pf-11. クラス階層，継承

（Python 入門）

URL: https://www.kkaneko.jp/pro/pf/index.html

金子邦彦

オブジェクトとメソッド

- **オブジェクト**：コンピュータでの**操作や処理の対象となるもの**

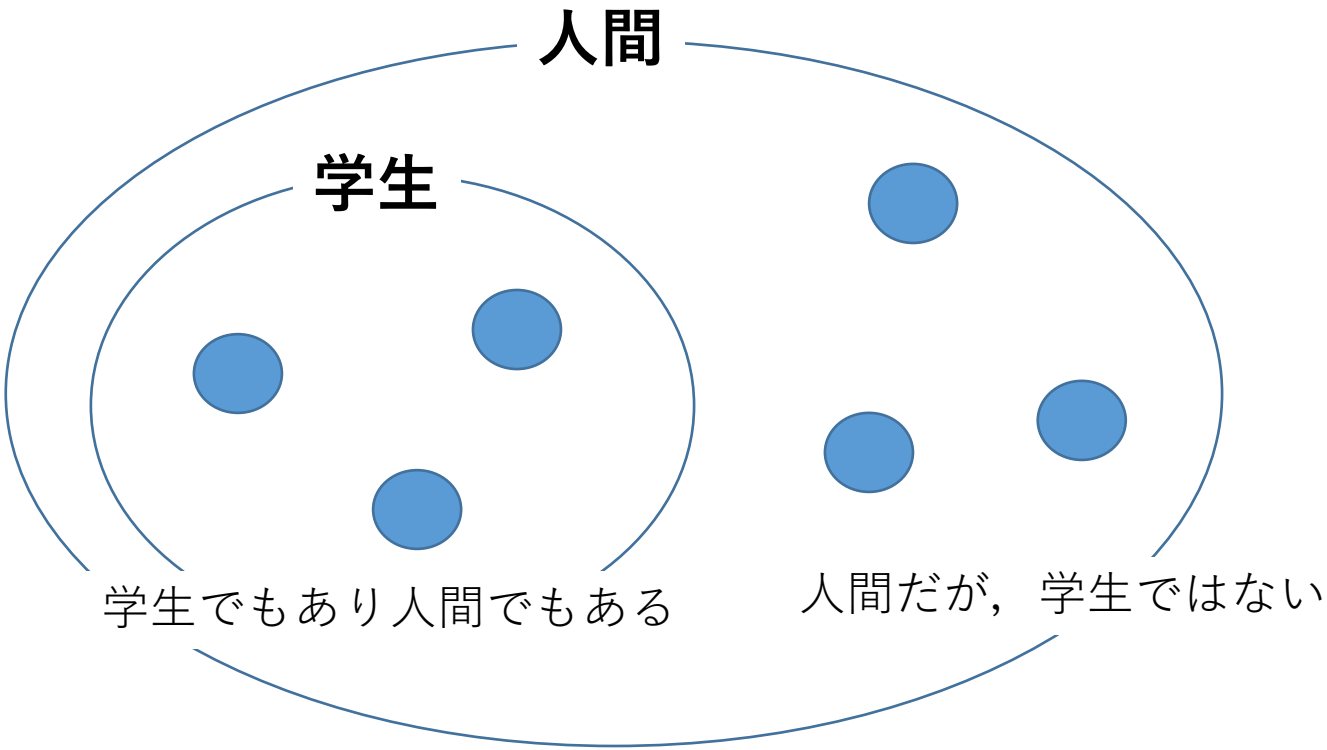
hero.moveDown()	**hero　オブジェクト** **moveDown()　メソッド** 間を「.」で**区切って**いる

- **メソッド: オブジェクト**に属する機能や操作．オブジェクトがもつ能力に相当する
- **引数：メソッド**が行う操作の詳細に関する情報，**メソッド**呼び出しのときに、引数を指定できる

hero.attack("fence", 36, 26)

クラスとオブジェクト

クラスは，**同じ種類のオブジェクトの集まり**と考えることができる



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)
8
9  p = Point(1, 2, "red")
10 p.printout()
```

クラス名: Point

属性: x, y, color

メソッド: __init__, printout

オブジェクト生成の際に，メソッド __init__ が自動で実行される

trinket

- **Trinket** は**オンライン**の **Python、HTML** 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）

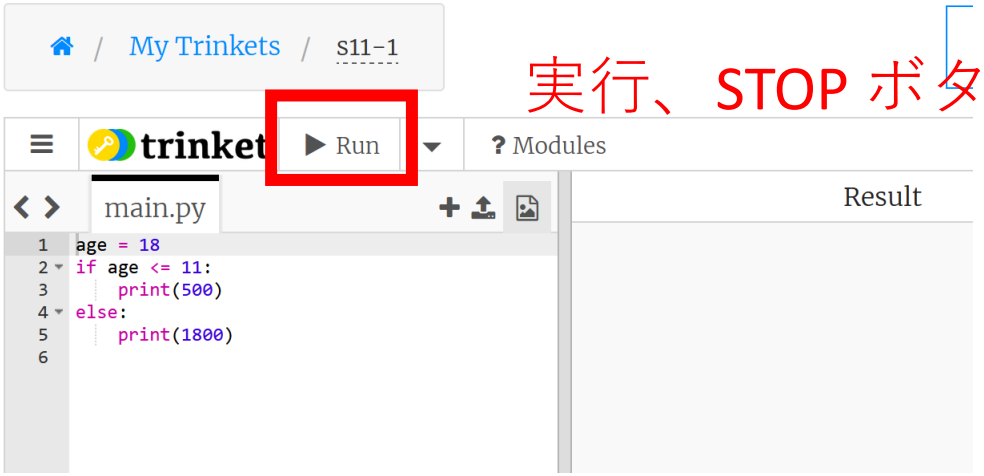
- **Python の標準機能**を登載、その他、次のパッケージがインストール済み

math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

trinket でのプログラム実行

- **trinket は Python, HTML などのプログラムを書き実行できる**サイト
- https://trinket.io/python/0fd59392c8

のように、違うプログラムには違う URL が割り当てられる



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習

資料：8

【トピックス】

- **クラス定義**
- **class**
- **オブジェクト生成**
- **メソッド呼び出し**

① trinket の次のページを開く

https://trinket.io/python/b0698edcb7

② 実行結果が，次のように表示されることを確認

このプログラムは，**オブジェクト p を生成**する．そして，

メソッド printout を呼び出して，属性値を表示させる

main.py

```
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()
```

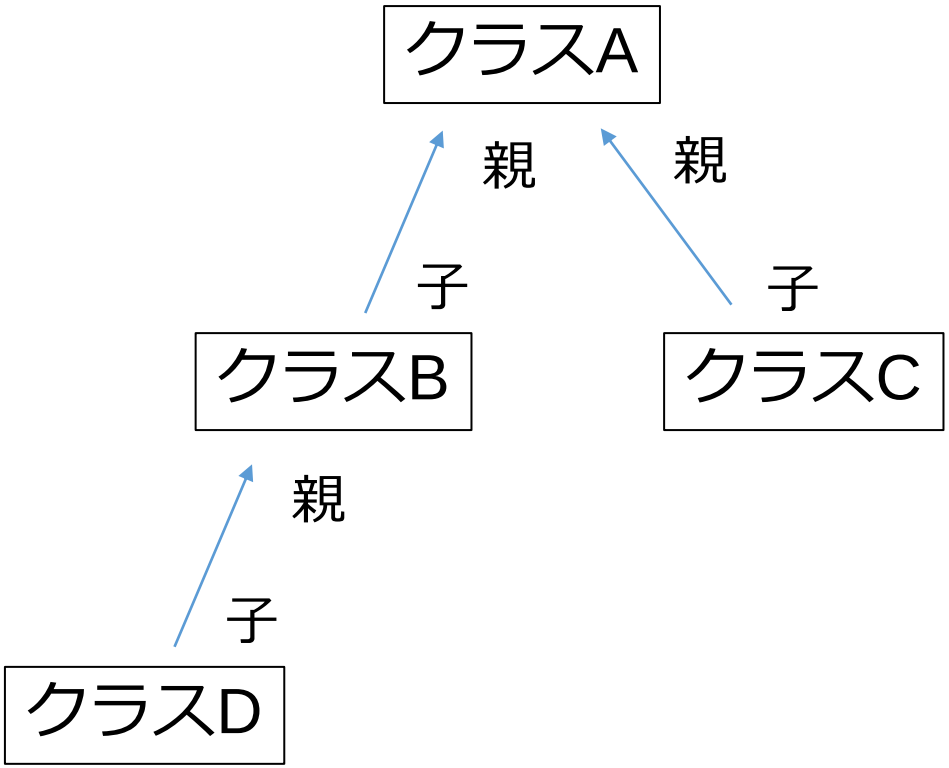
Result

Powered by trinket

```
(1, 2, 'red')
```

クラス階層

クラス階層では，**<u>複数のクラスが親子関係をなす</u>**



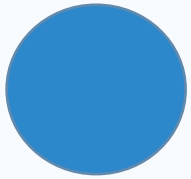
クラス Point

オブジェクト

場所（1, 2）
色 red

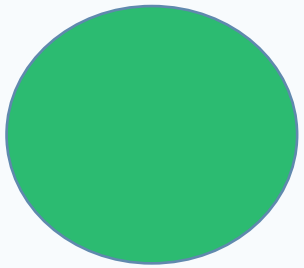
クラス Ball

オブジェクト

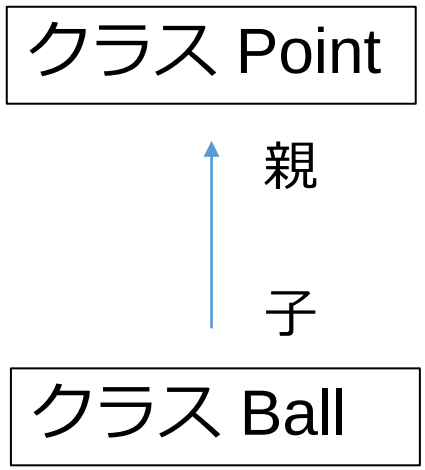


半径 1，場所（8, 10）
色 blue

オブジェクト



半径 3，場所（2, 4）
色 green



類似した 2つのクラス

Point

属性

- x
- y
- color

メソッド

- printout

x, y, color は**同じ**

r の有り無しが違う

Ball

属性

- x
- y
- color
- r

メソッド

- printout

printout は名前は同じだが，中身が違う

継承

- **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

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

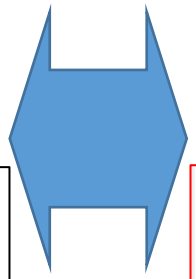
```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
    def printout(self):
        print(self.x, self.y, self.color)
```

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)
```

クラス階層を考えない

働きは同じ



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)
```

クラス階層を考える

演習

資料：19

【トピックス】

- **クラス定義**
- **Class**
- **サブクラスのクラス定義**

① trinket の次のページを開く

https://trinket.io/python/8f5efe7d0e

② 実行結果が，次のように表示されることを確認

このプログラムは，**オブジェクト p, a, b を生成**する．そして，**メソッド printout を呼び出し**て，属性値を表示させる