第8回 Python における実装：継承、クラス変数とインスタンス変数

2021年6月8日

オブジェクト指向型言語

東京都立大学　システムデザイン学部

情報科学科　西内信之

【授業計画・内容授業方法】(変更後：予定)

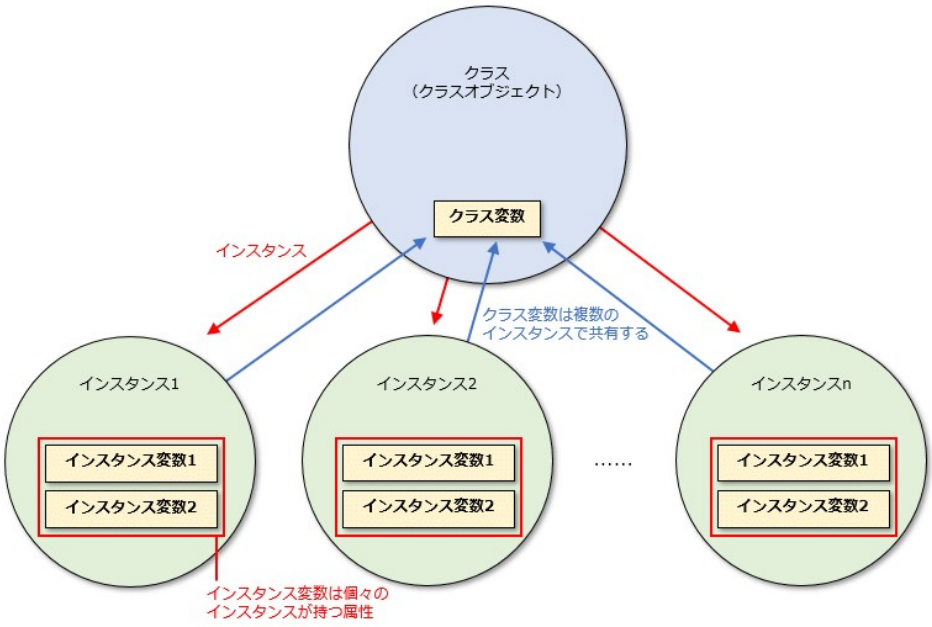
1. イントロダクション（ガイダンス）
2. オブジェクト指向の歴史・特徴・概念
3. クラス、継承、ポリモーフィズム
4. Python における実装（基礎、変数、条件、繰り返し）
5. Python における実装（データの型）
6. Python における実装（関数）
7. Python における実装（クラス）
8. **Python における実装（継承、クラス変数とインスタンス変数）**
9. 中間試験（レポート）
10. Python における実装（標準ライブラリ、外部ライブラリ）
11. Python における実装（tkinter）
12. Pythonで作るゲーム
13. オブジェクト指向開発と設計1
14. オブジェクト指向開発と設計2
15. 期末試験（レポート）・まとめ

1. Pythonにおける実装（クラス変数とインスタンス変数）

クラス内に定義できる変数：
クラス変数とインスタンス変数

クラス変数：すべてのインスタンス間で共通した値をもつ変数。
どのインスタンスからでも、クラスからでもどちらからでも呼び出し可能。インスタンスが異なっても同じ値を取得可能。

インスタンス変数：それぞれのインスタンスごとに独立した変数。
インスタンスを複数作成すればそれぞれ異なる値を持つ。



クラス内でのアクセス方法（インスタンスを作る前）

```
class MyClass1:
    x = 123

    def __init__(self):
        self.y=456
        print(self.x)
        print(self.y)
```

クラス変数

インスタンス変数

self.クラス変数

self.インスタンス変数

```
a=MyClass1()
```

```
123
456
```

クラス外部からのアクセス方法

```
class MyClass1:
    x = 123

    def __init__(self):
        self.y=456
        print(self.x)
        print(self.y)
```

クラス変数

インスタンス変数

```
a=MyClass1()
```

インスタンス化

```
123
456
```

```
MyClass1.x
```

クラス名.クラス変数

```
123
```

```
a.x
```

インスタンス名.クラス変数

```
123
```

```
a.y
```

インスタンス名.インスタンス変数

```
456
```

インスタンス変数とクラス変数

変数の種類	クラス外部からのアクセス方法	クラス内でのアクセス方法
インスタンス変数	インスタンス.インスタンス変数	self.インスタンス変数
クラス変数	クラス.クラス変数 インスタンス.クラス変数	self.クラス変数 （値を読み出すときだけ使う）

クラス外部からのインスタンス変数の追加

```
class MyClass1:
    x = 123

    def __init__(self):
        self.y=456
        print(self.x)
        print(self.y)
```

```
a=MyClass1()
```

インスタンス化

```
123
456
```

```
a.z=789
```

インスタンス変数zの追加

```
a.x
```

```
123
```

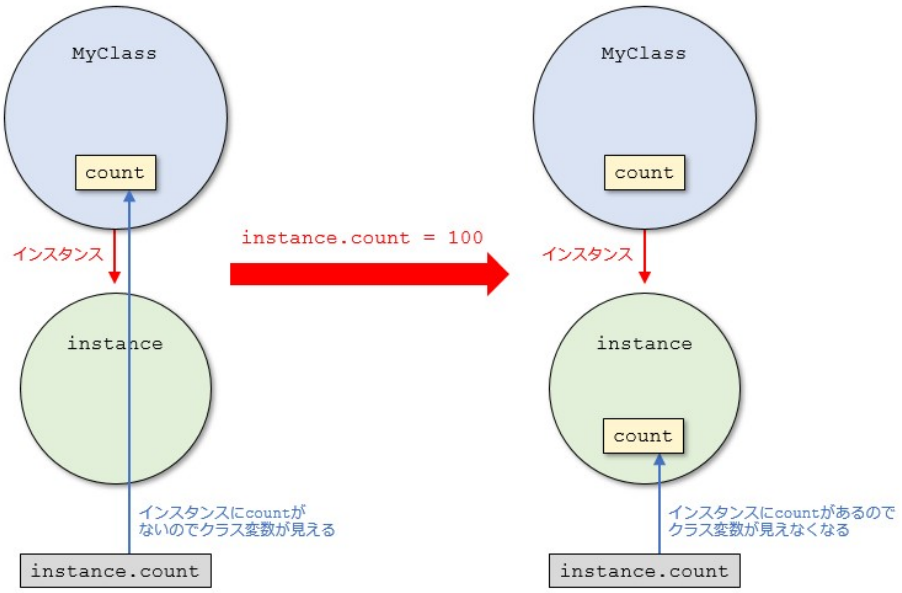
```
a.y
```

```
456
```

```
a.z
```

インスタンス変数zの表示

```
789
```



クラス変数にもインスタンス変数にも値がある場合は、インスタンス変数を優先して参照

分かりやすいコードを書くために

**「インスタンスで
クラス変数と同名のインスタンス変数を
作らないようにする」**

**「クラス内で
クラス変数と同名のインスタンス変数を
作らないようにする」**

▼クラス内の変数を使う
　クラス内で定義した変数「x=123」をメソッドの中で呼び出すには「self.x」とする必要がある

```
class MyClass3:
    x = 123 ←――――――― クラス内の変数

    def method1(self):
        print(self.x) ←――――― 注意

mycls = MyClass3()
mycls.method1()

#出力
# 123
```

▼グローバル変数を使う
　classの外で定義された変数を使う場合は、selfは不要

```
x = 123 ←――――――― グローバル変数

class MyClass3:

    def method1(self):
        print(x) ←――――― 注意

mycls = MyClass3()
mycls.method1()

#出力
# 123
```

2. Pythonにおける実装（継承）

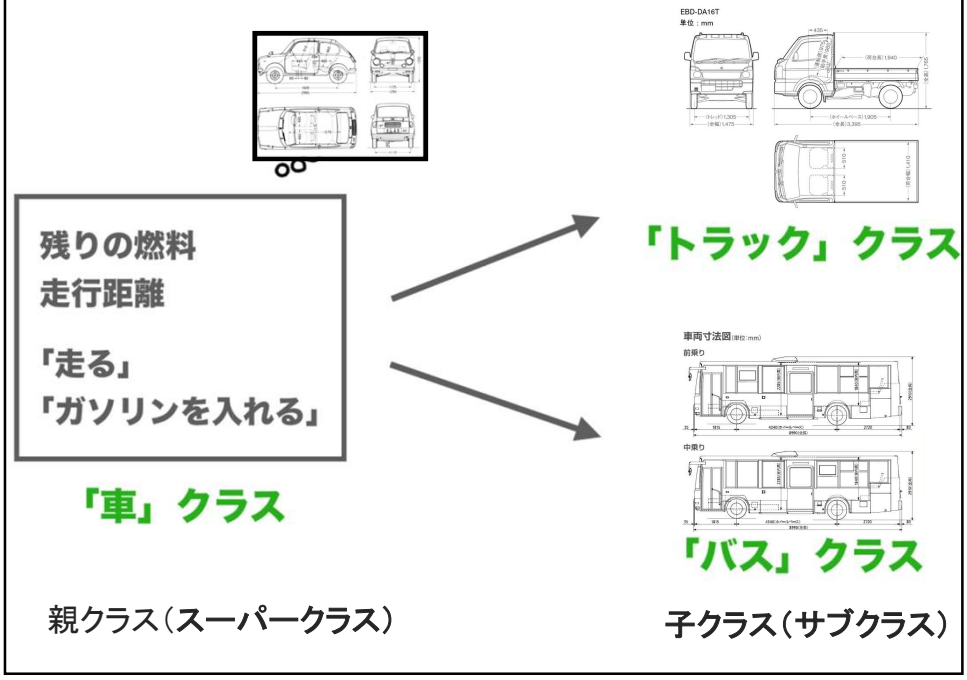
2.1 継承とは

継承：モノの種類（クラス）の共通点と相違点を
体系的に整理する仕組み

あるクラスAからあるクラスBへ、
クラスで定義したデータやメソッドを受け継がせることができる。

継承元になるクラスを親クラス、継承先になるクラスを子クラス
「この2つのクラスには親子関係がある」ともいう。

（スーパークラス、サブクラスともいう。）



2.2 継承の書式

親クラスの書式

```
class 親クラス名:
      変数の定義
      関数の定義
```

子クラスの書式

```
class 子クラス名(親クラス名):
      変数の定義
      関数の定義
```

```
class Car():
    body_color="blue"
    def drive_forward(self):
        print("前進します")
    def drive_reverse(self):
        print("バックします")
    def horn(self):
        print("プップー")
```

親クラス

```
class Bus(Car):
    def __init__(self):
        print("バスです")
```

子クラス

```
tmubus= Bus()
```

```
バスです
```

```
tmubus.body_color
```

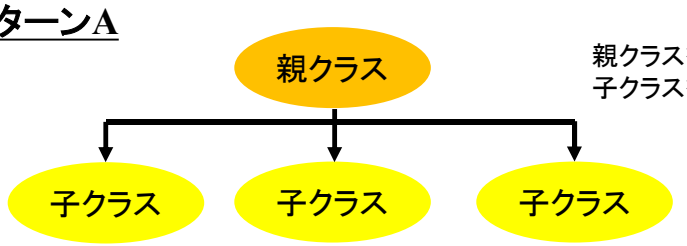
```
'blue'
```

```
tmubus.horn()
```

```
プップー
```

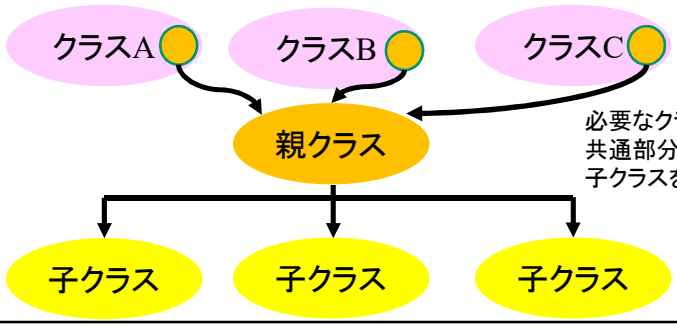
親クラスを継承して子クラスを作るパターン

パターンA



親クラスを作って
子クラスを作っていく

パターンB



必要なクラスを作って
共通部分で親クラスを作って
子クラスを作っていく

パターンA

親クラスを
作ってから
継承で子ク
ラス作成

```
class Car():
    def drive_forward(self):
        print("前進します")
    def drive_reverse(self):
        print("バックします")
    def horn(self):
        print("プップー")
```

```
class Bus(Car):
    def __init__(self):
        print("バスです")
```

```
class Truck(Car):
    def __init__(self):
        print("トラックです")
```

パターンB

2つのクラス

```
class Bus():
    def __init__(self):
        print("バスです")
    def drive_forward(self):
        print("前進します")
    def drive_reverse(self):
        print("バックします")
    def horn(self):
        print("プップー")
```

```
class Truck():
    def __init__(self):
        print("トラックです")
    def drive_forward(self):
        print("前進します")
    def drive_reverse(self):
        print("バックします")
    def horn(self):
        print("プップー")
```

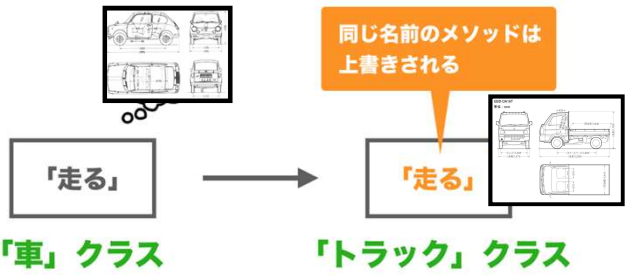
共通部分で親クラスを作ってから
継承で子クラス作成

```
class Car():
    def drive_forward(self):
        print("前進します")
    def drive_reverse(self):
        print("バックします")
    def horn(self):
        print("プップー")
```

```
class Bus(Car):
    def __init__(self):
        print("バスです")
```

```
class Truck(Car):
    def __init__(self):
        print("トラックです")
```

2.3 オーバーライド



「上書き」という意味で、親クラスで定義したメソッドを
子クラスで同名のメソッドを上書きする。

```
class Car():
    body_color="blue"
    def drive_forward(self):
        print("前進します")
    def drive_reverse(self):
        print("バックします")
    def horn(self):
        print("プップー")
```

```
class Truck(Car):
    def __init__(self):
        print("トラックです")
    def horn(self):
        print("ブオーン")
```

← オーバーライド

```
tmutruck=Truck()
```

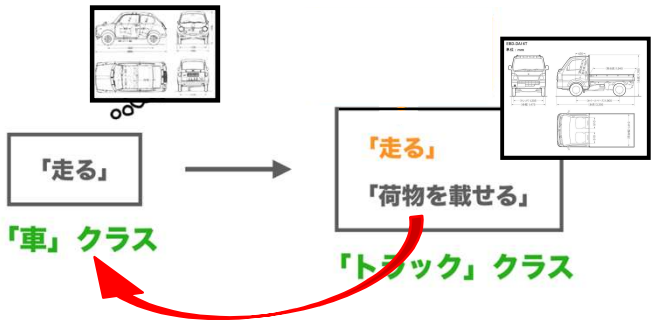
```
トラックです
```

```
tmutruck.horn()
```

```
ブオーン
```

← オーバーライドの結果

2.5　子クラスの中で親クラスを呼び出す



親クラスの呼び出し

```
super(子クラス，インスタンス)
```

```
class Car():
    body_color="blue"
    def drive_forward(self):
        print("前進します")
    def drive_reverse(self):
        print("バックします")
    def horn(self):
        print("プップー")
```

```
class Bus(Car):
    def __init__(self):
        print("バスです")
    def drive_horn(self):
        parent_class = super(Bus, self)
        parent_class.drive_forward()
        print("ブーブーブー")
```

```
tmubus=Bus()
```

```
バスです
```

```
tmubus.drive_horn()
```

```
前進します
ブーブーブー
```

オーバーライドするとメソッド名がかぶって、
子クラスのメソッドが優先される。
（子クラスのメソッドだけ呼び出されます。）

子クラスの中で親クラスのメソッドを呼び出す

`super().メソッド名()`

```
class Teacher:
 def __init__(self):
   print('Teacher')

class Student(Teacher):
 def __init__(self):
   super().__init__()
   print('Student’)

s=Student()
```

練習問題1（宿題として提出）

Animalクラスを親クラスとして作成します。
クラス変数legs＝4を定義して、
walkメソッドとして、“歩く”を表示、
cryメソッドとして、“鳴く”を表示、
get_legs_numメソッドとして、legsを表示する
メソッドを定義せよ。

ファイル名のつけ方

39492959_0608. ipynb
学修番号_日付. ipynb

練習問題2（宿題として提出）

「継承」をつかって、Animalクラスを親クラスとする、
Dogクラスを子クラスとして作成します。
子クラスでは、“犬です”を表示するコントラクタを定義して
cryメソッドとして、“わんわん”を表示させてください（オーバーライド）。

Dogクラスをインスタンス化して、pochiというインスタンスを作成し、親クラスのメソッドを全て実行してください。

演習問題（宿題として提出）：

練習問題1の親クラスで、コントラクタを使って、
legsを引数とする形に変更してください。

子クラスとしてSnakeクラスを作成します。
子クラスでは、“へびです”を表示するコントラクタを定義して、
walkメソッドとして“はう”を表示し、
cryメソッドとして、“シャー”を表示させてください。

さらに、子クラスのコントラクタでは、引数をnumとし
親クラスのコントラクタを呼び出し、
そのときの引数もnumとする（親クラスの呼び出し）。
（親クラスのコントラクタ内でself.legsに代入される。）

Snakeクラスをインスタンス化して、nyoroインスタンスを作成し、
親クラスのメソッドを全て実行してください。

宿題

練習問題～演習問題を1つのファイルに。
Kibacoから. ipynbファイルを提出。
テキストボックスに感想。

締め切り：6月14日

ファイル名のつけ方

39492959_0608. ipynb

学修番号_日付. ipynb