Chapter7　「円を動かすメソッド」の補足

　第7章では、「円を動かす」という処理をしています。その処理は、元の円を白色（背景色）で描画して消し、それから新しい位置に、円を描画し直すことで動かしています。

　たとえば、「Lesson 7-8　プログラムをブロック化して1つの機能を与える」のp.210の「example07-08-01.py」のmoveメソッドの処理が、それです。

```
def move(self, canvas):
    # いまの円を消す
    canvas.create_oval(self.x - 20, self.y - 20, self.x + 20, self.y + 20,
fill="white", width=0)
    # X座標、Y座標を動かす
    self.x = self.x + self.dx
    self.y = self.y + self.dy
    # 次の位置に円を描画する
    canvas.create_oval(self.x - 20, self.y - 20, self.x + 20, self.y + 20,
fill=self.color, width=0)
    # 端を超えていたら反対向きにする
    …略…
```

　tkinterライブラリの性質として、この方法で円を消すと、どんどんメモリが消費されるという問題があります。実際、長時間動かしっぱなしにすると、だんだんと速度が遅くなってくるのがわかります。

　実は、tkinterライブラリは、内部で「描画したもの」を、すべてメモリ上に保存して管理しています。そのため、この方法ですと、1回の処理につき「赤い円」と「白い円」を管理するためのメモリが消費されます。

　こうしたことがないようにするには、「白い円で上書きして消すのではなくて、赤い円自体を、キャンバスから取り除く」という方法をとります。

　方法は、いくつかあります。

■全部消す

　キャンバスのdeleteメソッドを「.delete("all")」として呼び出すと、描画したものすべてを削除できます。そのため、上のexample07-08-01.pyの例であれば、moveメソッドを次のように修正します。

```
def move(self, canvas):
    # いまの円を消す
    # ★ここを変更した★
    canvas.delete("all")
    # ★以下は同じ★
    # X座標、Y座標を動かす
    self.x = self.x + self.dx
    self.y = self.y + self.dy
    # 次の位置に円を描画する
    canvas.create_oval(self.x - 20, self.y - 20, self.x + 20, self.y + 20,
fill=self.color, width=0)
    # 端を超えていたら反対向きにする
    …略…
```

■オブジェクトを指定して消す

この方法は、すべてを消してしまうため、「Lesson 7-7　たくさんの円を動かそう」は対応できませんし、「Lesson 7-8」の後ろにある「■たくさんの円を動かす」のように修正する場合、そして「Lesson 7-9　円だけでなく、四角、三角も混ぜてみよう」のように、複数の円（もしくは四角形や三角形）を描画する場合には利用できません。全部ではなく、動かす対象のものだけを消す必要があるからです。

本文では説明していませんが、create_ovalメソッドやcreate_rectangleメソッド、create_polygonメソッドなどを呼び出すと、その戻り値として、描画したオブジェクトを取得できます。そしてdeleteメソッドで、そのオブジェクトを渡すと、それだけを削除できます。

たとえばexample07-09-01.py（p.229）では、drawメソッドで描画して、eraseメソッドで削除しています。これは、下記のように、「白い円」を描くことで描画しています。

【元のソース】

```
class Ball:
    def __init__(self, x, y, dx, dy, color):
        self.x = x
        self.y = y
        self.dx = dx
        self.dy = dy
        self.color = color

    def move(self, canvas):
        …略…

    def erase(self, canvas):
        canvas.create_oval(self.x - 20, self.y - 20, self.x + 20, self.y + 20,
fill="white", width=0)

    def draw(self, canvas):
        canvas.create_oval(self.x - 20, self.y - 20, self.x + 20, self.y + 20,
fill=self.color, width=0)

…略…
```

この部分を、次のように、deleteメソッドで削除するようにすると、この問題は解決します。

【変更後】

```
class Ball:
    def __init__(self, x, y, dx, dy, color):
        self.x = x
        self.y = y
        self.dx = dx
        self.dy = dy
        self.color = color
        ## 削除対象を示す変数を追加
        self.taisyou = None

    def move(self, canvas):
        …略…

    def erase(self, canvas):
        ## 削除対象が設定されていたら削除
        if self.taisyou is not None:
            canvas.delete(self.taisyou)

    def draw(self, canvas):
        ## 描画した円を削除対象として保存
        self.taisyou = canvas.create_oval(self.x - 20, self.y - 20, self.x + 20,
self.y + 20, fill=self.color, width=0)
```

「tkinterのオブジェクト」と「自分で作っている円や三角、四角などのオブジェクト」とが混同してわかりにくくなるため、本書では、話を濁していますが、本来であれば、このようにtkinerライブラリの性質上、「描いたものはdeleteメソッドで削除する」のが、正しい書き方です。