第8章クラスとオブジェクト指向開発

12T4069L
佐鳥 恭太郎

オブジェクト

・データと命令（メソッド）が一緒になったもの
　　メソッドでデータを操作できる

・似た性質を持つデータに対して、同じ操作を行う場合は、同じメソッドを使う

例:　　リスト [‘a’, ‘b’, ‘c’]から要素’b’を取り除く

=> [‘a’, ‘b’, ‘c’].remove(‘b’)

集合{‘a’, ‘b’, ’c’}から要素’b’を取り除く

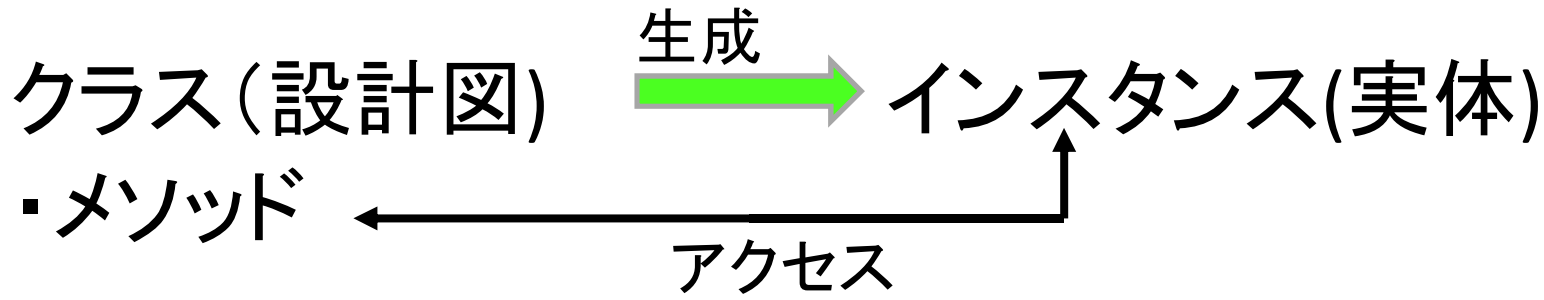
=> {‘a’, ‘b’, ’c’}.remove(‘b’)

クラスからオブジェクトを作る

オブジェクトの作り方

オブジェクト(インスタンス) = クラス名(引数,...)

・クラスから作ったオブジェクトをインスタンスと呼ぶ



クラスを作る

```
>>> class クラス名（スーパークラス名,...）:
...     pass
...
```

・「クラス名」という名前のクラスを作る
・「スーパークラス名」を指定すると、
そのクラスの機能を引き継げる（継承）

・一つインデントしてクラスの内容を記述する
*passは何もしないという意味

インスタンスのアトリビュート

```
>>> class Foo :
...     pass
...
>>> a = Foo()
>>> a.x
=>AttributeError
>>> a.x = 10
>>> a.x
10
```

・インスタンス内にxは存在しないので
エラーになる

・アトリビュート(属性)xが
生成され、10が代入された

メソッドの定義と初期化メソッド

・アンダースコア2つ

```
>>> class クラス名：
・・・    def __init__(self, 引数1, 引数2, …) :
・・・        self.アトリビュート1= 引数1
・・・        self.アトリビュート2 = 引数2
・・・    def メソッド名(self, 引数, ….) :
・・・        処理
・・・        return 戻り値
・・・
```

・クラス作成時に呼ばれる

・アトリビュートを追加

・クラス作成時とメソッド呼び出し時の引数selfは省略する

```
>>> インスタンス = クラス名（引数1, 引数2, …）
>>> インスタンス.メソッド名(引数,…)
```

アトリビュート

・アトリビュートに想定していない型が入力されることがある

```
#長方形(正方形)のクラス
>>> class Square :
・・・     def __init__(self, x1, y1) :
・・・         self.x = x1
・・・         self.y = y1
・・・     def area(self) : # 面積
・・・         return self.x * self.y
>>> a = Square(2, 3)
>>> a.area()
6
>>> b = Square('z', 3)
>>> b.area()
'zzz'
```

オブジェクトのクラスを調べる

isinstance(オブジェクト, クラス名)

「オブジェクト」が「クラス名」のインスタンスであれば真、そうでなければ偽を返す。

アトリビュートを隠す

・Pythonのクラスのアトリビュートはpublicで、誰でも変更できる

・変更できないようにするには

=>ルール

・アトリビュート名やメソッド名の先頭に
_(アンダースコア1つ)をつける

「クラスの内部だけで利用する」という約束
実際は変更できる

=>構文

・アトリビュート名やメソッド名の先頭に
__(アンダースコア2つ)をつける

変数名を内部的に他のものに置き換える

頑張ればアクセスできる

課題8

座標(x, y)系上の点(0, 0)と点(10, 10)の距離を求めよ

ただし、座標(x,y)はクラスのアトリビュート(属性)として記憶し、そのクラスに点と点の距離を求めるメソッドを実装すること