アクセッサ（ゲッタとセッタ）

スポーツクラブの運営の都合上、各会員の体重の値を頻繁に取得・設定する必要が生じたことにあわせて、体重の値を取得・設定する機能をクラスに追加することにします。

そこで思いつくのが、以下のように二つのメソッドを定義することです。

```
def get_weight(self) -> float
    """体重を取得"""
    return self.__weight
```

```
def set_weight(self, weight: float) -> None
    """体重を設定"""
    self.__weight = weight
```

これらのメソッドによって、体重の取得・設定が次のように行えます。

```
w = yamada.get_weight()      # 山田君の体重を取得
sekine.set_weight(72.5)      # 関根君の体重を72.5に設定
```

このような、データ属性の値を取得・設定するメソッドは、**ゲッタ**（*getter*）と**セッタ**（*setter*）と呼ばれます。なお、二つの総称は、**アクセッサ**（*accessor*）です。

ゲッタとセッタを定義する際の定石は、**@property デコレータ**を使う手法です。**List 11-4** に示すのが、そのプログラムです。

List 11-4 chap11/Member04.py

```
# スポーツクラブの会員クラス（第４版）

class Member:
    """スポーツクラブの会員クラス（第４版）"""

    # --- 中略：__init__, lose_weight, printは、第３版と同じ --- #

    @property
    def weight(self) -> float:
        """体重を取得（ゲッタ）"""
        return self.__weight

    @weight.setter
    def weight(self, weight: float) -> None:
        """体重を設定（セッタ）"""
        self.__weight = weight if weight > 0.0 else 0.0

# 会員クラスのテスト
yamada = Member(15, '山田太郎', 72.7)

yamada.weight = 67.3                    # 山田君の体重を設定
print('yamada.weight =', yamada.weight)     # 山田君の体重を取得
```

実行結果

```
yamada.weight = 67.3
```

▪ ゲッタの定義

前置きとして“@property”を付けて定義します。

メソッド名は、値を調べるデータ属性を表すのに適切な名前とします。この名前を《アクセッサ名》と呼ぶことにします。この場合、データ *__weight* のアクセッサ名を *weight* としています。

ゲッタの本体では、データの値の返却を行います。

▪ セッタの定義

前置きとして“@アクセッサ名 .setter”を付けて定義します。

メソッドの名前は、ゲッタと同じ名前、すなわち、アクセッサ名とします。

セッタの本体で行うことは、仮引数で受け取った値を、データに代入することです。なお、本プログラムでは、体重が負値とならないよう、仮引数 *weight* に **0.0** 以下の値を受け取った場合、データ *__weight* には **0.0** を代入します。

▶ 文法的には、*weight* は、アクセッサ名ではなく、プロパティ名です。ゲッタの定義で *weight* というプロパティ名を作ると、その後で @プロパティ名 .setter の指定が可能になります。

▪ ゲッタとセッタとの呼出し

セッタとゲッタを定義すると、“**インスタンス名 . アクセッサ名**”形式の式で、値の取得・設定ができる（そのため、代入の右辺にも左辺にも置ける）ようになります。

本プログラムでは、*yamada.weight* への代入と値の取出しを行っています。

*

本プログラムでは、ゲッタとセッタの両方を定義しました。**ゲッタのみを定義すれば、「外部からいつでも値を調べられるものの、値を書きかえられないデータ属性」が実現できます。**

▶ セッタとゲッタの他に、もう一つ、データ属性を削除するためのデリータ（*deleter*）を定義することが可能です。*__weight* のデリータの定義は、次のようになります。

```
@weight.deleter
def weight(self):
    del self.__weight
```


Column 11-4 @property デコレータ

ここでは、@property デコレータについて補足します。

▪ デコレータ

デコレータとは、別の関数を返す関数を作り出すための仕組みです。左下のように定義しておくと、内部的に右下のように変換されます。

```
@deco                          def func(...):
def func(...):          ➡          関数本体
    関数本体                   func = deco(func)
```

すなわち、@deco 付きの関数 *func* を定義すると、*deco*(*func*) を *func* で呼び出せるようになります。

▪ property クラス

property は、標準で提供されるクラスであり、以下のデータをもちます（コンストラクタの呼出しでこれらの値をキーワード引数で指定できるようになっています）。

- 属性値を取得するための関数 **fget**
- 属性値を設定するための関数 **fset**
- 属性値を削除するための関数 **fdel**
- 属性の文書化文字列を表す **doc**