Python講座

第５週

内容

1. クラス

2. モジュール

3. パッケージ

内容

1. クラス

2. モジュール

3. パッケージ

クラス

- Pythonでもクラスがある

```
class クラス名:
    定義 ...
```

w5e1.py

データ

- クラスは属性とメソッドを持つ

```
class Sample:

    def __init__(self, var):
        self.var = var  # 属性

    def method(self):  # メソッド
        print(“method”)
```

w5e2.py

クラス変数

- クラス全体で共有されるデータ

```
// C++
class Sample {
   static int counter = 0;

   …
}
```

```
// Python
class Sample:
    counter = 0

    def __init__(self):
        …
```

w5e3.py

アクセス権限

- Pythonにはprivateやpublicなどのアクセス修飾子がない
- 全てのメンバはpublicなのでどこからでもアクセスできる
  - 慣例として「self._名前」はprivate

w5e4.py

getter/setter

- Pythonでgetterを用意するときは@propertyを使うと良いらしい

```
class Sample{
    def __init__(self):
        self._name = ‘sample’

    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, value):
        self._name = value
```

w5e5.py

クラス作成

- 複素数クラスを作成してみましょう
- 複素数クラスは他の複素数との和を計算した結果を新たな複素数クラスのインスタンスとして返すaddメソッドを持っています
- setterは実装せずにクラスを作成してください

継承

- クラスを継承して子クラスを作ることができる

```
class ChildClass(ParentClass):
    定義 …
```

w5e6.py

内容

1. クラス

2. モジュール

3. パッケージ

モジュール

- 少し複雑なコードを書くときにはいくつかのスクリプトに分割したい

- それぞれのスクリプトをモジュールと呼ぶ.

import文

```
import　モジュール名
from モジュール名 import 関数名など
```

- 他のモジュール内のクラスや関数を使いたいときはimportをする必要がある

- モジュールから特定のクラスや関数だけをimportしたい場合はfrom ~ import ~ を使う

- from ~ import ~ の場合，呼び出し時にモジュール名を省略できる

w5e7.py

モジュール

- 自作した場合のモジュール名はファイル名
- 例: w5e7.py　→ モジュール名はw5e7
- 自作モジュールの場合も「import　モジュール名」でOK

w5e8.py

内容

1. クラス

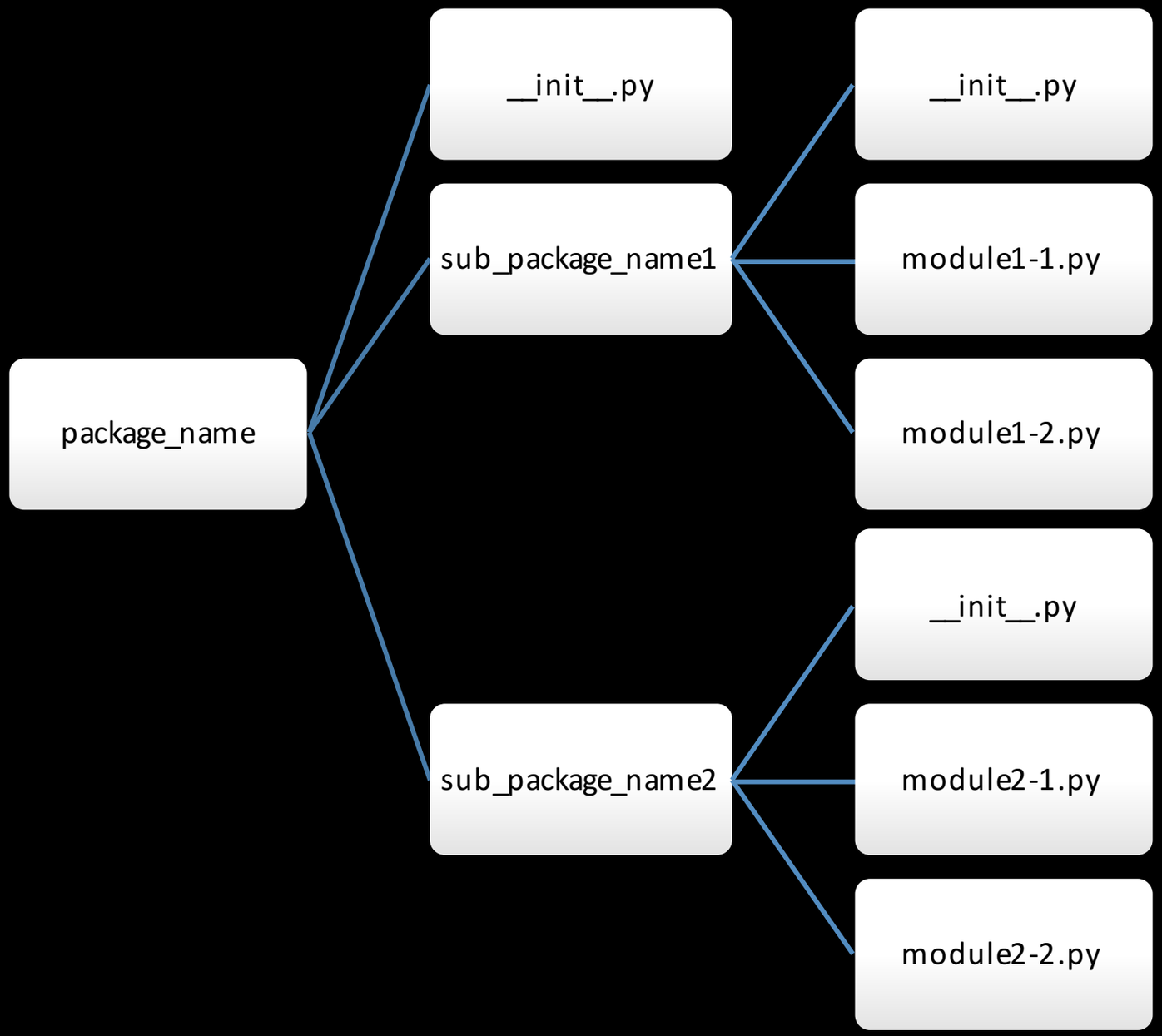
2. モジュール

3. パッケージ

パッケージ

- いくつかのモジュールをまとめたものをパッケージと呼ぶ
- パッケージ名はフォルダ名と等しい
- フォルダをパッケージと認識させるためには
  「__init__.py」が必要

パッケージ


__init__.py
__init__.py
sub_package_name1
module1-1.py
package_name
module1-2.py
__init__.py
sub_package_name2
module2-1.py
module2-2.py

パッケージ

```
import package.sub_package1.module1-1
from package import sub_package1.module1-1
from package import *    # 推奨されない
```

- パッケージからモジュールをimportする場合も「import」を使う

- 「from ~ import ~」も使うことができる