

po-9. クラス階層, 継承

トピックス: クラス階層, 継承 (Python Tutor による演習)

URL: <https://www.kkaneko.jp/pro/po/index.html>

(Python プログラミングの基本)

金子邦彦



全体まとめ

- **クラス階層**では、**複数のクラスが親子関係**をなす
- **子クラスの定義**では、**親クラスの指定**、**親クラスの__init__へのアクセス**を行う

親クラスの指定

```
class Ball(Point):  
    def __init__(self, x, y, r, color):  
        super(Ball, self).init(x, y, color)  
        self.r = r  
    def printout(self):  
        print(self.x, self.y, self.r, self.color)
```

親クラスの `__init__` へのアクセス

- **親クラスの階層はヒエラルキー**で、**クラスに継承**される
- **子クラス**において、**同じ名前のメソッドが別定義**されることもある

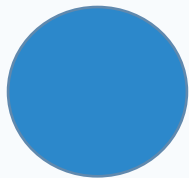
アウトライン



	項目
	復習
9-1	クラスとオブジェクト
9-2	クラス定義, オブジェクト生成
9-3	演習

クラス Ball

オブジェクト



半径 1, 場所 (8, 10)
色 blue

オブジェクト



半径 3, 場所 (2, 4)
色 green

- **2つのオブジェクトともに、同じクラス Ball** と考えることができる
- **オブジェクト**は**属性**を持つ。**半径, 場所, 色などの属性**を
考えることができる。
- **メソッド**は、**オブジェクト**に
属する操作や処理。**確認や属性の変化のための
メソッド**を考えることができる。

クラス定義の例



```
1 class Ball:
2     def __init__(self, x, y, r, color):
3         self.x = x
4         self.y = y
5         self.r = r
6         self.color = color
7     def printout(self):
8         print(self.x, self.y, self.r, self.color)
```

クラス名: Ball

メソッド: __init__, printout

属性: x, y, r, color

※ __init__ は, オブジェクト生成のためのメソッド

Python Tutor の起動



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/>

③ 「**Python**」 をクリック ⇒ **編集画面**が開く

Learn Python, JavaScript, C, C++, and Java

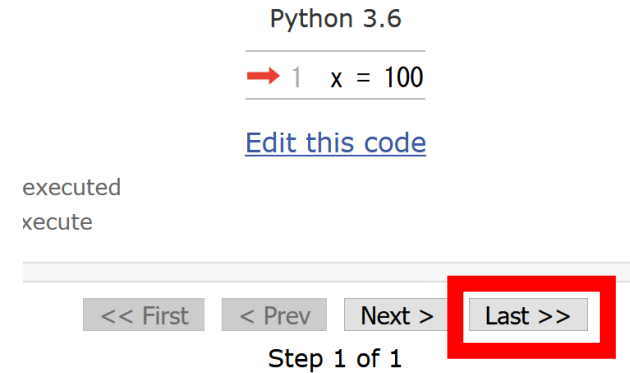
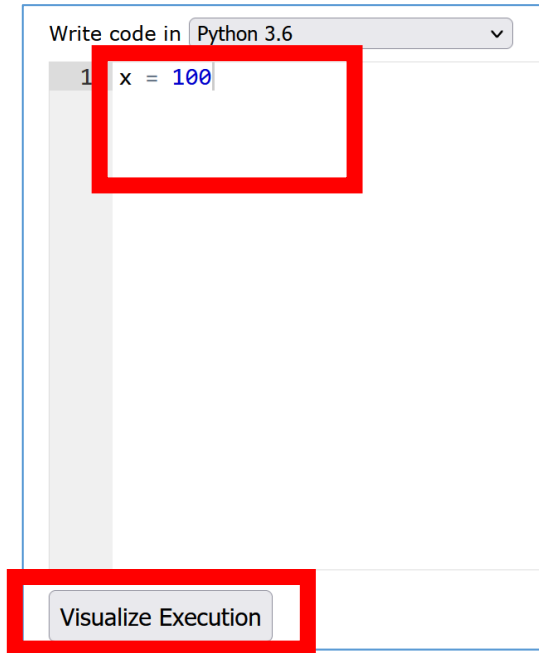
This tool helps you learn Python, JavaScript, C, C++, and Java programming by [visualizing code execution](#). You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

Over 15 million people in more than 180 countries have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

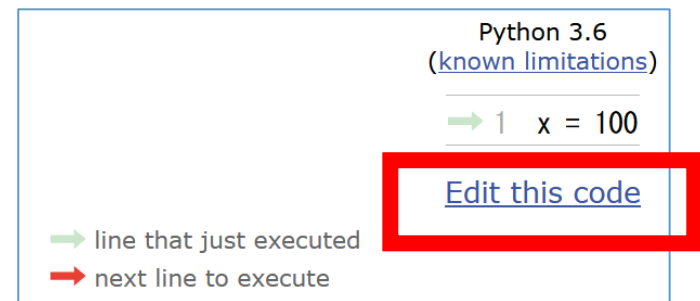
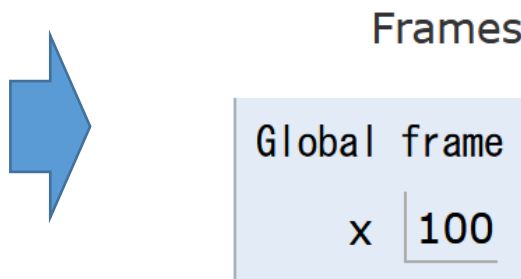
You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

Python Tutor でのプログラム実行手順



(1) 「**Visualize Execution**」をクリックして**実行画面**に切り替える

(2) 「**Last**」をクリック。



(3) 実行結果を確認する。

(4) 「**Edit this code**」をクリックして**編集画面**に戻る

Python Tutor 使用上の注意点①



- 実行画面で、次のような赤の表示が出ることがある →
無視してよい

過去の文法ミスに関する確認表示
邪魔なときは「Close」

Python Tutor: Visualize code in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

Python 3.6
([known limitations](#))

```
→ 1 x = 100
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 1 of 1

[Customize visualization](#)

Frames Objects

You just fixed the following error:

```
1 x = 100!
```

SyntaxError: invalid syntax (<string>, line 1)

Please help us improve this tool with your feedback.
What misunderstanding do you think caused this error?

Submit Close [Hide all of these pop-ups](#)

Python Tutor 使用上の注意点②



「please wait ... executing」のとき，10秒ほど待つ。



→ 混雑しているときは，「Server Busy・・・」
というメッセージが出ることがある。

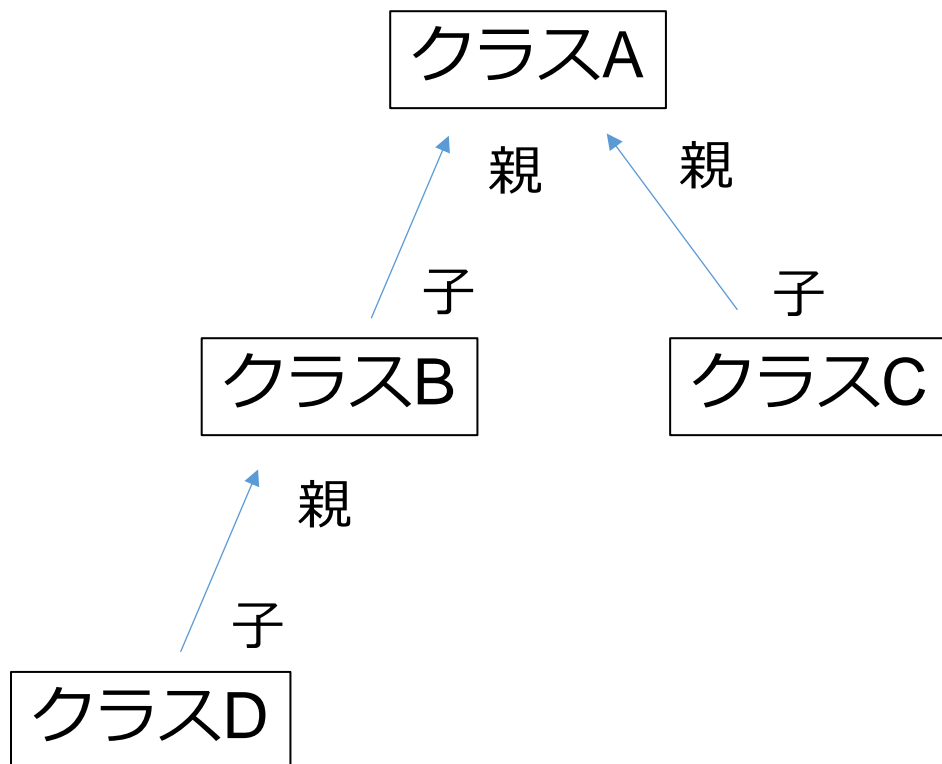
混雑している。少し（数秒から数十秒）待つと自動で表示が変わる（変わらない場合には，操作をもう一度行ってみる）

9-1. クラス階層

クラス階層



クラス階層では、複数のクラスが親子関係をなす



クラス Point

オブジェクト

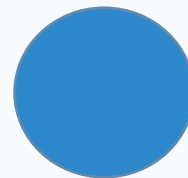


場所 (1, 2)
色 red

クラス Ball

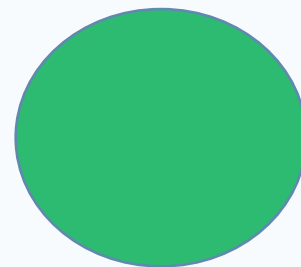


オブジェクト



半径 1, 場所 (8, 10)
色 blue

オブジェクト



半径 3, 場所 (2, 4)
色 green

Point クラス定義の例



```
1 class Point:
2     def __init__(self, x, y, color):
3         self.x = x
4         self.y = y
5         self.color = color
6     def printout(self):
7         print(self.x, self.y, self.color)
```

クラス名: Point

メソッド: __init__, printout

属性: x, y, color

※ __init__ は, オブジェクト生成のためのメソッド

演習

資料 : 15 ~ 16

【トピックス】

- クラス定義
- オブジェクト生成
- メソッドアクセス

① Python Tutor のエディタで次のプログラムを入れる



クラス定義, オブジェクト生成, メソッドアクセス

```
1 class Point:
2     def __init__(self, x, y, color):
3         self.x = x
4         self.y = y
5         self.color = color
6     def printout(self):
7         print(self.x, self.y, self.color)
8
9 p = Point(1, 2, "red")
10 p.printout()
```

クラス定義

オブジェクト生成

メソッドアクセス

字下げも正確に



② 実行し，結果を確認する

メソッド printout による表示

Python 3.6
(known limitations)

```

1 class Point:
2     def __init__(self, x, y, color):
3         self.x = x
4         self.y = y
5         self.color = color
6     def printout(self):
7         print(self.x, self.y, self.color)
8
9 p = Point(1, 2, "red")
10 p.printout()

```

[Edit this code](#)

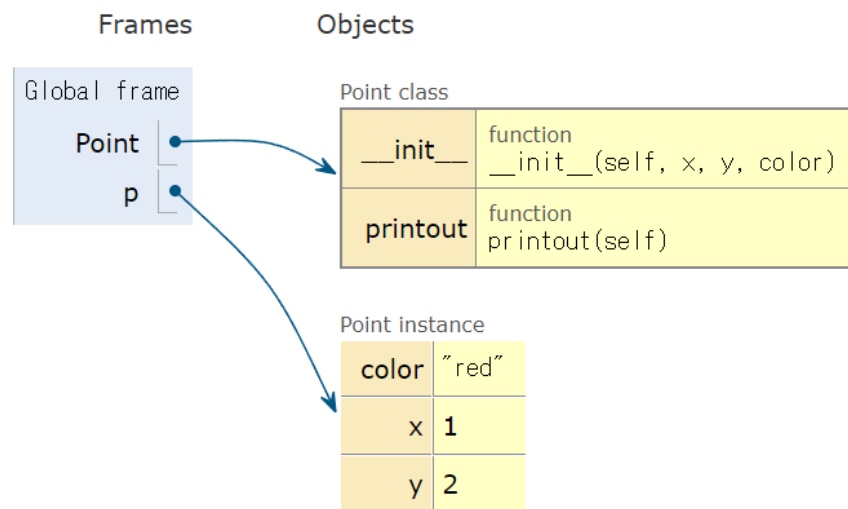
→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Done running (11 steps)

Print output (drag lower right corner to resize)

1 2 red



オブジェクト c

「Visual Execution」をクリック。そして「Last」をクリック。結果を確認。
「Edit this code」をクリックすると，エディタの画面に戻る

Ball クラス定義の例 (クラス階層を考えない場合)

```
1 class Ball:
2     def __init__(self, x, y, r, color):
3         self.x = x
4         self.y = y
5         self.r = r
6         self.color = color
7     def printout(self):
8         print(self.x, self.y, self.r, self.color)
```

クラス名: **Ball**

メソッド: **__init__, printout**

属性: **x, y, r, color**

※ **__init__** は, オブジェクト生成のためのメソッド

類似した 2つのクラス



Point

属性

x
y
color

メソッド

printout

Ball

属性

x
y
color
r

メソッド

printout



x, y, color は**同じ**

r の有り無しが
違う

**printout は名前は
同じだが、中身が違う**

Point クラスと Ball クラスの定義の例 (クラス階層を考えない場合)



Point

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def printout(self):
        print(self.x, self.y, self.color)
```

Ball

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

x, y, color は同じ

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**

Point クラスと Ball クラスの定義の例 (クラス階層を考える場合)



Point

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def printout(self):
        print(self.x, self.y, self.color)
```

Ball

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

x, y, color について
繰り返し書くことはなくなる

class Ball(Point)

Ball クラスは Point クラスの子である

Point

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def printout(self):
        print(self.x, self.y, self.color)
```

Ball

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

super(Ball, self).__init__(x, y, color)

親クラスである Point クラスの
メソッド `__init__` にアクセス。
その引数は `x, y, color`

クラス階層を考える場合と考えない場合の違い



Point

Point

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def printout(self):
        print(self.x, self.y, self.color)
```

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def printout(self):
        print(self.x, self.y, self.color)
```

働きは
同じ

Ball

Ball

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

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

クラス階層を考えない

クラス階層を考える

演習

資料 : 24 ~ 25

【トピックス】

- ・ サブクラスのクラス定義

① Python Tutor のエディタで次のプログラムを入れる



Ball クラスのクラス定義, オブジェクト生成,
メソッドアクセスを追加

```
1 class Point:
2     def __init__(self, x, y, color):
3         self.x = x
4         self.y = y
5         self.color = color
6     def printout(self):
7         print(self.x, self.y, self.color)
8
9 class Ball(Point):
10    def __init__(self, x, y, r, color):
11        super(Ball, self).__init__(x, y, color)
12        self.r = r
13    def printout(self):
14        print(self.x, self.y, self.r, self.color)
15
16 p = Point(1, 2, "red")
17 p.printout()
18 a = Ball(8, 10, 1, "blue")
19 b = Ball(2, 4, 3, "green")
20 a.printout()
21 b.printout()
```

追加

追加

② 実行し，結果を確認する

メソッド printout
による表示

Python 3.6
[\(known limitations\)](#)

```
1 class Point:
2     def __init__(self, x, y, color):
3         self.x = x
4         self.y = y
5         self.color = color
6     def printout(self):
7         print(self.x, self.y, self.color)
8
9 class Ball(Point):
10    def __init__(self, x, y, r, color):
11        super(Ball, self).__init__(x, y, color)
12        self.r = r
13    def printout(self):
14        print(self.x, self.y, self.r, self.color)
15
16 p = Point(1, 2, "red")
17 p.printout()
18 a = Ball(8, 10, 1, "blue")
19 b = Ball(2, 4, 3, "green")
20 a.printout()
21 b.printout()
```

[Edit this code](#)

→ line that just executed
→ next line to execute

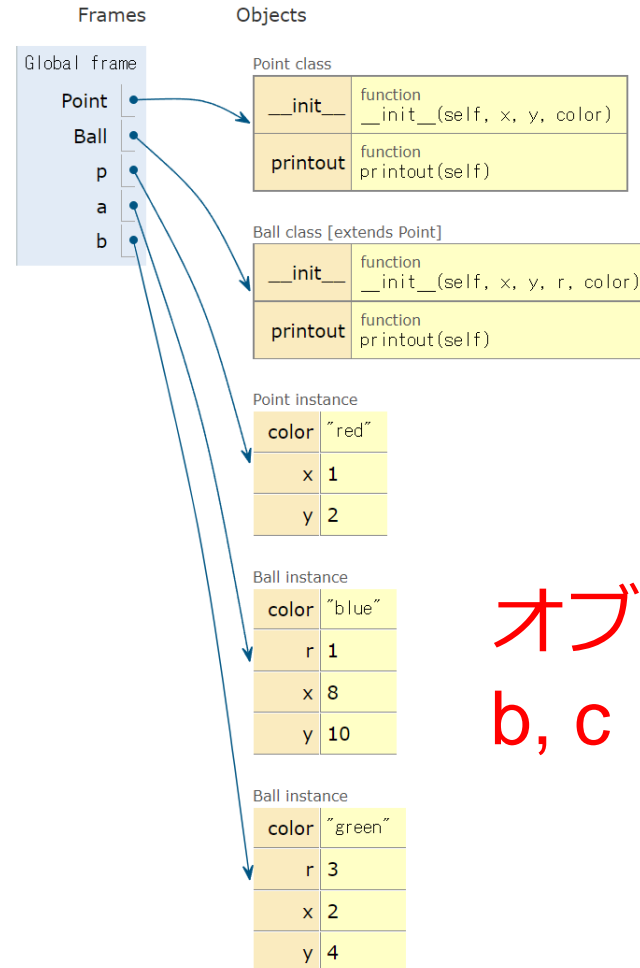
<< First < Prev Next > Last >>

Done running (40 steps)

[Customize visualization](#)

Print output (drag lower right corner to resize)

```
1 2 red
8 10 1 blue
2 4 3 green
```



オブジェクト a,
b, c

「**Visual Execution**」をクリック。そして「**Last**」をクリック。結果を確認。
「**Edit this code**」をクリックすると，エディタの画面に戻る

まとめ



- **クラス階層**では、複数のクラスが**親子関係**をなす
- **子クラスの定義**では、**親クラスの指定**、**親クラスの `__init__` へのアクセス**を行う

親クラスの指定

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

親クラスの `__init__` へのアクセス

9-2. 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**属性**と**メソッド**は, **子クラス**に**継承**される
- **子クラス**において, **同じ名前のメソッド**が**別定義**されることもある

親

Point

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def printout(self):
        print(self.x, self.y, self.color)
```

子

Ball

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

- 属性 **r** を追加
- メソッド **printout** は別定義

属性 r を追加

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

メソッド `printout` は別定義

演習

資料 : 32 ~ 36

【トピックス】

・ 継承

① Python Tutor のエディタで次のプログラムを入れる



Point クラスに, 属性 x と y を 0 にするメソッド reset を追加

```
1 class Point:
2     def __init__(self, x, y, color):
3         self.x = x
4         self.y = y
5         self.color = color
6     def printout(self):
7         print(self.x, self.y, self.color)
8     def reset(self):
9         self.x = 0
10        self.y = 0
11
12 class Ball(Point):
13     def __init__(self, x, y, r, color):
14         super(Ball, self).__init__(x, y, color)
15         self.r = r
16     def printout(self):
17         print(self.x, self.y, self.r, self.color)
18
19 p = Point(1, 2, "red")
20 p.printout()
21 a = Ball(8, 10, 1, "blue")
22 b = Ball(2, 4, 3, "green")
23 a.reset()
24 b.reset()
25 a.printout()
26 b.printout()
```

追加

追加



② 実行し，結果を確認する

Point クラスのメソッド reset が Ball クラスに継承されていることを確認

メソッド printout
による表示

```

Python 3.6
(known limitations)
0  def printout(self):
7      print(self.x, self.y, self.color)
8  def reset(self):
9      self.x = 0
10     self.y = 0
11
12 class Ball(Point):
13     def __init__(self, x, y, r, color):
14         super(Ball, self).__init__(x, y, color)
15         self.r = r
16     def printout(self):
17         print(self.x, self.y, self.r, self.color)
18
19 p = Point(1, 2, "red")
20 p.printout()
21 a = Ball(8, 10, 1, "blue")
22 b = Ball(2, 4, 3, "green")
23 a.reset()
24 b.reset()
25 a.printout()
26 b.printout()

```

Edit this code

Line that just executed:
next line to execute

<< First < Prev Next > Last >>

Done running (50 steps)

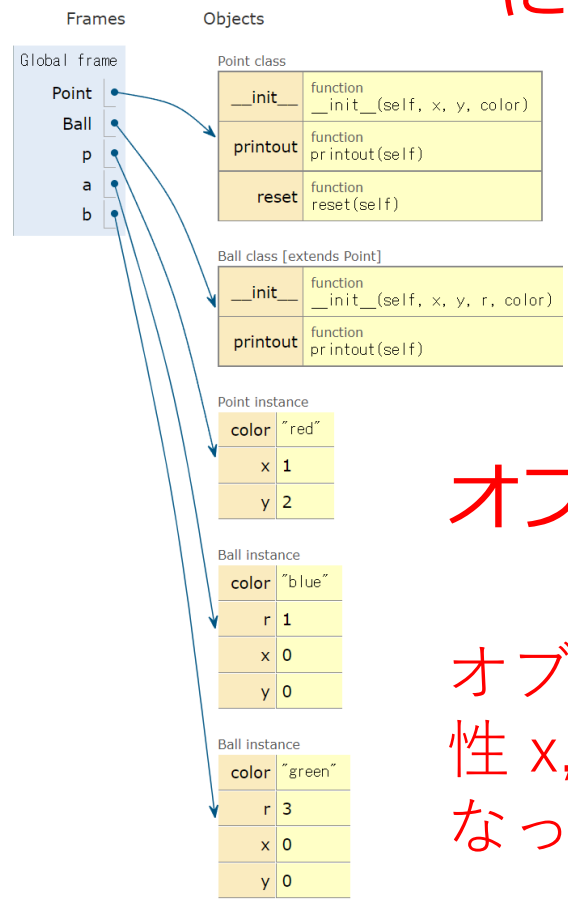
[imize visualization](#)

Print output (drag lower right corner to resize)

```

1 2 red
0 0 1 blue
0 0 3 green

```



オブジェクト a, b, c

オブジェクト a, b の属性 x, y の値が 0 になっている

「Visual Execution」をクリック。そして「Last」をクリック。結果を確認。
「Edit this code」をクリックすると，エディタの画面に戻る

Python では、次のプログラムにより、**オブジェクト `a` のメソッド名、属性名などを表示**できる

```
print(dir(a))
```

③ Python Tutor のエディタで,



プログラムの末尾に「**print(dir(a))**」を追加する

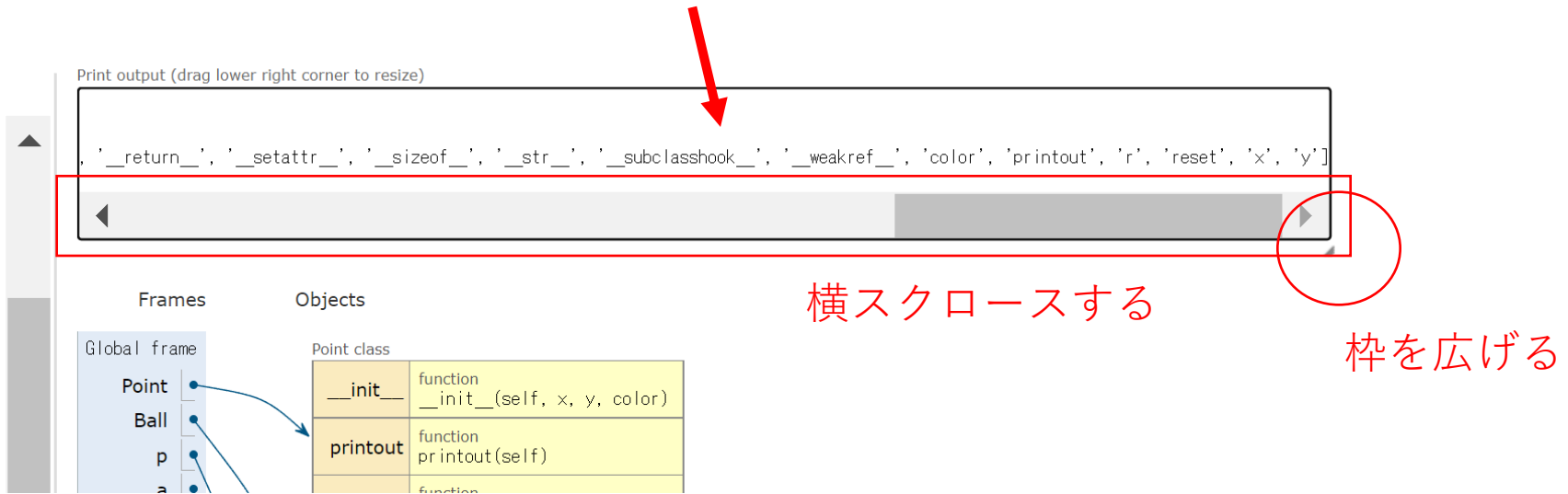
```
1 class Point:
2     def __init__(self, x, y, color):
3         self.x = x
4         self.y = y
5         self.color = color
6     def printout(self):
7         print(self.x, self.y, self.color)
8     def reset(self):
9         self.x = 0
10        self.y = 0
11
12 class Ball(Point):
13     def __init__(self, x, y, r, color):
14         super(Ball, self).__init__(x, y, color)
15         self.r = r
16     def printout(self):
17         print(self.x, self.y, self.r, self.color)
18
19 p = Point(1, 2, "red")
20 p.printout()
21 a = Ball(8, 10, 1, "blue")
22 b = Ball(2, 4, 3, "green")
23 a.reset()
24 b.reset()
25 a.printout()
26 b.printout()
27 print(dir(a))
```

追加

④ 実行し，結果を確認する

Ball クラスには，Point クラスの属性，メソッドが継承されている

__class__ など「__」で始まるもの：システムが自動で追加したメソッド，属性
color, printout, r, reset, x, y: プログラムで書いたメソッド，属性



Print output (drag lower right corner to resize)

```
['__return__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'color', 'printout', 'r', 'reset', 'x', 'y']
```

横スクロールする

枠を広げる

Frames	Objects
Global frame	Point class
Point	<code>__init__</code> function <code>__init__(self, x, y, color)</code>
Ball	<code>printout</code> function <code>printout(self)</code>
p	
a	

「**Visual Execution**」をクリック。そして「**Last**」をクリック。結果を確認。
「**Edit this code**」をクリックすると，エディタの画面に戻る

全体まとめ

- **クラス階層**では、**複数のクラスが親子関係**をなす
- **子クラスの定義**では、**親クラスの指定**、**親クラスの__init__へのアクセス**を行う

親クラスの指定

```
class Ball(Point):  
    def __init__(self, x, y, r, color):  
        super(Ball, self).init(x, y, color)  
        self.r = r  
    def printout(self):  
        print(self.x, self.y, self.r, self.color)
```

親クラスの `__init__` へのアクセス

- **親クラスの階層はヒエラルキー**で、**クラスに継承**される
- **子クラス**において、**同じ名前のメソッドが別定義**されることもある

Python 関連ページ



- Python まとめページ

<https://www.kkaneko.jp/tools/man/python.html>

- Python 入門（スライド資料とプログラム例）

<https://www.kkaneko.jp/pro/pf/index.html>

- Python プログラミングの基本（スライド資料とプログラム例）

<https://www.kkaneko.jp/pro/po/index.html>

- Python プログラム例

<https://www.kkaneko.jp/pro/python/index.html>

- 人工知能の実行（Google Colaboratory を使用）

<https://www.kkaneko.jp/ai/ni/index.html>

- 人工知能の実行（Python を使用）（Windows 上）

<https://www.kkaneko.jp/ai/deepim/index.html>