Python講座

2016/6/9(木)

今回の内容

- class①
- モデリング①②③
- class②
- 使用例
- 解説
- 属性の指定
- コンストラクタ
- self
- メソッド
- 演習問題①②補足

※P.76~

class①

- オブジェクト指向プログラミングと呼ばれる記法の代名詞のようなもの
- モデリングしたオブジェクトを中心としてプログラムを記述していく記法
- オブジェクトシステムの一種。様々なプログラミング言語で「クラス」という機構を通して「オブジェクト」を扱う記法が採用されている

モデリング(模型化)①

- コンピュータが処理を行えるように、理解できる形へ変換する作業のこと
- 作業時の大事なことは、対象をよく観察し重要な部分だけを選んで抜き出すこと
- 抜き出した情報を属性と呼び、それをまとめたものがオブジェクトと呼ばれる

モデリング(模型化)②

- 例）トランプ
- 実際のカードや柄はプログラムをするにあたって不要な情報
- 基本的に必要な情報はマークと数字の2つ
- この2つの情報をデータとして扱うようプログラムすることでトランプのゲームが再現できる

1. ロイヤル･フラッシュ （Royal Flush）
   1種類のスーツで最も数位の高い 5 枚の揃った役
2. ストレート･フラッシュ （Straight Flush）
   1種類のスーツで 5 枚の数位が連続して揃った役
3. フォー･オブ･ア･カインド （4 of a Kind）
   同数位のカードが 4 枚全て揃った役
4. フルハウス （Full House）
   同数位のカードが 3 枚と残りの 2 枚がペアの役
5. フラッシュ （Flush）
   1種類のスーツだけが 5 枚揃った役
6. ストレート （Straight）
   5 枚のカードの数位が連続して揃った役
7. スリー･オブ･ア･カインド （3 of a Kind）
   同数位のカードが 3 枚揃った役
8. ツーペア （Two Pair）
   同数位のペアが 2 組ある役
9. ワンペア （One Pair）
   同数位のペアが 1 組ある役
10. ノーペア （No Pair）
    ペアも何もない状態

＜手役比較のルール＞

- カードの数位は「A」が最も強く、以下「K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2」の順となる。
- ストレートで「A」を使う場合、「K」もしくは「2」の並びとして使える。
- スーツによるランク付けは行わない。手役を構成する数位のみを基準とする。
- ツーペア同士の勝敗判定は、「高ランクのペア」→「低ランクのペア」→「残りの 1 枚」の順で決める。

モデリング(模型化)③

オブジェクト	属性
トランプのカード	マーク、数字
ブロック崩しのボール	座標(x, y)、スピード
RPGのキャラ	攻撃力、防御力 .etc
物体の運動	質量、重力、摩擦 .etc

class②

- オブジェクトそのものを表現する仕組みとして「クラス」というものがある。
- 「クラス」というのはオブジェクトの種類を表すためのもので、それぞれのオブジェクトが持つ属性や性質をプログラム上で定義できる
- 文書によっては属性のことを「インスタンス変数」と呼んでいる場合もあるので注意

使用例

- クラスの名前は大文字で始めるのが一般的
- 赤枠:クラス名(青文字):
  クラスの内容
- 青枠:クラスを変数に代入(オブジェクトの生成)
- 緑枠:属性の指定

```
class Ball:
    pass

b = Ball()
b.x = 10
b.y = 20

print("{},{}".format(b.x,b.y))
```

```
10,20
```

解説

- クラスの内容はpass
(何もしないという意味)
- 変数にオブジェクトを作る
ことで、変数にクラスの内
容を適用できるようになる
- 変数.属性で変数(オブジェク
ト)の属性にアクセスできる

```
class Ball:
    pass

b = Ball()
b.x = 10
b.y = 20

print("{},{}".format(b.x,b.y))
```

```
10,20
```

属性の指定

- 先ほどの場合、属性がなんでも指定できるため収拾がつかない

↓

- 黒枠：クラス内に使う属性をメソッドとしてまとめておくことでわかりやすくする

※クラス内の関数をメソッドという

```
class Ball:
    def __init__(self):
        self.x = 10
        self.y = 20

b = Ball()
print("{},{}".format(b.x,b.y))
```

```
10,20
```

コンストラクタ

- def __init__(self):
- オブジェクトを生成する時に、必要な属性とその初期値を決めておけるメソッド
- コンストラクタと呼ばれる
- アンダーバー2つ、init、アンダーバー2つなので注意

```
class Ball:
    def __init__(self):
        self.x = 10
        self.y = 20
```

self

- selfはそのオブジェクト自身を示す。
- 変数にオブジェクトを生成
- 同時に属性(変数.x,変数.y)の値を生成
- selfはほとんどのメソッドに使用されるため重要

```
class Ball:
    def __init__(self):
        self.x = 10
        self.y = 20

b = Ball()
print("{},{}".format(b.x,b.y))
```

```
10,20
```

メソッド

- 赤枠：メソッドに引数を使うことも可能。selfは引数に数えられないためスルー
- 青枠：変数.メソッド名()でメソッドの内容が実行できる
- returnで値を返すことも可能
- つまり、使い方は関数と同じ

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

    def move(self):
        self.x += 15
        self.y += 25

b = Ball(10,20)
print("{},{}".format(b.x,b.y))
b.move()
print("{},{}".format(b.x,b.y))
```

```
10,20
25,45
```

演習問題①

- テーマ：クラスを使う
- 題材：前回の演習問題(円がランダムに動くやつ)
- 以下のメソッドを含むクラスを作る

1. コンストラクタ　※円も描画
2. ボールの動作
3. 画面端の動作

演習問題②

- テーマ：トランプの山札を作る

1. マークと数字を扱うCardクラスを用意
2. Trumpクラスで、Cardクラスを使ってマークと数字を指定し山札に追加する ※13×4 = 計52枚
3. 山札からランダムに五枚選び、手札とする
   その手札と五枚が抜き取られた山札を表示させる

演習問題②補足

- Card クラス
1. コンストラクタ
- Trumpクラス
1. コンストラクタ
2. 山札の表示
3. 手札の表示