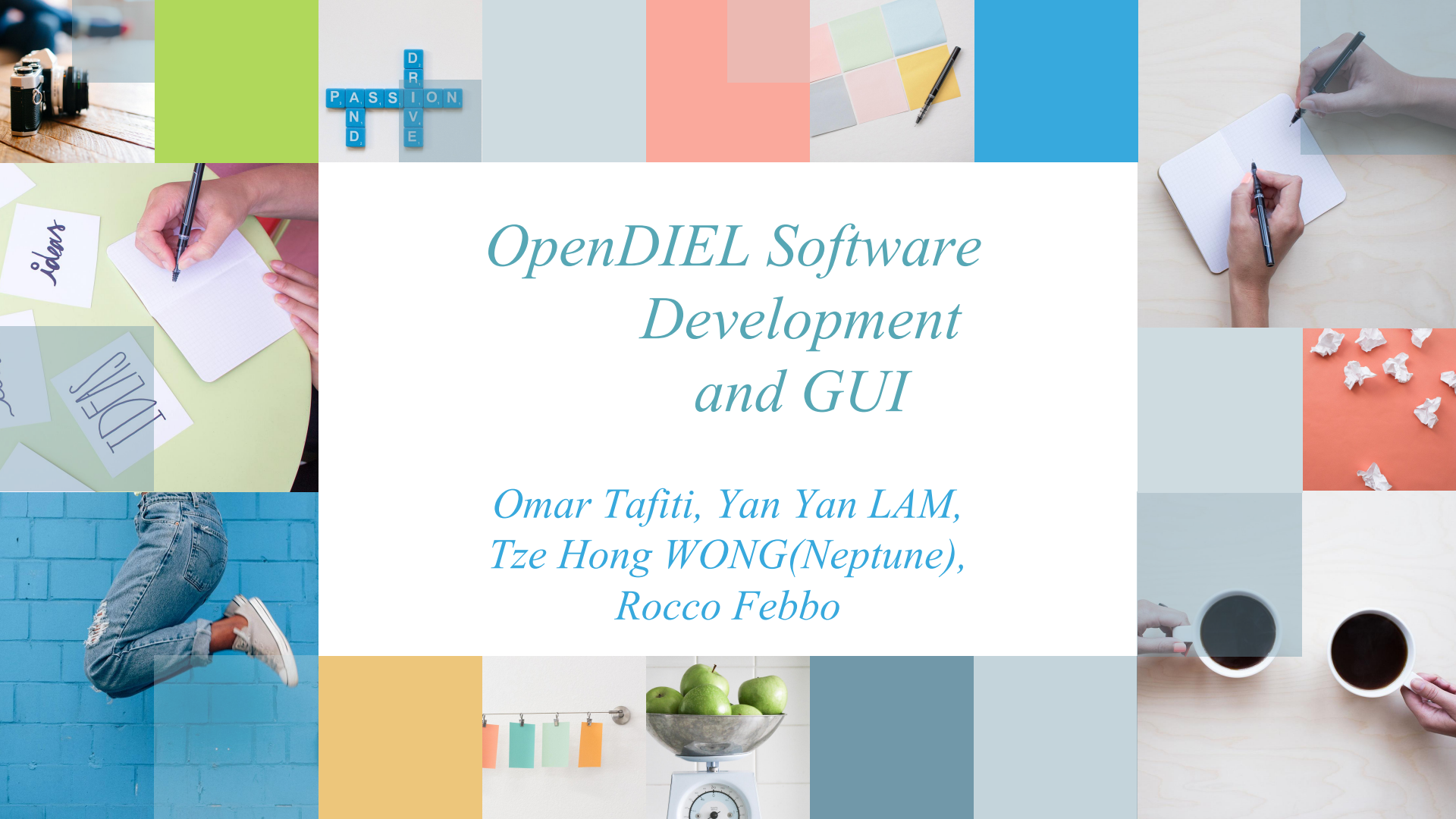
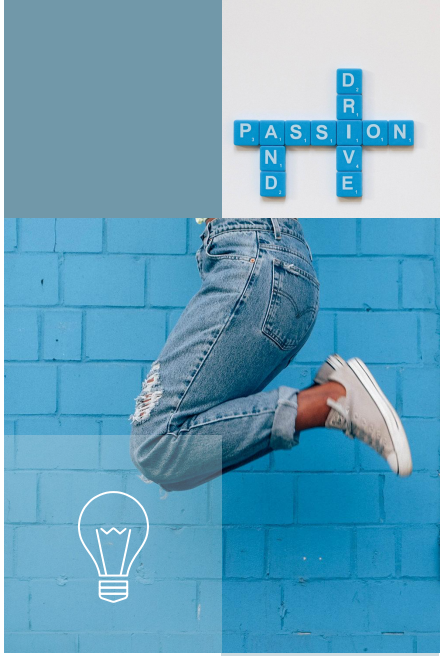


# OpenDIEL Software Development and GUI

*Omar Tafiti, Yan Yan LAM,*
*Tze Hong WONG(Neptune),*
*Rocco Febbo*

# How does openDIEL work?

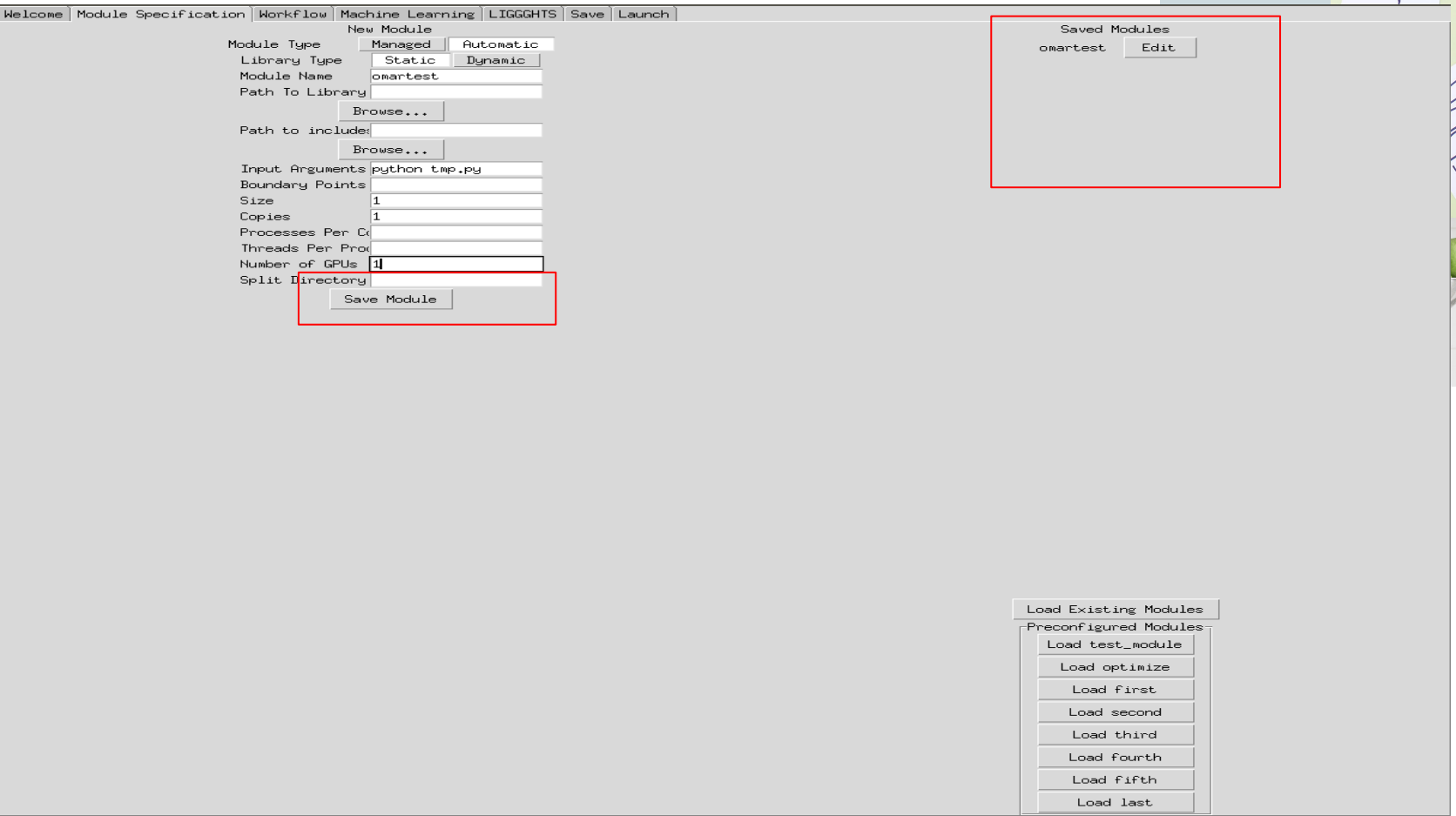
- openDIEL is a wrapper to schedule work on a set of resources (workflow engine)
- Create a driver program in C
- Create a makefile
- Create modules to be run
- Create workflow.cfg file
- Execute mpirun (calculate number of processes)



# Functionalities of GUI

- GUI seeks to eliminate some of the tasks in this tedious process
- Load module
- Create workflow
- Calculate number of processes
- Call mpirun system command

# GUI

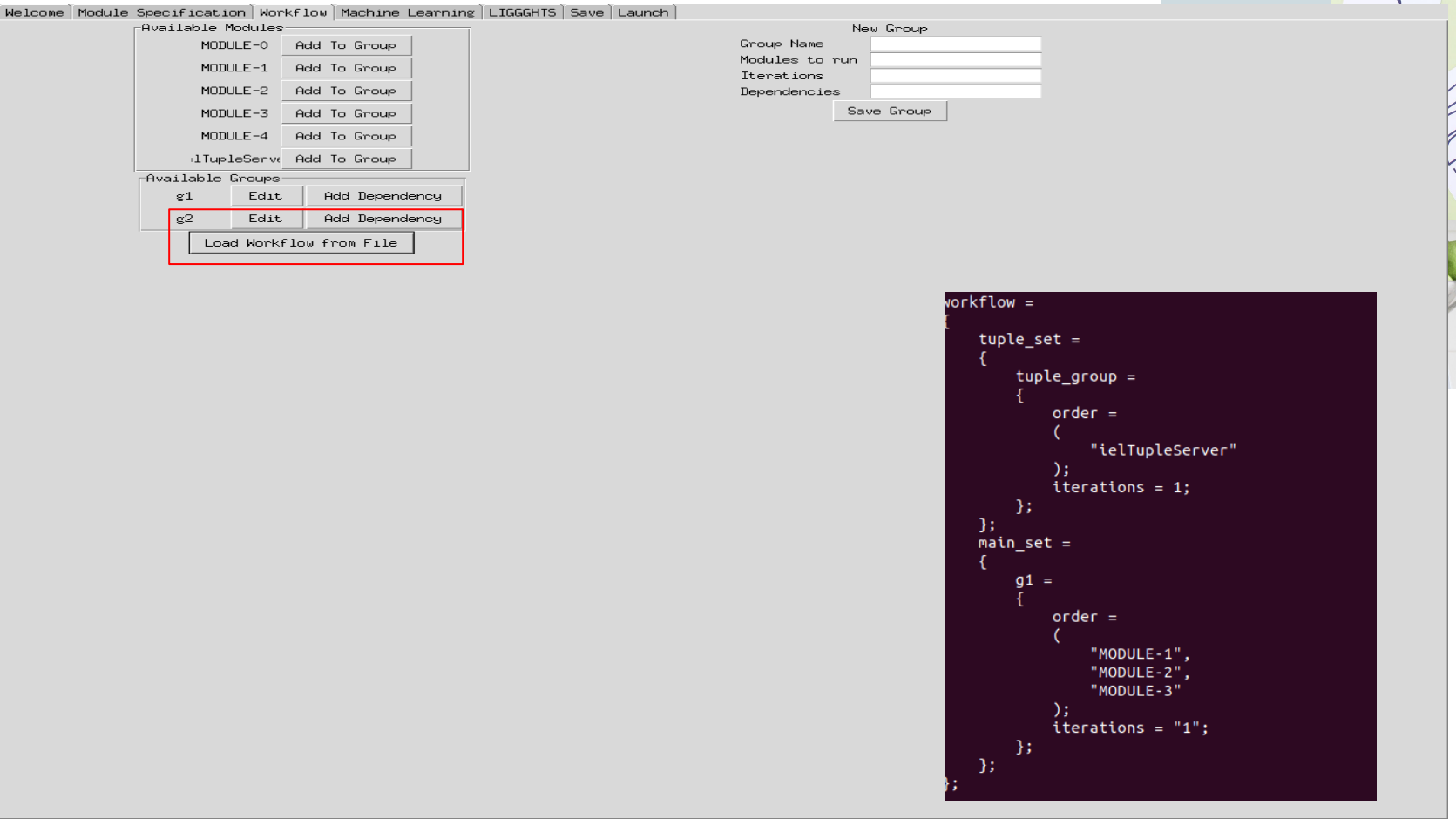

Welcome
Module Specification
Workflow
Machine Learning
LIGGGHTS
Save
Launch
New Module
Module Type
Managed
Automatic
Library Type
Static
Dynamic
Module Name
omartest
Path To Library
Browse...
Path to includes
Browse...
Input Arguments
python tmp.py
Boundary Points
Size
1
Copies
1
Processes Per Co
Threads Per Proc
Number of GPUs
1
Split Directory
Save Module
Saved Modules
omartest
Edit
Load Existing Modules
Preconfigured Modules
Load test_module
Load optimize
Load first
Load second
Load third
Load fourth
Load fifth
Load last

# GUI


Welcome
Module Specification
Workflow
Machine Learning
LIGGGHTS
Save
Launch
New Module
Module Type
Managed
Automatic
Library Type
Static
Dynamic
Module Name
Path To Library
Browse...
Path to includes
Browse...
Input Arguments
Boundary Points
Size
Copies
Processes Per Co
Threads Per Proc
Number of GPUs
Split Directory
Save Module
Saved Modules
MODULE-0
Edit
MODULE-1
Edit
MODULE-2
Edit
MODULE-3
Edit
MODULE-4
Edit
lTupleServe
Edit
Load Existing Modules
Preconfigured Modules
Load test_module
Load optimize
Load first
Load second
Load third
Load fourth
Load fifth
Load last


D
R
P A S S I O N
V
E

# GUI




```
workflow =
{
    tuple_set =
    {
        tuple_group =
        {
            order =
            (
                "ielTupleServer"
            );
            iterations = 1;
        };
    };
    main_set =
    {
        g1 =
        {
            order =
            (
                "MODULE-1",
                "MODULE-2",
                "MODULE-3"
            );
            iterations = "1";
        };
    };
};
```

# GUI

Welcome | Module Specification | Workflow | Machine Learning | LIGGGHTS | Save | Launch

CFG File Name: workflowAM.cfg

Create Configuration File

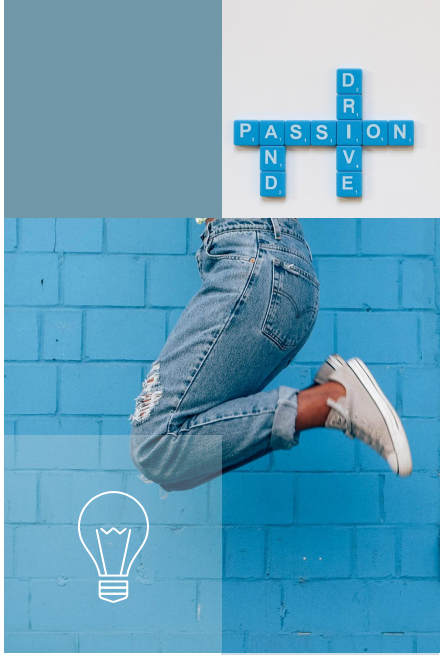
```
tuple_space_size = 0;
number_of_gpu = 1;
modules =
{
    MODULE-0 =
    {
        function = "MODULE-1";
        args =
        (
            "('../helloiexe',)"
        );
        size = 5;
        libtype = "static";
    };
    MODULE-1 =
    {
        function = "MODULE-1";
        args =
        (
            "../helloiexe"
        );
        size = 5;
        libtype = "static";
    };
    MODULE-2 =
    {
        function = "MODULE-3";
        args =
        (
            "('../helloifexe',)"
```

# GUI


Welcome
Module Specification
Workflow
Machine Learning
LIGGGHTS
Save
Launch
Example Workflows
Launch Fantest
Display Attribute Info.
Defined Workflow
Output Directory N
Output Directory Location
Browse...
Launch Job

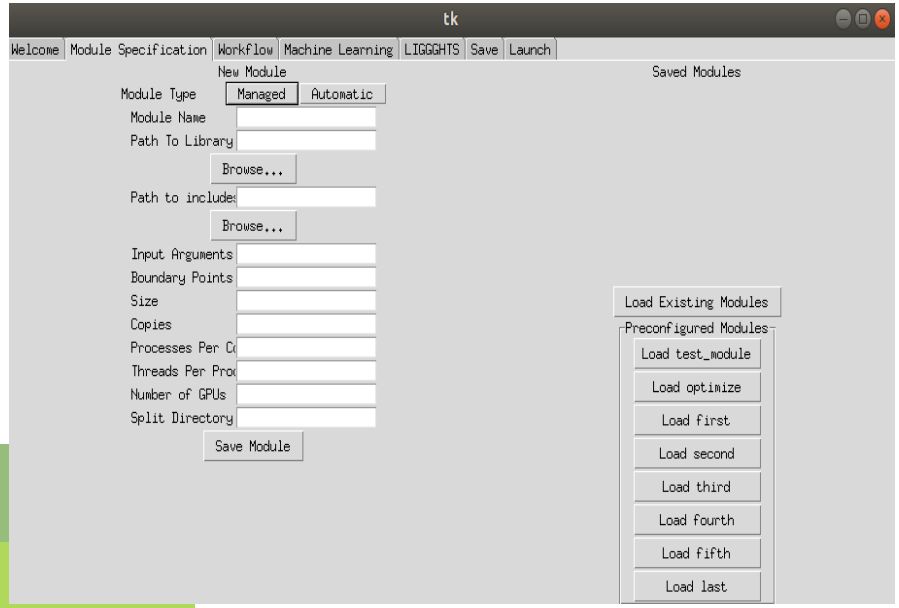
# Challenges we faced

- Syntax Errors
- Understanding the code
- Configuration file formatting
- Semantic Errors
- Loading modules created using the GUI
- Launching mpirun command from non-source directory
- Design issues

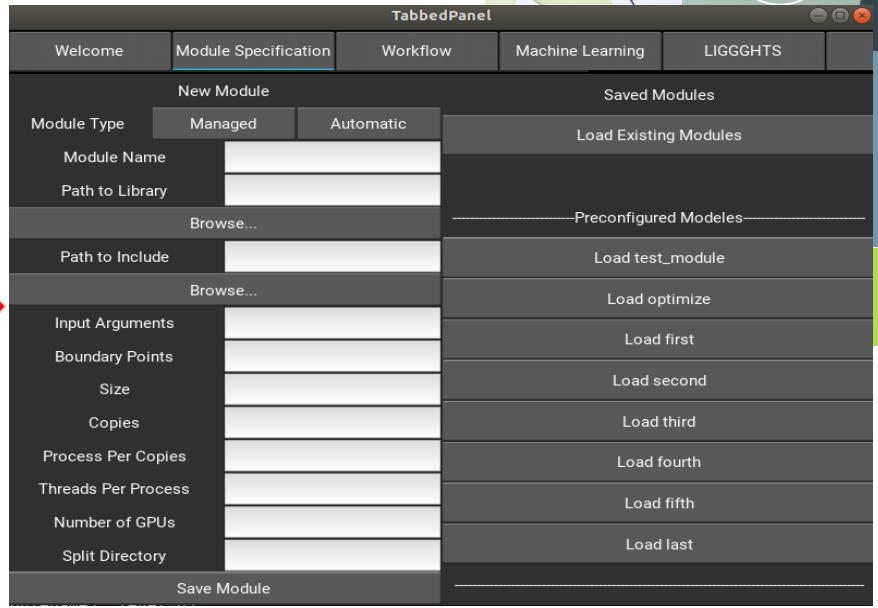


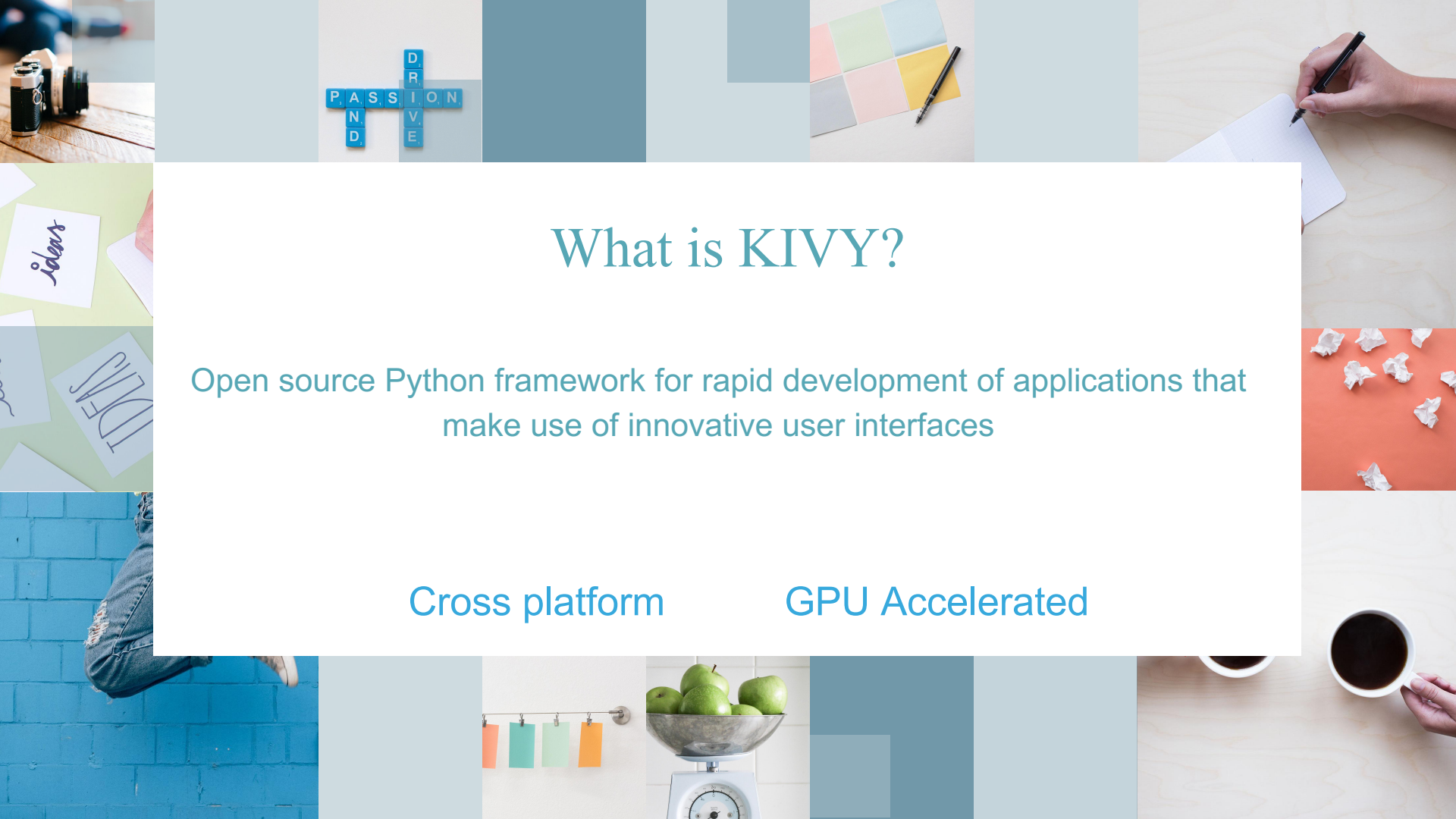
# Future Development


## Tkinter




## KIVY





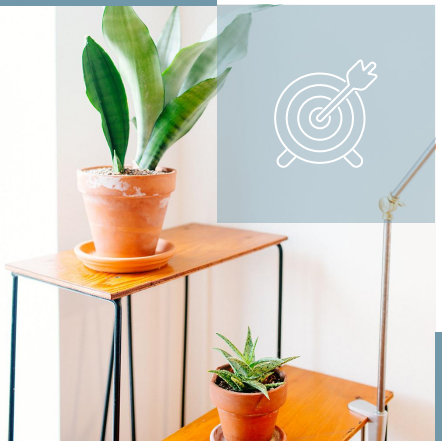
# What is KIVY?

Open source Python framework for rapid development of applications that make use of innovative user interfaces

Cross platform

GPU Accelerated

# Why KIVY?



Tkinter:

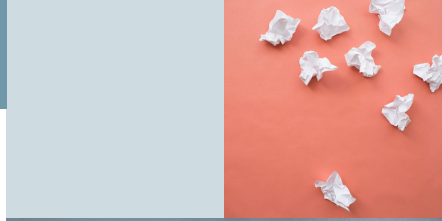
outdated

not visually appealing

KIVY :

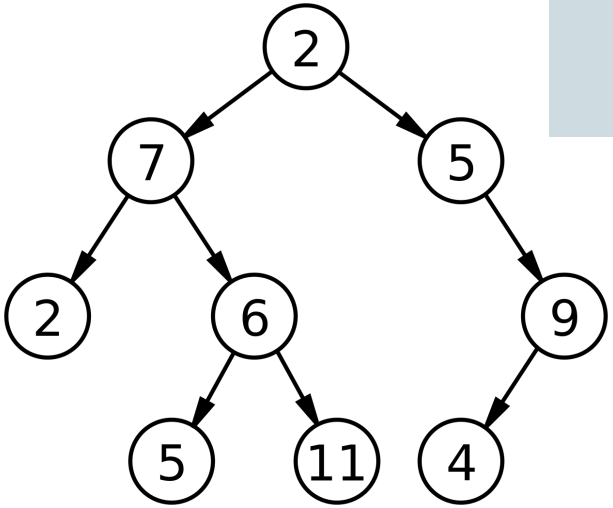
all intensive purposes —
>design language

mobile app development



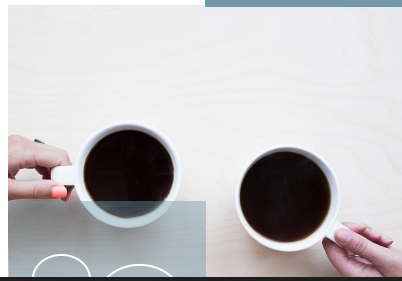
# How does KIVY work?

- Must be familiar with python
- KV language which allows you to create your widget tree in a declarative way
- bind widget properties to each other or to callbacks in a natural manner



# Differences in KIVY vs. Tkinter

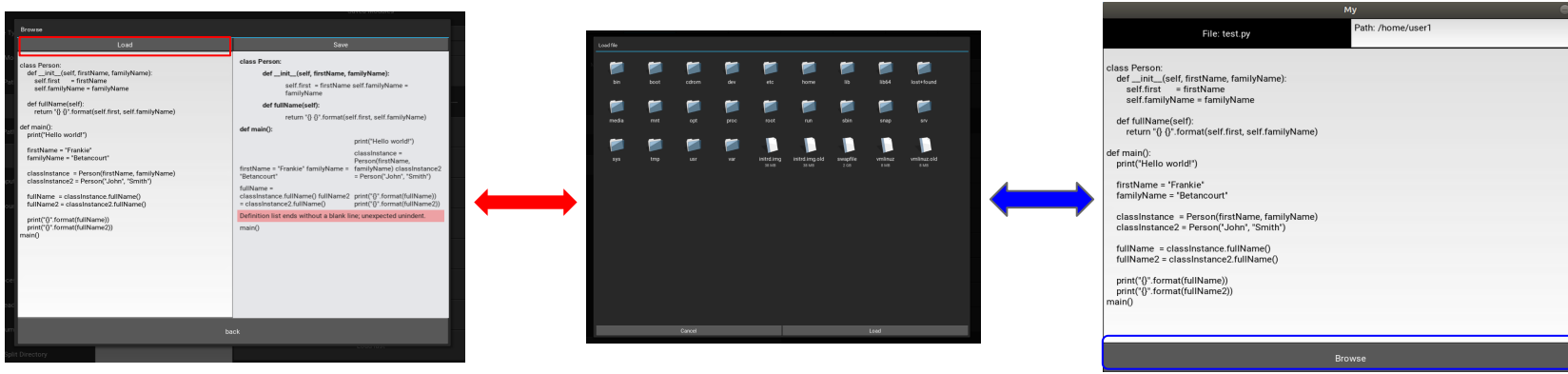
- Tkinter
  - desktop focused
- KIVY
  - automatically formats widgets to most appealing design
  - works across all platforms
  - special language for defining layout. This allows you to keep your logic and presentation separate. (Ex. CSS and HTML)



```
#:kivy 1.1.0

Root:
    text_input: text_input

    BoxLayout:
        orientation: 'vertical'
        BoxLayout:
            size_hint_y: None
            height: 30
            Button:
                text: 'Load'
                on_release: root.show_load()
            Button:
                text: 'Save'
                on_release: root.show_save()

        BoxLayout:
            TextInput:
                id: text_input
                text: ''

            RstDocument:
                text: text_input.text
                show_errors: True

<LoadDialog>:
    BoxLayout:
        size: root.size
        pos: root.pos
        orientation: "vertical"
        FileChooserIconView:
            id: filechooser

        BoxLayout:
            size_hint_y: None
            height: 30
            Button:
                text: "Cancel"
                on_release: root.cancel()

            Button:
                text: "Load"
                on_release: root.load(filechooser.path, filechooser.selection)

<SaveDialog>:
    text_input: text_input
    BoxLayout:
```

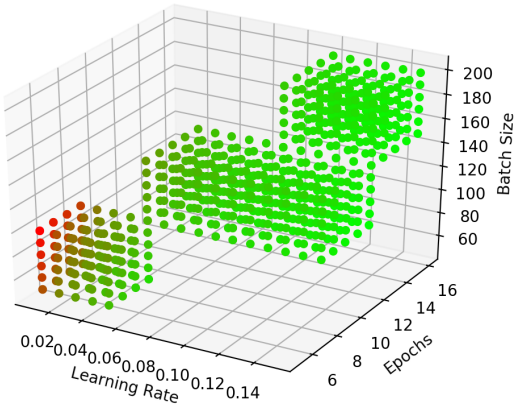
# How is Kivy going now

- GUI (done)
- functionality (currently working on it)
  - Hide and show widget (done)
  - browse file (finished, need improvement[bugs fixed, better searching engine & GUI]

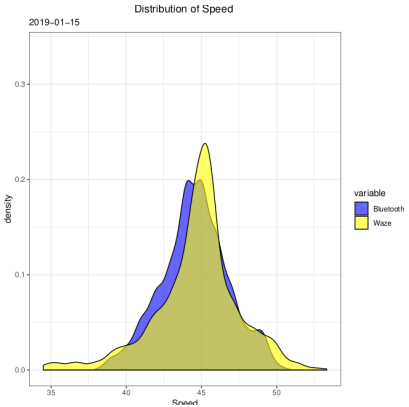



# Applications of openDIEL

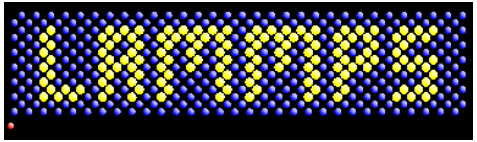
## Grid Engine with MagmaDNN



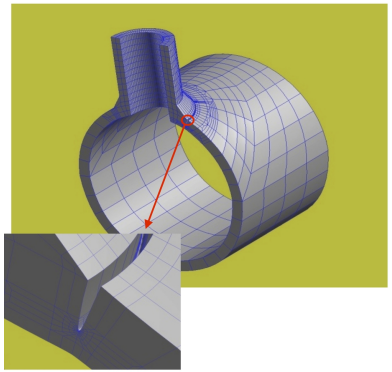

## Traffic Flow Data Analytics



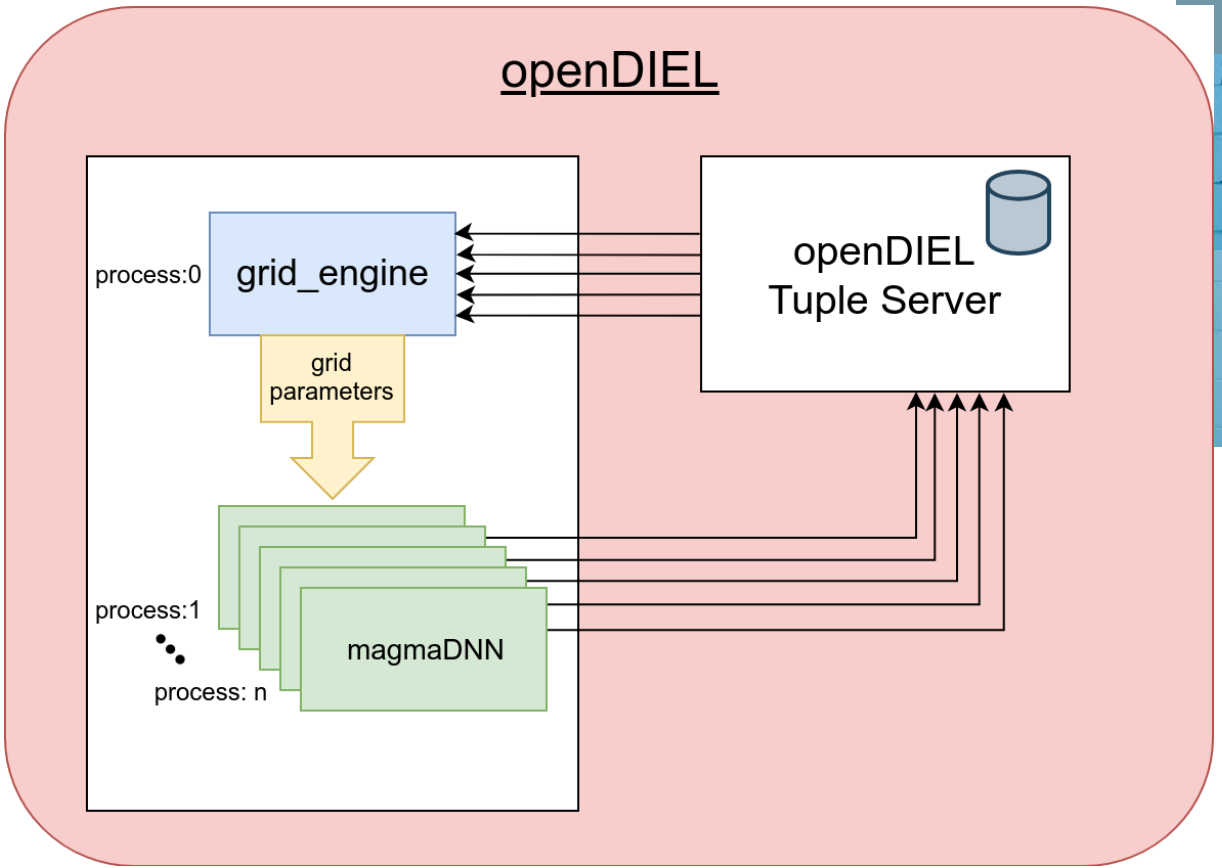

## Computational Chemistry of Epoxy System



## Computational Mechanics: Warp 3D

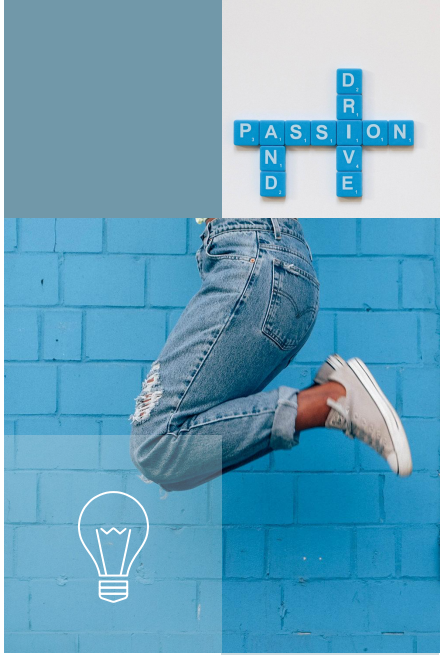


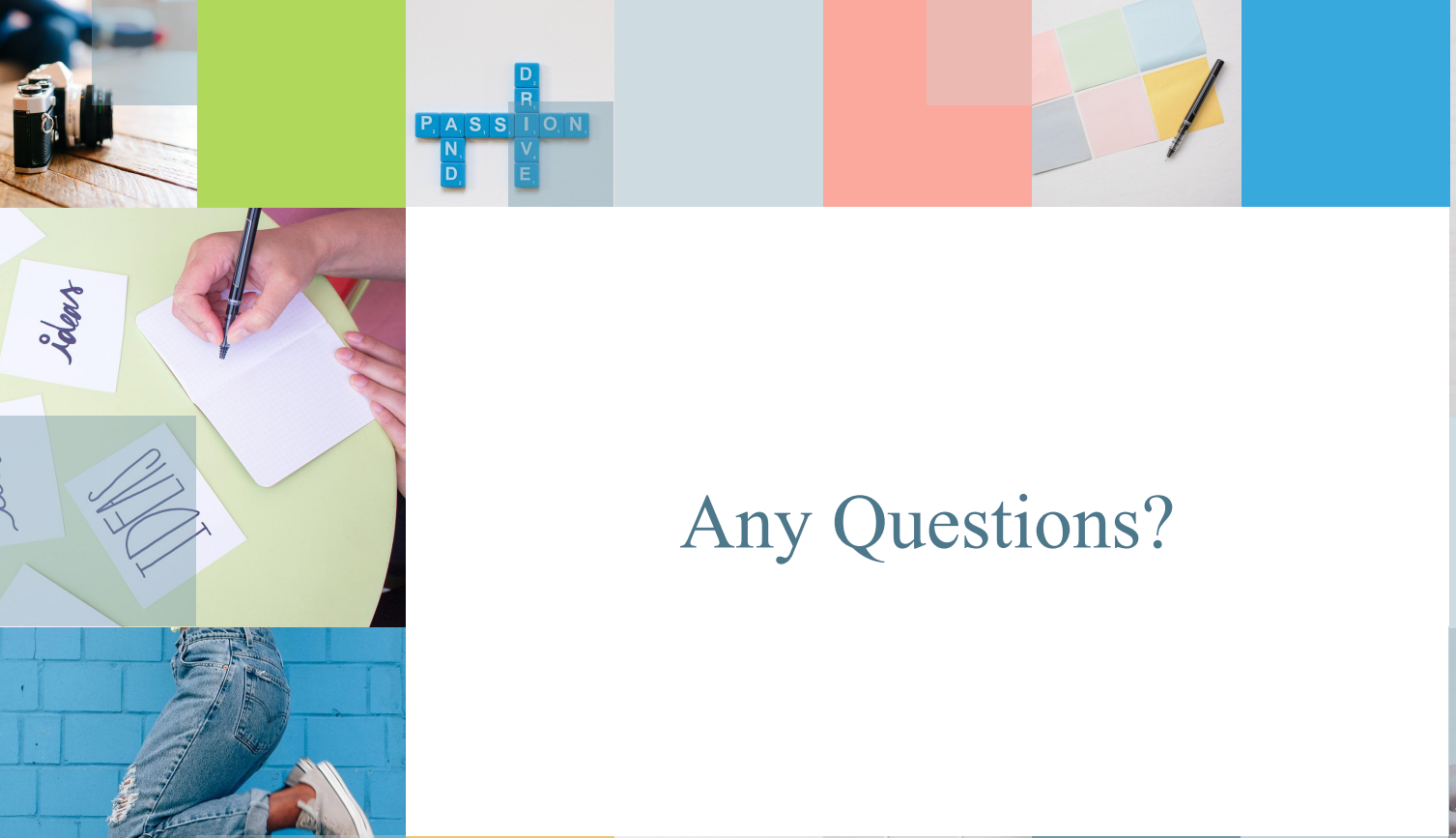
# Example Workflow: MagmaDNN Grid Engine




# Research Goal

Fully functional GUI on KIVY to provide a stylish, user-friendly platform to help use openDIEL






# Any Questions?