Separate Code into Several Files

Reasons to Separate

- Each team member do one part of the code, then combine
- Make code more readable / easy debugging
- Good for large program

Reasons to not Separate

- Code becomes more difficult to be debugged
- Program development time is shorter if not separate
- Good for small program

Empty / no __init__.py case

Python interpreter does not automatically know which folder to search for the module, with import statement

- Put __init__.py into that directory to tell that there is a module to search for

- Without putting the empty __init__.py, jupyter notebook & terminal python may create __pycache__ folder and make the code work too. (tested on ubuntu 17.04, python 3.6)

- __.init__.py can be an empty file, but should import other file

Files structure

- Package = directory name
  - Module that contain other module
- Module = file name (.py)

Import statement allow the search to occur

```
program/
    main.ipynb
    main_sub.py
    pkg/
        f_pk.py
        pkg1/
            f_pk1.py
            pk1a/
                f_pk1a.py
            pk1b/
                __init__.py
                f_pk1b.py
        pkg2/
            f_pk2.py
            pk2a/
                f_pk2a.py
            pk2b/
                f_pk2b.py
```

Case 2: Using __init__.py to import other pkg/modu

import parent.one

- Implicitly execute
  - parent/__init__.py
  - parent/one/__init__.py

```
parent/
    __init__.py
    one/
        __init__.py
    two/
        __init__.py
    three/
        __init__.py
```

https://docs.python.org/3/reference/import.html
https://docs.python.org/3.6/tutorial/modules.html

Content of __init__.py

May contain __all__ (list when want to import *)

Simplest method is to import each files / folders that are in the same path of __init__.py

Case 2: File structures

```
program/
    main.ipynb
    main_sub.py
    better_package/
        f_bpk.py
        __init__.py
        bpkg1/
            __init__.py
            f_bpk1.py
            f2_bpk1.py
            bpk1a/
                __init__.py
                f_bpk1a.py
            bpk1b/
                __init__.py
                f_pk1b.py
```

```
Program/
        better_package/
                bpkg2/
                        __init__.py
                        f_bpk2.py
                        bpk2a/
                                f_bpk2a.py
                                __init__.py
                        bpk2b/
                                f_bpk2b.py
                                __init__.py
```

Code in __init__.py

program/better_package/__init__.py

```
from . import f_bpk
from . import bpkg1
from . import bpkg2
```

If use just import f_bpk error will occur.

This is because import f_bpk means that f_bpk must be in the same folder of the main program, but it does not.

Dot “.” means from the directory that __init__.py is located

Dotdot “..” is for parent directory

Code in __init__.py

program/better_package/bpkg1/__init__.py

```
from . import f2_bpk1
from . import f_bpk1
from . import bpk1a
from . import bpk1b
```

program/better_package/bpkg1/bpk1a/__init__.py

```
from . import f_bpk1a
```

program/better_package/bpkg1/bpk1b/__init__.py

```
from . import f_bpk1b
```

Code in __init__.py

program/better_package/bpkg2/__init__.py

```
from . import bpk2a
from . import bpk2b
from . import f_bpk2
```

program/better_package/bpkg2/bpk2a/__init__.py

```
from . import f_bpk2a
```

program/better_package/bpkg2/bpk2b/__init__.py

```
from . import f_bpk2b
```

Import other jupyter notebook function

Need to run some code first, then import as usual.

- The code to be run is given in the notebook example

```
program2/
    main_load_nb.ipynb
    pkg/
        module_a.ipynb
```

program2/pkg/module_a.ipynb

```
def ma_hello():
    print('hello from module a')
```