# GOING FROM MATLAB TO PYTHON
# A GENERIC WORKFLOW

08. OCTOBER 2018 I RAJALEKSHMI DEEPU,SANDRA DIAZ

# WHERE TO START?

- Identify the type of computation you are performing
- Identify computational steps in your code
- Do basic profiling on your original Matlab code
- Identify input / output
- If you need parallel, think parallel from the beginning

# WHAT TO DO NEXT?

- Take one step at a time and translate the functionality
- Build a unit test which allows you to compare to known results => your Matlab results
- Document your code as you write it
- Commit frequently
- Check the code performance
- Try to adhere to programming standards (PEP8): *Python Enhancement Proposal. (https://www.python.org/dev/peps/pep-0008/)*

# HOW TO IMPLEMENT YOUR STEPS?

- Each computation step has an input and an output
- Identify the interfaces among steps
- Identify the correct data structures in your computation
- Python provides a larger diversity of data structures (lists, dictionaries, arrays, matrices, etc…)

# HOW TO IMPLEMENT YOUR STEPS?

- Python has a large community of users. Look for modules which can make your computation easier.
- If you use Matlab commands from a specific toolbox, look for equivalent Python modules.
- Break down computation into functions, classes or even modules.
- Always think about clean / reusable / maintainable code

# DEBUGGING

- Python DeBugger (pdb)
  - Python's interactive source code debugger
  - Available as a module; *import pdb*

- Print statements
  - Better to use with a filename and line number.

```
10      if x > 23:
11          print "Debugging: my_file.py, line 11"
12          print "Hello!"
```

# Thank you for your attention!