

オブジェクト指向 プログラミング

クラス，インスタンス，メソッド

クラスの作成

Sample2_4_Object/sample_class.py

```
2
3 class Student:
4     def __init__(self, id=0, name='none'):
5         self.id = id
6         self.name = name
7
8     def info(self):
9         print(f'学籍番号:{self.id:04d}\t氏名:{self.name}')
10
11
12 a = Student()
13 b = Student(1234, 'taro')
14
15 print(f'id:{a.id}, name:{a.name}')
16
17 a.info()
18 b.info()
19
```

クラス定義

初期化メソッド

インスタンス変数

メソッド

```
id:0, name:1234
学籍番号:0000    氏名:none
学籍番号:1234    氏名:taro
```

クラス変数とインスタンス変数

Sample2_4_Object/sample_class.py

```
1
2 class Student:
3     count = 0
4
5     def __init__(self, id=0, name='none'):
6         self.id = id
7         self.name = name
8         Student.count += 1
9
10    def info(self):
11        print(f"学籍番号:{self.id:04d}\t氏名:{self.name}, {Student.count}名が登録されています。")
12
13 a = Student()
14 b = Student(1234, "taro")
15
16 print(f"id:{a.id}, name:{a.name}")
17
18 a.info()
19 b.info()
20
21 c = Student(2356, 'hanako')
22 c.info()
23 a.info()
24
```

クラス変数

インスタンス変数

クラス変数

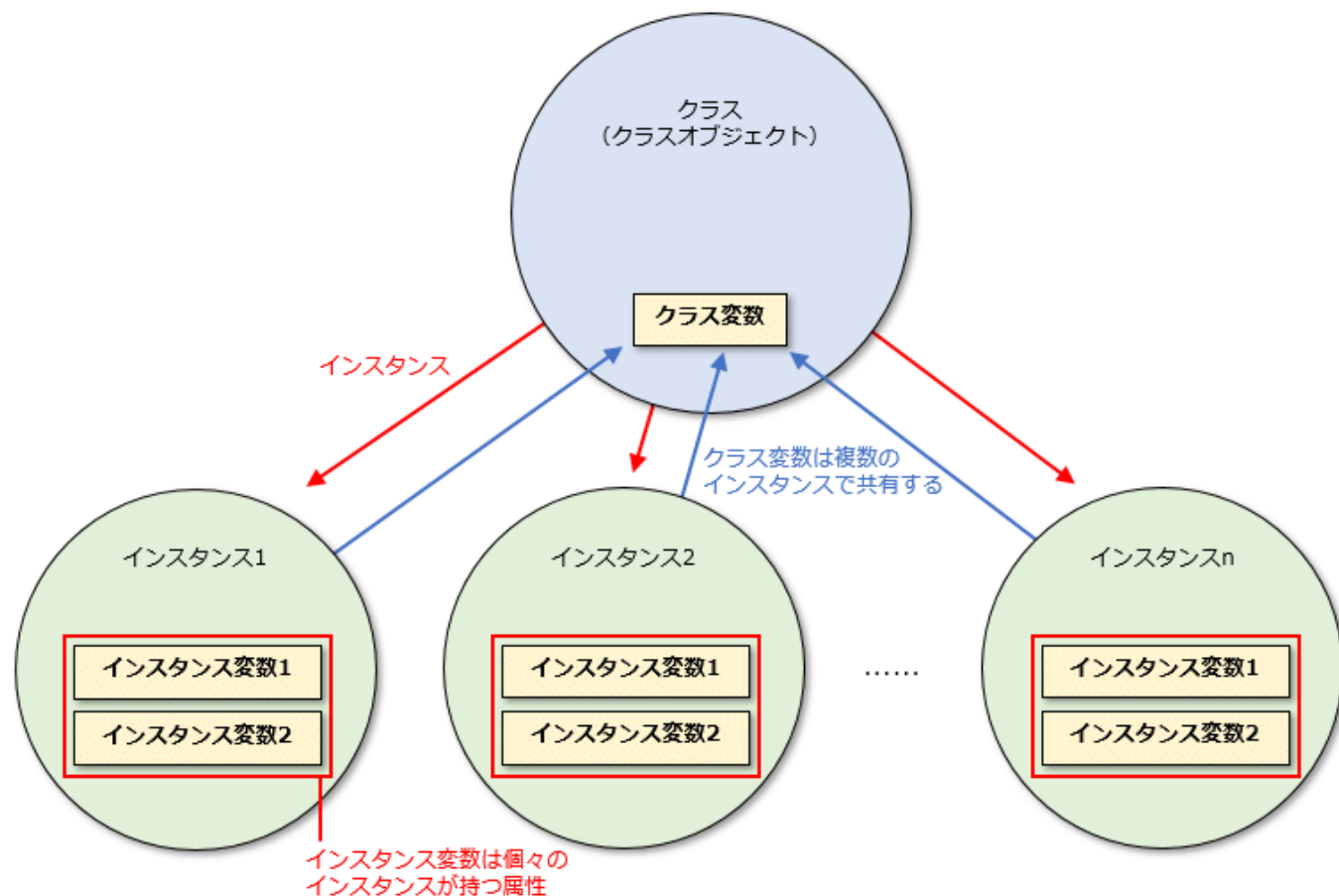
インスタンス変数

クラス変数を扱う場合,
[クラス名].[クラス変数名]

出力

```
id:0, name:none
学籍番号:0000 氏名:none, 2名が登録されています.
学籍番号:1234 氏名:taro, 2名が登録されています.
学籍番号:2356 氏名:hanako, 3名が登録されています.
学籍番号:0000 氏名:none, 3名が登録されています.
```

クラス変数とインスタンス変数



クラスの継承

Sample2_4_Object/sample_class2.py

```
1
2 ✓ class Student:
3     count = 0
4
5 ✓     def __init__(self, id=0, name='none'):
6         self.id = id
7         self.name = name
8         Student.count += 1
9
10 ✓     def info(self):
11         print(f"学籍番号:{self.id:04d}\t氏名:{self.name}, {Student.count}名が登録されています。")
12
13
14 ✓ class InternationalStudent(Student): #Studentクラスを継承
15     """留学生の場合"""
16
17 ✓     def __init__(self, id, name, nationality='none'):
18         super().__init__(id, name) #基底クラスの初期化関数を実行
19         self.nationality = nationality
20
21 ✓     def info(self):
22         print(f"学籍番号:{self.id:04d}\t氏名:{self.name} 国籍:{self.nationality}, {Student.count}名が登録されています。")
23
```

Studentクラスを継承した新しいクラス定義

infoメソッドをオーバーライド（上書き）

クラスの継承

24) リックしてブレークポイントを追加します。

```
25 a = Student()  
26 b = Student(1234, "taro")  
27  
28 a.info()  
29 b.info()  
30  
31 c = Student(2356, 'hanako')  
32 c.info()  
33 a.info()  
34  
35 d = InternationalStudent(555, 'john', 'USA')  
36 d.info()  
37  
38 b.info()
```

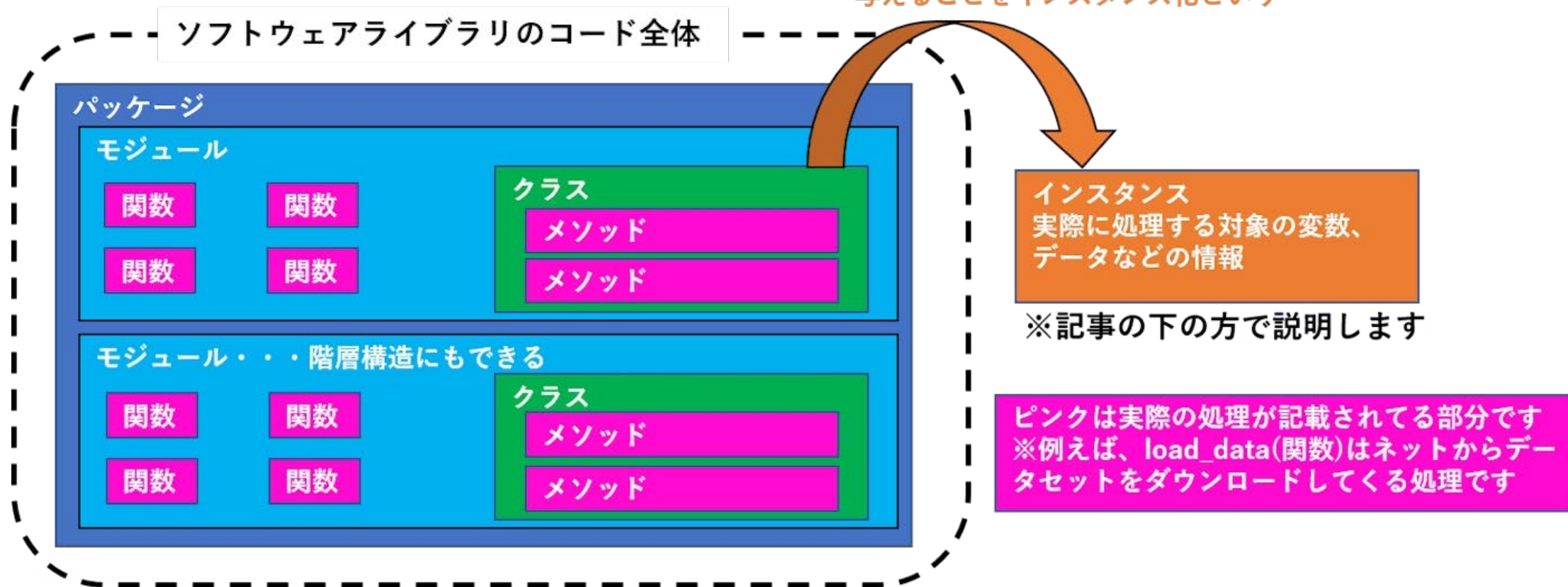
出力

学籍番号:0000	氏名:none, 2名が登録されています.
学籍番号:1234	氏名:taro, 2名が登録されています.
学籍番号:2356	氏名:hanako, 3名が登録されています.
学籍番号:0000	氏名:none, 3名が登録されています.
学籍番号:0555	氏名:john 国籍:USA, 4名が登録されています.
学籍番号:1234	氏名:taro, 4名が登録されています.

ソフトウェアライブラリ

クラス, モジュール, パッケージ

クラスに対して、実際に処理する対象(変数、データ)を与えることをインスタンス化という



2. パッケージ、モジュール、関数の例

(例) tensorflow



モジュールが階層構造になっている

こっちの表記の方が分かりやすいかも

```
tensorflow
├── keras
│   ├── datasets
│   │   ├── mnist
│   │   │   └── load_data
│   │   ├── boston_housing
│   │   ├── cifar10
│   │   └── etc...
│   ├── activations
│   ├── layers
│   └── etc...
├── initializer
├── nn
└── etc...
```

tf.keras.datasets.mnist

ドットは上図において、1つ下の階層に行くことを意味する

TensorFlow Core v2.1.0

概要 Python JavaScript C++ Java

Python r2.0

- tf
- tf.audio
- tf.autograph
- tf.bitwise
- tf.compat
- tf.config
- tf.data
- tf.debugging
- tf.distribute
- tf.dtypes
- tf.errors
- tf.estimator
- tf.experimental
- tf.feature_column
- tf.graph_util
- tf.image
- tf.initializers
- tf.io
- tf.keras
 - Overview
 - Input
 - Model
 - Sequential
 - activations
 - applications
 - backend

TensorFlow > API > TensorFlow Core v2.1.0 > Python

Module: tf.keras.datasets

TensorFlow 1 version

Public API for tf.keras.datasets namespace.

Modules

- `boston_housing` module: Boston housing price regression dataset.
- `cifar10` module: CIFAR10 small images classification dataset.
- `cifar100` module: CIFAR100 small images classification dataset.
- `fashion_mnist` module: Fashion-MNIST dataset.
- `imdb` module: IMDB sentiment classification dataset.
- `mnist` module: MNIST handwritten digits dataset.
- `reuters` module: Reuters topic classification dataset.

①tensorflow全てのパッケージ、モジュールの階層情報が一覧で表示されている。

②現在開いているtf.keras.datasetsはモジュールであることが示されている。

③現在開いているtf.keras.datasetsよりも1つ下の階層にはどのようなモジュールがあるか

自作モジュールの活用

ほかのソースコードファイルの関数を
import して利用する

sierpinski1.py

```
import turtle as t
```

triangle

sierpinski

```
t.shape('turtle')  
sierpinski(100)  
t.mainloop()
```

test_sp1.py

```
from sierpinski1 import sierpinski
```

余計なものがついてきて実行される

```
sierpinski(200)
```

```
t.mainloop()
```

`if __name__ == '__main__':`

- 定義したクラスや関数を、別のプログラムコードから呼び出して使う場合、クラスや関数の定義部と、動作確認のための記述部を区別するために利用する。

