templates
│   │       └── README.md
│   ├── master_met
│   │   ├── input
│   │   │   └── README.md
│   │   └── output
│   │       └── README.md
│   └── pyfluxpro
│       ├── input
│       │   └── README.md
│       └── output
│           └── README.md
```

data/ directory : stores all data.

Output dir- outputs generated by code. Input dir- inputs required for that dir.

- README file present for each location
- master_met/ input : stores all data required for creating master met
- master_met/ output : data outputs from mastermet processor
- eddypro/ input : all data required for eddypro processor module
- eddypro/ templates : eddypro project template file to run EddyPro software
- eddypro/ output : data outputs from eddypro processor module
- pyfluxpro/ input : data inputs for pyfluxpro processor module
- pyfluxpro/ output : data outputs from pyfluxpro processor module

Code Structure and Documentation

```
└── ameriflux-pipeline
    ├── CHANGELOG.md
    ├── CONTRIBUTORS.md
    ├── NOTES.md
    ├── README.md
    ├── ameriflux_pipeline
    ├── docs
    ├── requirements.txt
    └── tests
```

- **ameriflux-pipeline** : main git repo
- CHANGELOG : lists all changes made to the code with links to the issues.
- CONTRIBUTORS : lists the contributors of the repo
- NOTES : all decisions made is tagged and represented in code.
- README : user guide for installing and running the program.
- ameriflux_pipeline/ : runnable python modules (dir)
- docs/ : documentations directory
- requirements : list of required packages
- tests : directory for unit tests

Code Structure and Documentation

```
./ameriflux_pipeline/
├── __init__.py
├── config.py
├── data
├── eddypro
├── enveditor.py
├── master_met
├── met_data_processor.py
├── metprocessor.py
├── pipeline.py
├── post_pyfluxpro.py
├── pre_pyfluxpro.py
├── pyfluxpro
└── utils
```

- **ameriflux_pipeline** : runnable python modules
- __init__ : initialization file
- config : configuration file
- data : directory to store all data
- eddypro/ : module for eddypro processing
- master_met/ : module for processing master meteorological data
- pyfluxpro/ : module for pyfluxpro processing
- utils/ : directory for other utility functions to support processing
- metprocessor : GUI for raw meteorological data processing
- enveditor : GUI for user settings
- pipeline : GUI for modular execution of each processes.
- pre_pyfluxpro : module for pre-pyfluxpro processing
- post_pyfluxpro : module for post-pyfluxpro processing

Code Structure and Documentation

```
├─ master_met
│   ├─ __init__.py
│   └─ mastermetprocessor.py
```

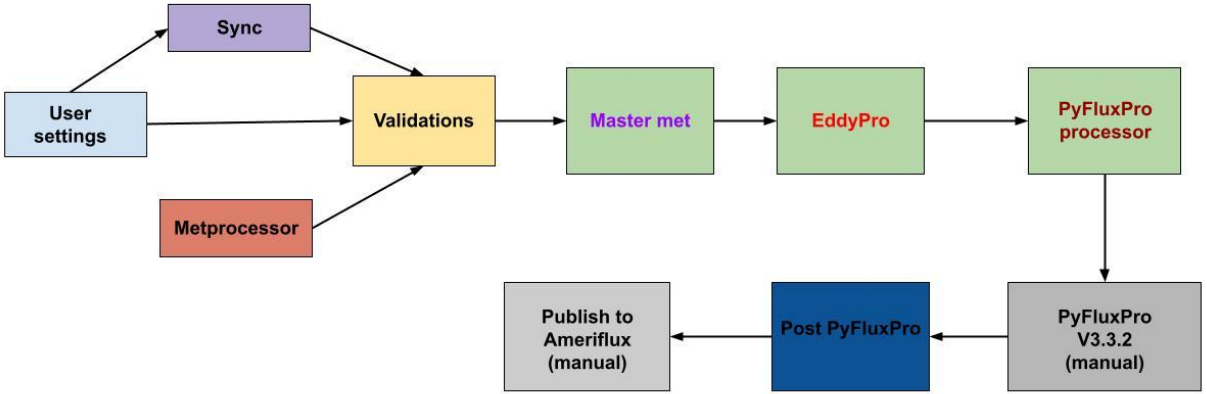
master_met/ directory : mastermet processor module.

- mastermetprocessor : python mastermet processor module

```
├─ eddypro
│   ├─ __init__.py
│   ├─ eddyproformat.py
│   └─ runeddypro.py
```

eddypro/ directory : eddypro processor module

- eddyproformat : creates eddypro input csv file
- runeddypro : code to run EddyPro software in a headless manner

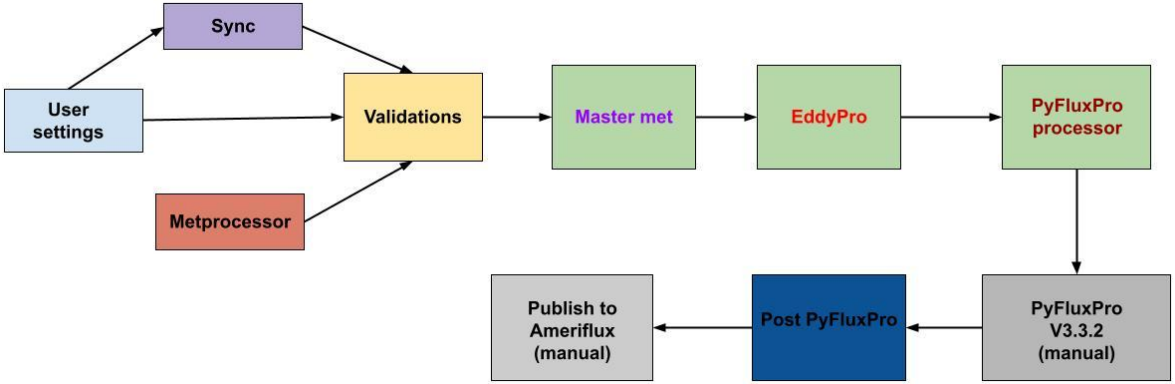


Code Structure and Documentation

```
├── pyfluxpro
│   ├── __init__.py
│   ├── amerifluxformat.py
│   ├── l1format.py
│   ├── l2format.py
│   ├── outputformat.py
│   └── pyfluxproformat.py
```

pyfluxpro/ directory : Pyfluxpro module

- pyfluxproformat : creates mainstem pyfluxpro input excel sheet
- amerifluxformat : creates pyfluxpro input excel sheet for ameriflux
- l1format : creates L1 control file for ameriflux
- l2format : creates L2 control file for ameriflux
- outputformat : creates ameriflux submission-ready csv file. This is called in post_pyfluxpro module.

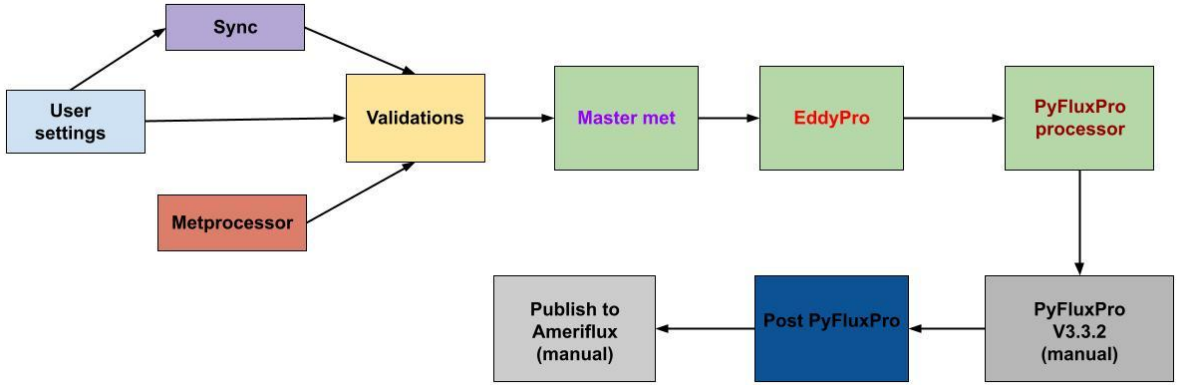


Code Structure and Documentation

utils/ directory : all utility functions that supports the pipeline

```
└── utils
    ├── __init__.py
    ├── data_util.py
    ├── input_validation.py
    ├── process_validation.py
    └── syncdata.py
```

- syncdata : Sync module. Can be turned on/off from enveditor.
- input_validation : validation module for all user inputs
- process_validation : validation module that verifies if each processes is executed as expected
- data_util : some functionality to support other modules



Code Structure and Documentation

```
├── docs
│   ├── eddypro
│   │   ├── eddyproformat.md
│   │   └── runeddypro.md
│   ├── enveditor.md
│   ├── master_met
│   │   ├── METEOROLOGICALDATA.md
│   │   └── mastermetprocessor.md
│   ├── metprocessor.md
│   ├── pipeline.md
│   ├── postpyfluxpro.md
│   ├── prepyfluxpro.md
│   ├── pyfluxpro
│   │   ├── amerifluxformat.md
│   │   ├── l1format.md
│   │   ├── l2format.md
│   │   ├── outputformat.md
│   │   └── pyfluxproformat.md
│   └── utils
│       ├── data_util.md
│       ├── input_validation.md
│       ├── process_validation.md
│       └── syncdata.md
```

docs/ directory :

- in-depth documentation on all modules and functions

THANK YOU

ILLINOIS

NCSA | National Center for
Supercomputing Applications