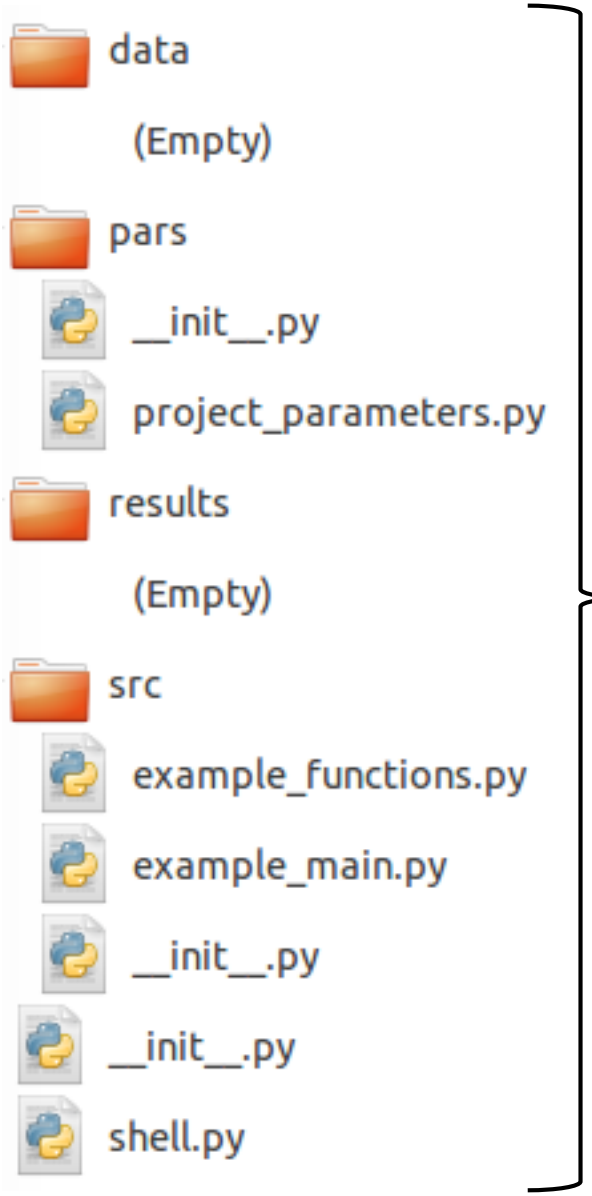
A basic project structure

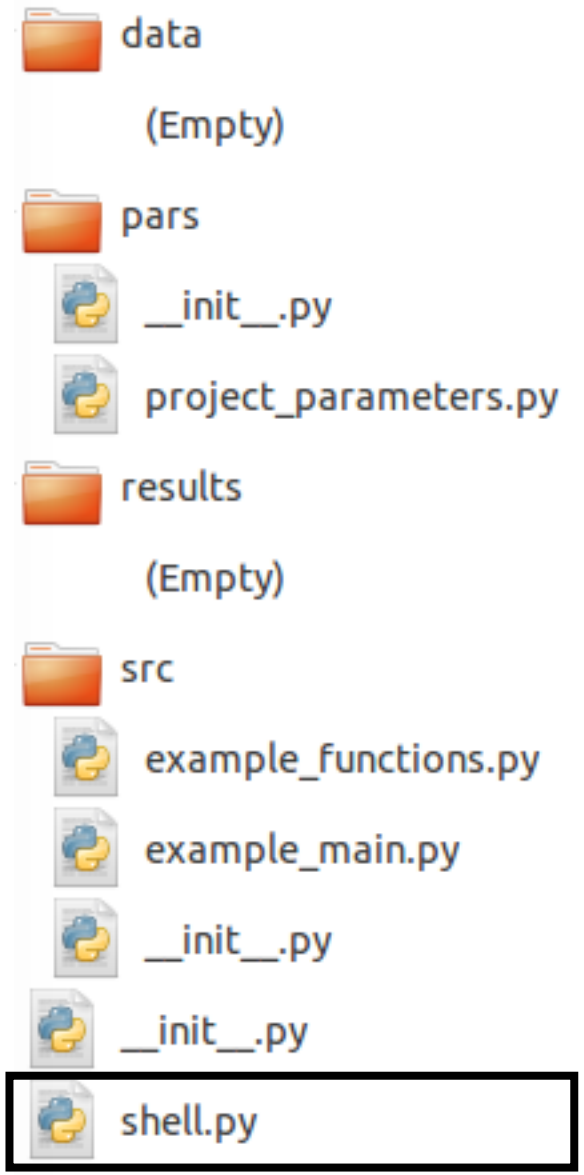
Philip Zurbuchen



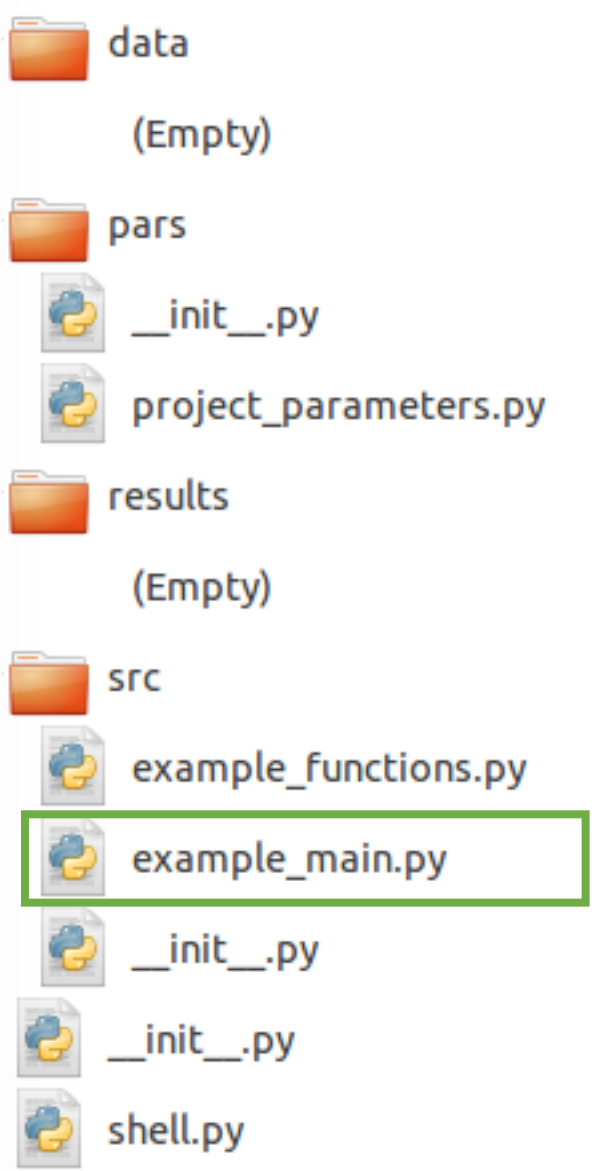
Main directory of my Project

- Add directory to sys.path. (See ‚shell.py‘ for code)
- Easily call modules.
- Always call functions from main directory
- __init__.py in source-code directories!

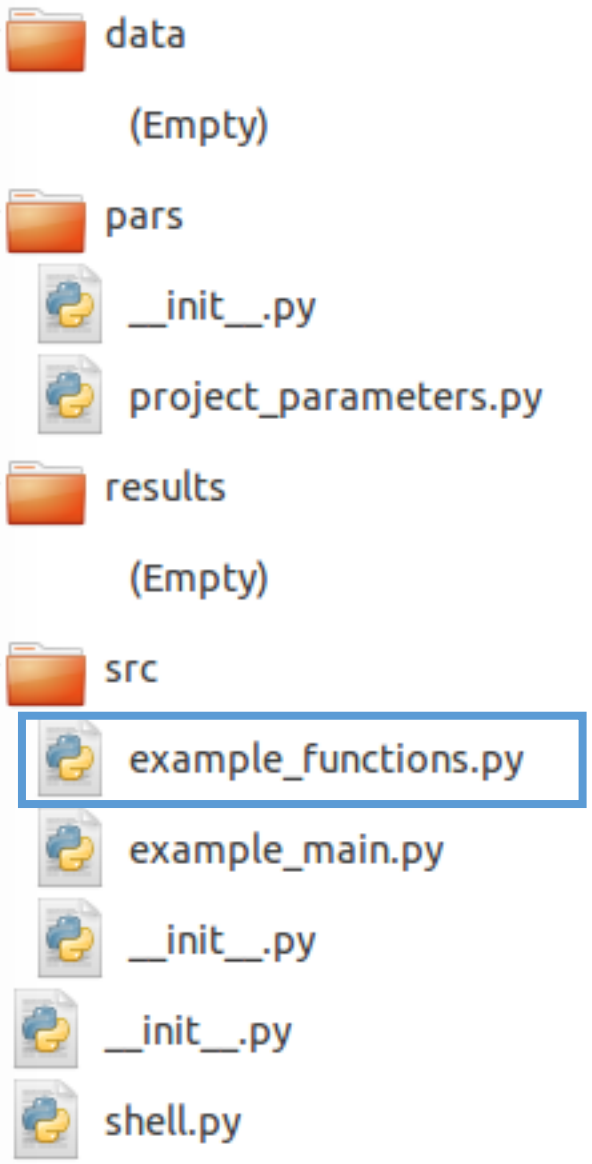
```
"""
Shell script, which calls all the main scripts (steps in the project).
Here, our first (and only step) is 'example'.
"""


"""
# On first execution, do this (Add current path to sys.path):
import sys
import os
my_path = os.path.abspath('')
if my_path not in sys.path:
    sys.path.append(my_path)
"""


# import from
from src.example_main import main_class as example_main_class

# make an instance of our class 'example'
example_inst = example_main_class()

# call the main function
example_inst.main()

# do the same for other examples
```



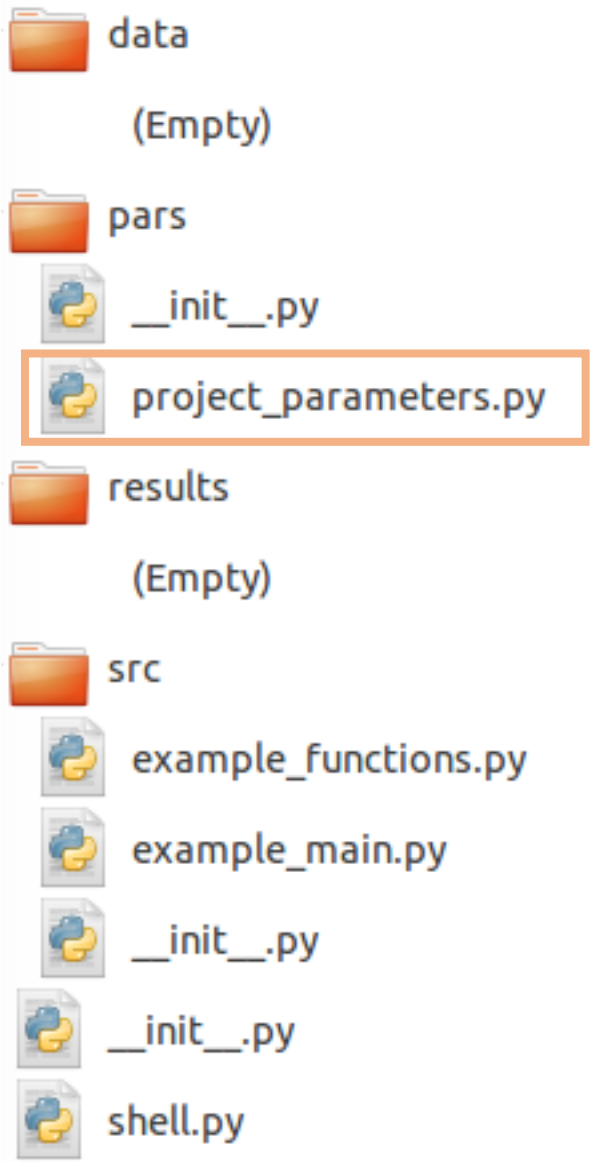
```python
# My class imports
from src.example_functions import funcs_class

# General imports
# none used in this example

class main_class(funcs_class):

    def __init__(self):

        # Get all methods AND parameters from funcs_class
        funcs_class.__init__(self)

    def main(self):

        # Evaluate square root
        self.square_root_of_parameter()

        # Print the result..
        self.print_result()
```



```python
# My class imports
from pars.project_parameters import params_class


# General imports
import math


class funcs_class(params_class):
    """
    The sub-functions for example_main.
    Keeping them in this sub-class keeps example_main neat and tidy.
    """


    def __init__(self):

        # get all parameters from pars/project_parameters
        params_class.__init__(self)

        # get parameters used in this step of the project ('example')
        self.example_parameters()



    def square_root_of_parameter(self):

        self.result = math.sqrt(self.value)



    def print_result(self):

        print "Result is :\n\t" + str(self.result)
```



```python
class params_class():
    """
    All the control parameters used in the project.
    """

    def __init__(self):

        # general parameters used in your whole project
        # ...
        pass

    def example_parameters(self):
        """
        parameters used in 'main'
        """

        self.value = 9

        # etc..


#   def get_pars_for_2nd_example(self):
        """
        more parameters for our next steps..
        (e.g. 2nd example)
        """
```



Pros

- Solid reproducability
- Flexibility
- Highly readable
- Easy path handling
- Practically no argument passing

Cons

- Not very basic (class inheritance)
- Keep variable names apart!
  - (Dictionaries..)
- Mpi4py a bit tricky
- Long _function scripts..?
  - Collapse functions