po-8. 継承に関する演習

（Python プログラミング体験学習，全９回）

https://www.kkaneko.jp/cc/po/index.html

金子邦彦

アウトライン

8-1 復習
8-2 継承に関する実習

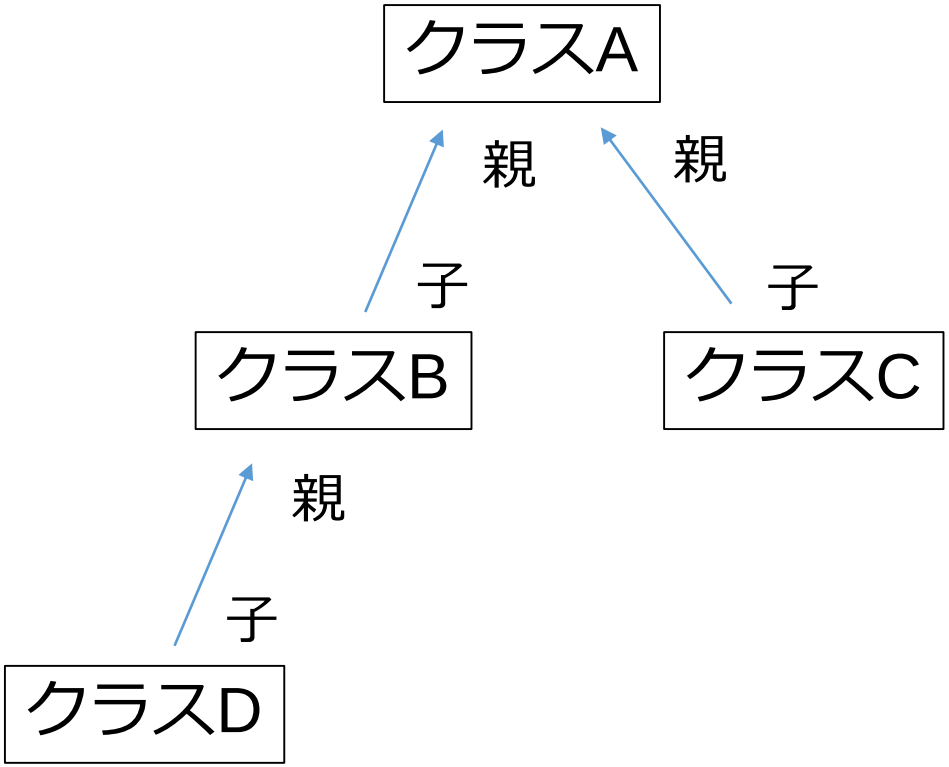
- **クラス階層**と**継承**の有用性を理解し，Python で扱えるようになる.

8-1．復習

クラス階層とは

まとめページ

クラス階層とは，**複数のクラスが親子関係をなす**こと

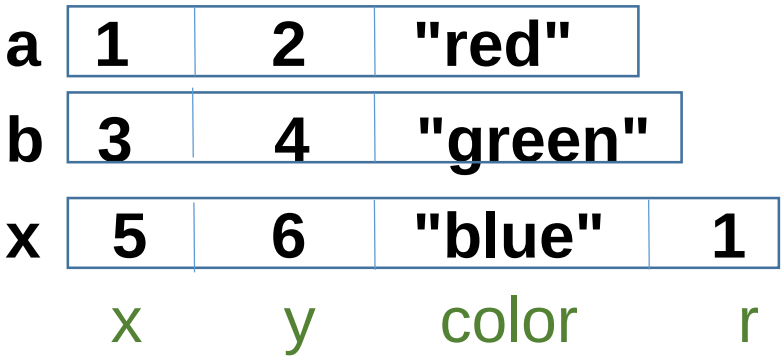


継承

- **継承**とは，**親クラス**の**属性**と**メソッド**を**子クラス**が**受け継ぐ**こと
- **親クラス**のことを「**スーパークラス**」，**子クラス**のことを「**サブクラス**」ともいう

Python のオブジェクトの生成

- 次の３つの**オブジェクト**を生成



- オブジェクト生成を行うプログラム

```
Ball a = new Ball(1, 2, "red");
Ball b = new Ball(2, 4, "green");
Circle x = new Circle(5, 6, "blue", 1);
```

クラスの類似性

- 類似した２つの**クラス**

Ball

属性

x

y

color

メソッド

move

reset

x, y, color は**同じ**

r の有り無しが違う

メソッドの**名前も中身も全く同じ**とする

Circle

属性

x

y

color

r

move

reset

２つのクラスのプログラム（親子関係にしない場合）

Ball

```
class Ball:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

追加

全く同じ

Circle

```
class Circle:
    def __init__(self, x, y, color, r):
        self.x = x
        self.y = y
        self.color = color
        self.r = r
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

問いかけ

同じようなプログラムを繰り返し書きたいですか？
-> No. クラス階層により解決

Ball

```
class Ball:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

追加

全く同じ

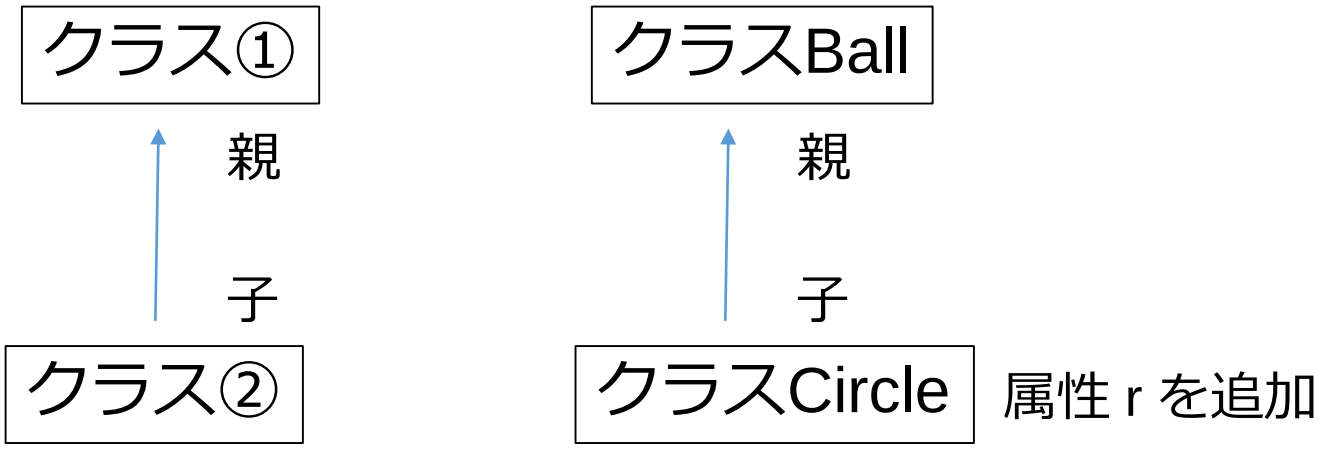
Circle

```
class Circle:
    def __init__(self, x, y, color, r):
        self.x = x
        self.y = y
        self.color = color
        self.r = r
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

クラスの親子関係

まとめページ

- クラス①が**親**，クラス②が**子**であるとき
  - クラス②は，クラス①の**属性**と**メソッド**を**すべて持つ**
  - クラス②で，クラス①にない**属性**や**メソッド**が**追加されることがある**



クラスの親子関係

```
class Ball:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

クラス名 **Ball**

属性 **x, y, color**

メソッド **move, reset**

```
class Circle(Ball):
    def __init__(self, x, y, color, r):
        super(Circle, self).__init__(x, y, color)
        self.r = r
```

クラス名 **Circle**

属性 **x**, **y**, **color, r**

メソッド **move, reset**

クラス **Circle** は，**親クラス**であるクラス **Ball** の**属性**と**メソッド**を**継承**する.

２つのクラスのプログラム
親子関係にしない場合とする場合の比較

Ball

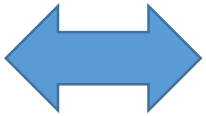
```
class Ball:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

Circle

```
class Circle:
    def __init__(self, x, y, color, r):
        self.x = x
        self.y = y
        self.color = color
        self.r = r
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

親子関係にしない
（同じようなプログラムを繰り返す）

働きは同じ



Ball

```
class Ball:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def move(self, xx, yy):
        self.x = self.x + xx
        self.y = self.y + yy
    def reset(self):
        self.x = 0
        self.y = 0
```

Circle

```
class Circle(Ball):
    def __init__(self, x, y, color, r):
        super(Circle, self).__init__(x, y, color)
        self.r = r
```

親子関係にする

Python でのクラスの親子関係の書き方

親子関係の指定「**class Circle(Ball):**」

子クラスである Circle で**追加される属性**，**メソッド**を書く

```
class Circle(Ball):
    def __init__(self, x, y, color, r):
        super(Circle, self).__init__(x, y, color)
        self.r = r
```

コンストラクタの定義.
super(Circle, self).__init__ により，**親クラス**の**コンストラクタ**を呼び出していることに注意

Python でのクラスの親子関係の書き方

```
class Circle(Ball):
    def __init__(self, x, y, color, r):
        super(Circle, self).__init__(x, y, color)
        self.r = r
```

親クラスの**コンストラクタ**を
呼び出して，**属性値**を設定

子クラスで追加した**属性**を設定

まとめ

- **クラス階層**とは，**複数のクラスが親子関係をなす**こと
- クラス①が**親**，クラス②が**子**であるとき
  - クラス②は，クラス①の**属性**と**メソッド**を**すべて持つ**
  - クラス②で，クラス①にない**属性**や**メソッド**が**追加されることがある**
- **親子関係**の指定は，「**class Circle(Ball):** 」のように書く．Circle が子，Ball が親.
- **継承**とは，**親クラス**の**属性**と**メソッド**を**子クラス**が**受け継ぐ**こと
- **親クラス**のことを「**スーパークラス**」，**子クラス**のことを「**サブクラス**」ともいう

実習の指示

- 資料：**１７～２３**

- 次のことを理解しマスターする
  - 親子関係にある２つのクラス定義
  - クラス定義での親クラスの指定
  - 親クラスのコンストラクタの呼び出し
  - 継承

実習

① **ウェブブラウザ**を起動する

② **Python Tutor** を使いたいので，次の URL を開く

http://www.pythontutor.com/

※ **Internet Explorer でうまく動かない**場合がある

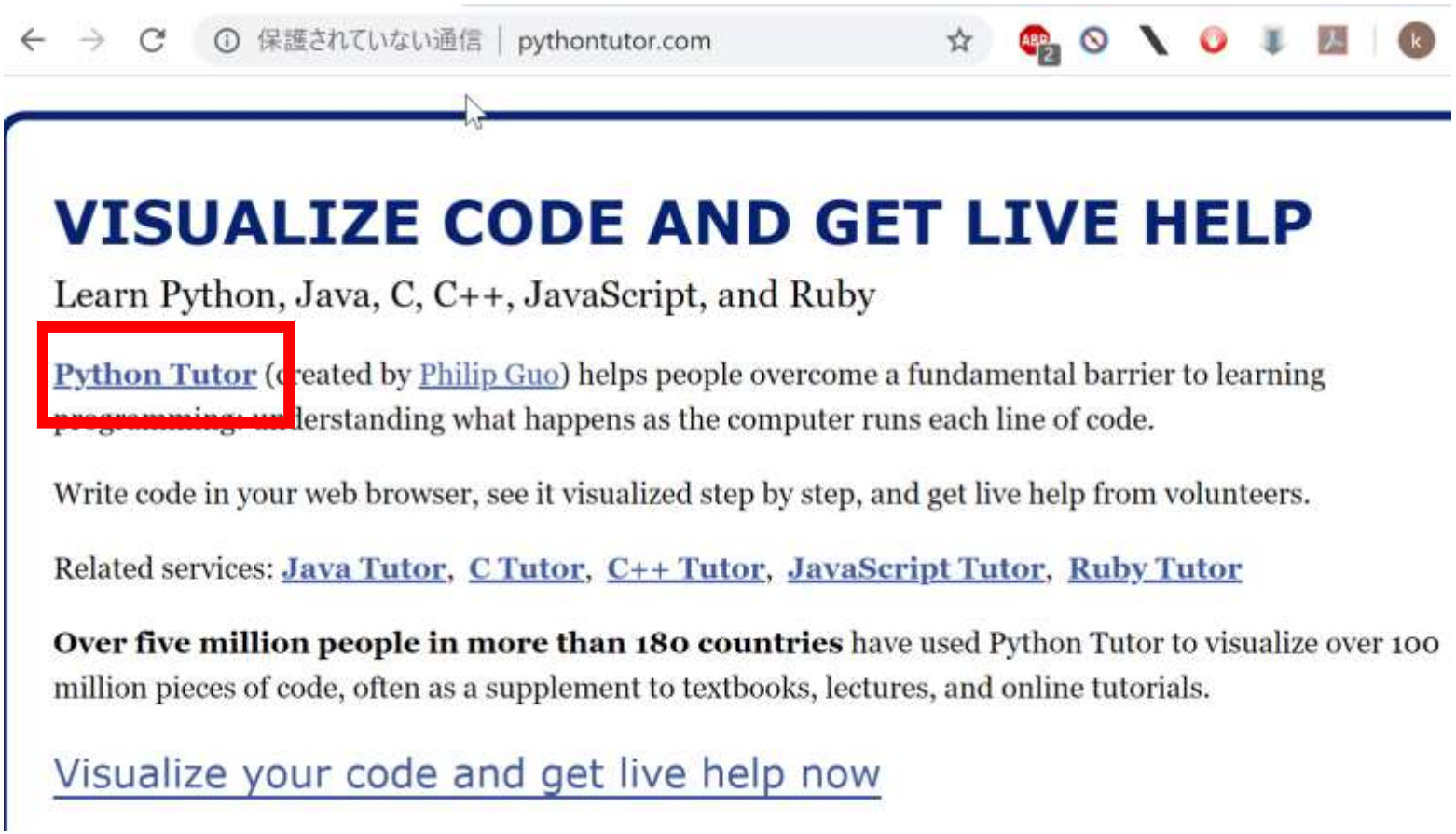
→ うまく動かないときは **Google Chrome** を試してください

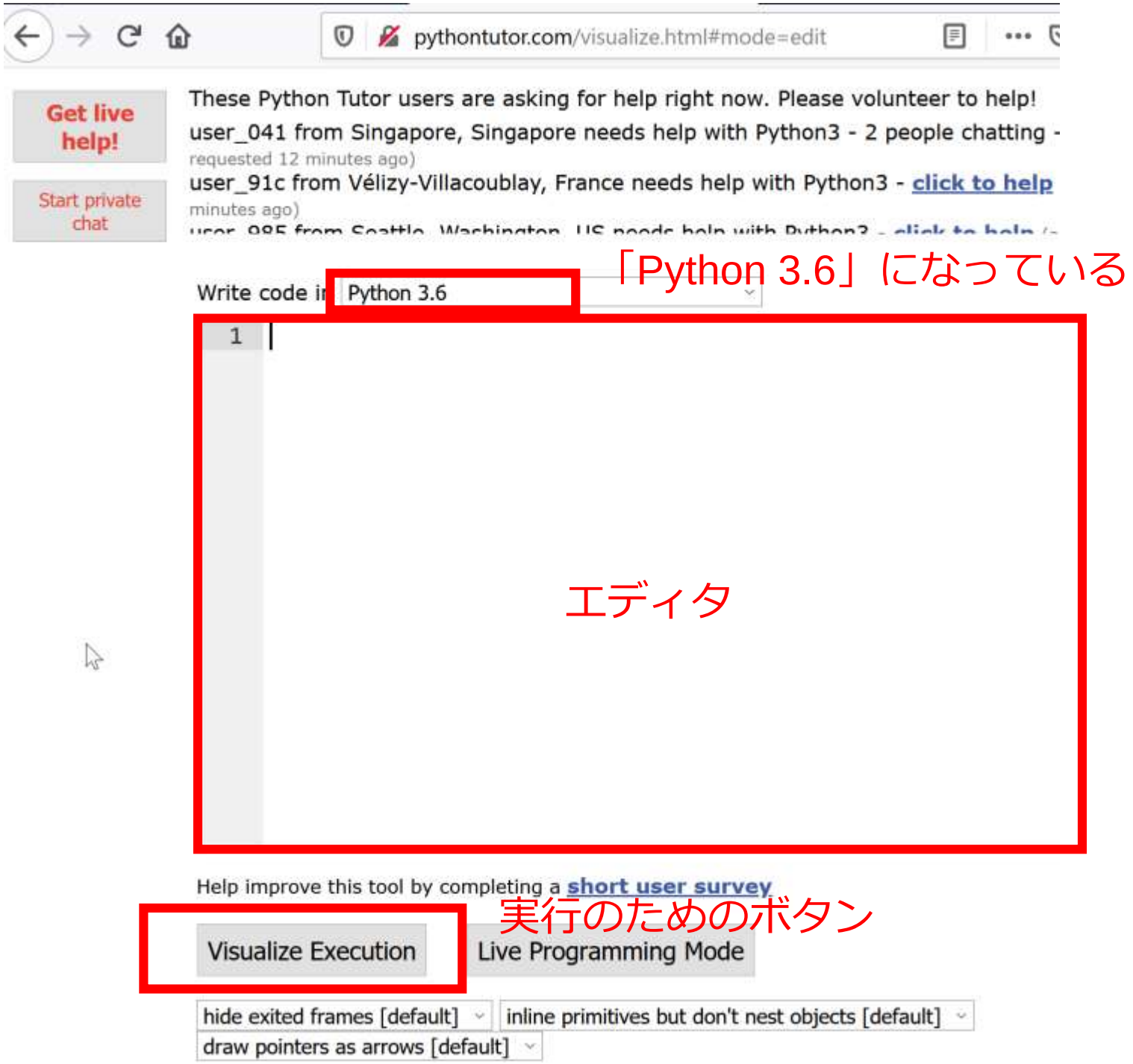
※ 途中で**「Server Busy・・・」というメッセージが出る**ことがある.

→ 混雑している．**少し（数秒から数十秒）待つ**と自動で表示が変わる（変わらない場合には，操作をもう一度行ってみる）

※ 日本語モードはない．英語で使う

③ 「Python Tutor」をクリック

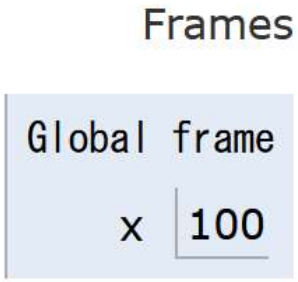



pythontutor.com/visualize.html#mode=edit
Get live help!
These Python Tutor users are asking for help right now. Please volunteer to help!
user_041 from Singapore, Singapore needs help with Python3 - 2 people chatting -
requested 12 minutes ago)
Start private chat
user_91c from Vélizy-Villacoublay, France needs help with Python3 - click to help
minutes ago)
「Python 3.6」になっている
Write code in
Python 3.6
1
エディタ
Help improve this tool by completing a short user survey
実行のためのボタン
Visualize Execution
Live Programming Mode
hide exited frames [default]
inline primitives but don't nest objects [default]
draw pointers as arrows [default]

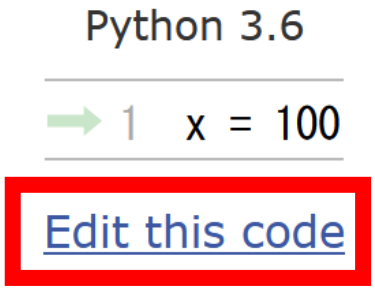
Python Tutor でのプログラム実行手順

①「**Visualize Execution**」をクリック.

②「**Last**」をクリック.



③ 結果を確認する.



④「**Edit this code**」をクリックして戻る

④ 次のプログラムを実行し，結果を確認しなさい

プログラムは，次ページに続いている

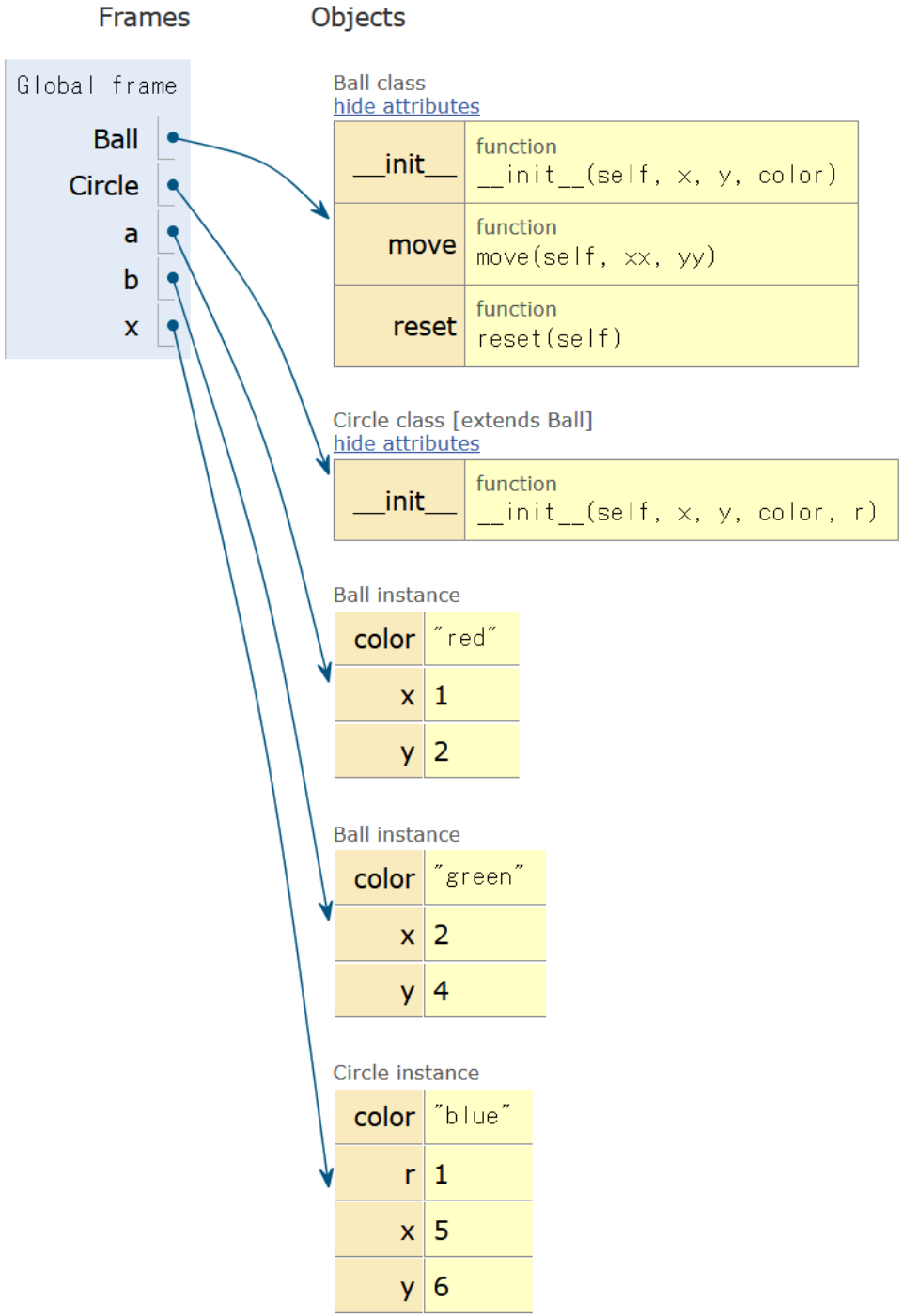
Ball クラス定義の部分

```
 1  class Ball:
 2      def __init__(self, x, y, color):
 3          self.x = x
 4          self.y = y
 5          self.color = color
 6      def move(self, xx, yy):
 7          self.x = self.x + xx
 8          self.y = self.y + yy
 9      def reset(self):
10          self.x = 0
11          self.y = 0
```

Circle クラス定義と，
コンストラクタによるオブジェクト生成

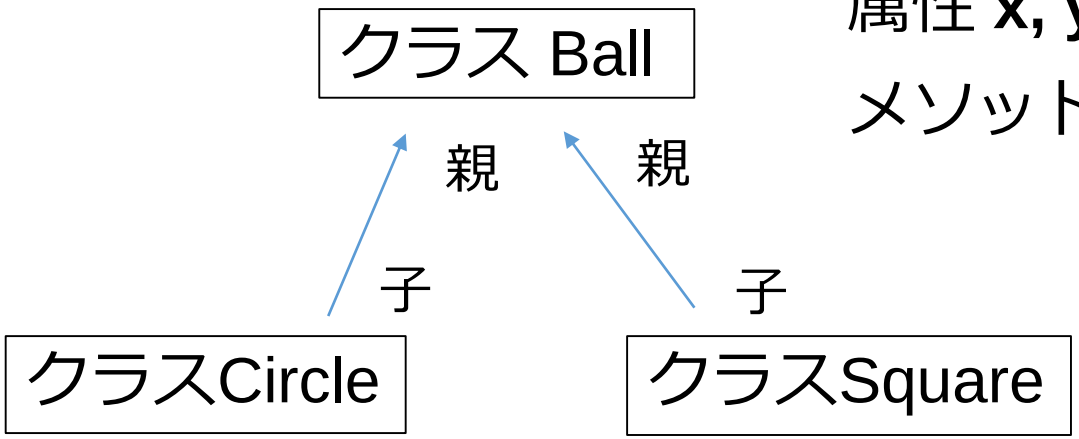
```
13  class Circle(Ball):
14      def __init__(self, x, y, color, r):
15          super(Circle, self).__init__(x, y, color)
16          self.r = r
17
18  a = Ball(1, 2, "red")
19  b = Ball(2, 4, "green")
20  x = Circle(5, 6, "blue", 1)
```

結果は次ページに記載


Frames
Objects
Global frame
Ball
Circle
a
b
x
Ball class
hide attributes
__init__
function
__init__(self, x, y, color)
move
function
move(self, xx, yy)
reset
function
reset(self)
Circle class [extends Ball]
hide attributes
__init__
function
__init__(self, x, y, color, r)
Ball instance
color
"red"
x
1
y
2
Ball instance
color
"green"
x
2
y
4
Circle instance
color
"blue"
r
1
x
5
y
6

8-2. 継承に関する実習

ここで行うこと



クラス名 **Ball**

属性 **x, y, color**

メソッド **move, reset**

クラス名 **Circle**

属性 **x, y, color, r**

メソッド **move, reset, area**

円

クラス名 **Square**

属性 **x, y, color, s**

メソッド **move, reset, area**

正方形

実習の指示

- 資料：**２７～３０**
- 次のことを理解しマスターする
  - Ball, Circle, Square のクラス定義により，継承をより深く理解する

① 次のようにプログラムを**書き換え**なさい

プログラムは，次ページに続いている

Circle クラス定義の部分

```
13  class Circle(Ball):
14      def __init__(self, x, y, color, r):
15          super(Circle, self).__init__(x, y, color)
16          self.r = r
17      def area(self):
18          return self.r * self.r * 3.14
```

② Circle クラス部分を，次のように**書き換え**なさい

```
13  class Circle(Ball):
14      def __init__(self, x, y, color, r):
15          super(Circle, self).__init__(x, y, color)
16          self.r = r
17      def area(self):
18          return self.r * self.r * 3.14
```

Square クラス定義，
コンストラクタによるオブジェクト生成，メソッド呼び出し

```
20  class Square(Ball):
21      def __init__(self, x, y, color, s):
22          super(Square, self).__init__(x, y, color)
23          self.s = s
24      def area(self):
25          return self.s * self.s
26
27  a = Ball(1, 2, "red")
28  b = Ball(2, 4, "green")
29  x = Circle(5, 6, "blue", 1)
30  y = Square(7, 8, "black", 2)
31  print(x.area())
32  print(y.area())
```

結果は次ページに記載

Print output (drag lower right

```
3.14
4
```