Pythonでのプロジェクトを進めるあたっての基本

参考：https://rinatz.github.io/python-book/ch04-07-project-structures/

Python の理想のプロジェクト構成は Kenneth Reitz 氏によって推奨されている構成に従うのが良い。 どのような構成なのかは The Hitchhiker's Guide to Pythonというサイトの Structuring Your Project の章で詳しく記述されている。

ディレクトリの基本構成は次のとおりです。

```
(project)
├── (project)  ............ プログラムのソースコードディレクトリ
│   ├── __init__.py
│   └── *.py
└── tests  ................ 単体テストのソースコードディレクトリ
    ├── __init__.py
    └── *.py
```

ポイント：

- プログラムのソースコードは必ず (project) 配下の 1 つのディレクトリ内に集約させる
    - なぜならディレクトリはPythonのパッケージを構成するものなので、複数のディレクトリでソースコードを管理すると複数の Python パッケージを開発していることになる。しかし通常 1 つのプロジェクト内に複数のパッケージを含めて開発することはない。
- ディレクトリ内にPythonファイルを入れる場合は必ず **init**.py を用意する必要がある。
- ライブラリとアプリケーションの区別ができる構成にする。ライブラリとアプリケーションの区分は以下の通り。 | 構成 | 説明 | | -------- | ---------- | | ライブラリ | 他のプログラムから import して使うプログラム | | アプリケーション | 直接実行するプログラム |

コマンドラインからの実行時について

そして main() を実行したい場合はターミナル上で次のようにします。

```
$ python -m sample
```

(project)/**main**.py が実行される。

問題のある構成例 1

```
sample
├── sample
│   ├── __init__.py
```

```
│   └── *.py
├── libs  ................. よくないディレクトリ
│   ├── __init__.py
│   └── *.py
└── tests
    ├── __init__.py
    └── *.py
```

- 問題点
    - sample/**init**.py が用意されていない

問題のある構成例 2

```
sample
├── sample
│   ├── __init__.py
│   └── *.py
├── main.py
└── tests
    ├── __init__.py
    └── *.py
```

main.pyのスクリプト内部の構造

```
#!/usr/bin/env python

import sample

def main():
    # sample パッケージを使った処理
    ...

if __name__ == '__main__':
    main()
```

- 問題点
    - ライブラリとアプリケーションの区別ができていない構成になっている
    - この例では
        - sample：ライブラリ
        - main.py：アプリケーション

    という位置づけ。ライブラリとアプリケーションの両方の側面を持つプログラムを書きたいという場合は

```
main.py -> sample/__main__.py
```

という名前で保存する。