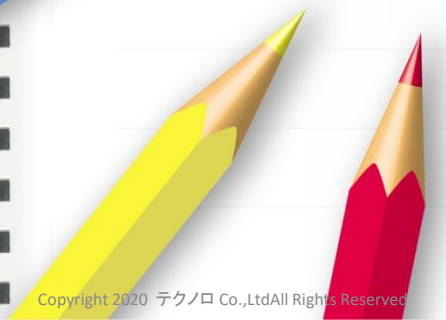
プログラミング教室のテクノロ

Pythonの道



本格的なゲーム開発②(前編)

もくじ

・パズルゲームをつくる（前半）

　複数のアルゴリズムを組み込む方法を学ぶ

ゲームの仕様を考える

本格的なゲーム開発にあたり、ゲームルール、画面構成、処理の流れなどの仕様を考える。

ゲームルール

①画面をクリックした位置にネコブロックを1つ配置する

②ネコを配置すると、画面上部から複数のネコが落ちてくる

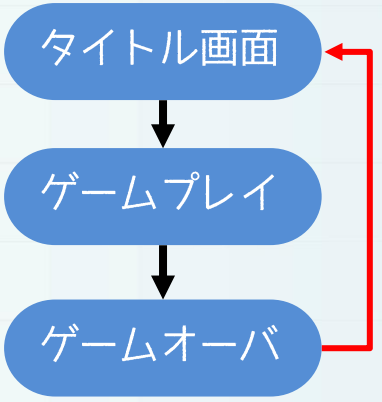
③ネコは下のマスに何もないと落下し、画面下から積みあがっていく

④同じ色のネコを縦、横、斜めいずれか3つ以上揃えると、消すことができる

⑤一列でも最上段に達してしまうとゲームオーバになる

画面構成

処理の流れ



マウス入力を組み込む①

bind()命令とイベントを受け取る関数を使ってマウスポインタを動かしたときの入力（座標）を受け付けるようにする。

```
import tkinter  ・・・・tkinterモジュールの呼出し

mouse_x = 0     ・・・・マウスポインタのx座標
mouse_y = 0     ・・・・マウスポインタのy座標

def mouse_move(e):  ・・・・マウスを動かした時に実行する関数
    global mouse_x,mouse_y ・・・・グローバル変数として扱うと宣言
    mouse_x = e.x  ・・・mouse_xにマウスポインタのx座標を代入
    mouse_y = e.y  ・・・mouse_yにマウスポインタのy座標を代入

def game_main():   ・・・・リアルタイム処理を行う関数
    fnt = ("Times New Roman",30)・・・・フォントを指定する変数
    txt = "マウス座標({},{})".format(mouse_x,mouse_y)
    cvs.delete("test")・・文字列の削除                    ・・表示する文字列
    cvs.create_text(456,384,text=txt,fill = "black"  ,font = fnt
     ・・・・キャンバスに文字列を表示する                    ,tag ="test")
    root.after(100,game_main)・・・0.1秒後に「game_main」関数を実行
```

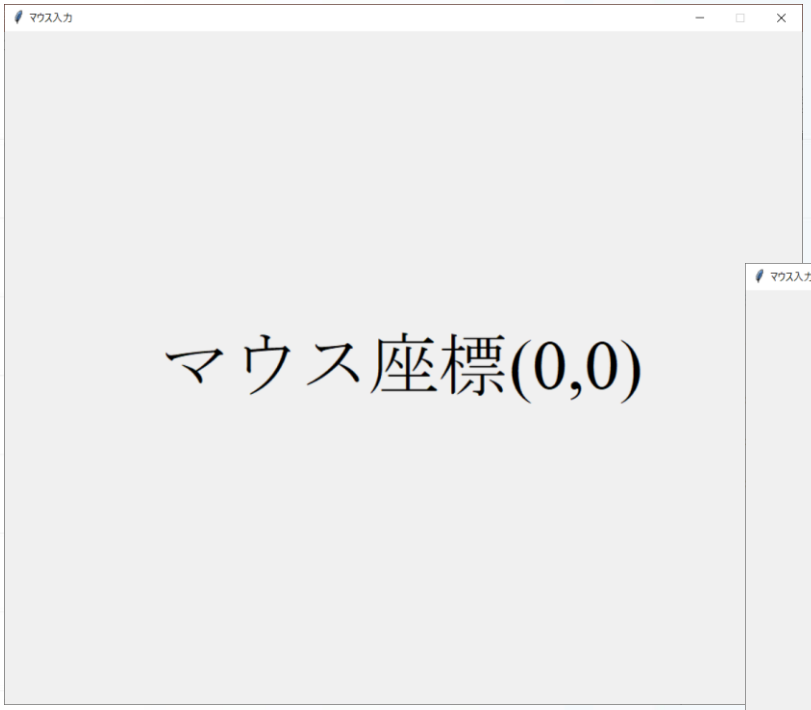
次のページに続く

マウス入力を組み込む①

```
root = tkinter.Tk()・・・・ウィンドウのオブジェクトを作る
root.title("マウス入力")・・・・タイトルを指定
root.resizable(False,False)・・・ウィンドウサイズを変更できないようにする
root.bind("<Motion>",mouse_move)
                                    ・・・マウスが動いた時に実行する関数を指定
cvs = tkinter.Canvas(root,width = 912,height=768)
                                              ・・・キャンバスの部品を作る
cvs.pack()・・・キャンバスを配置する
game_main()・・・メインの処理を行う関数を呼び出す
root.mainloop()  ・・・ウィンドウを表示
```

マウス入力を組み込む①

「Run Module」またはF5キーを押してプログラムを実行する。



マウスを動かすと座標が書き換わることを確認しよう。

format()命令を理解する

format()命令を使うと文字列内に変数を埋め込むことができる。
文字列内に変数を埋め込むことで文字列を動的に変えられる。

・format()命令の書き方

a ＝“ {},{}” .format(“ 文字列1” ,” 文字列2”)

【例】

a ＝“ {}は{}です。” .format(“ 本日” ,” 10日”)

print(a)

⇒本日は10日です。

b ＝“ {}君は{}のテストで{}点を取りました。” .format(“ 太郎” ,” 数学” ,” 100”)

print(b)

⇒太郎君は数学のテストで100点を取りました。

format()命令を理解する

【応用例】→動的な文字列を代入する

```
import datetime
c=datetime.datetime.now()
d = "現在時刻は{}時{}分です。".format(c.hour,c.minute)
print(d)
```

⇒現在時刻は〇時〇分です。※プログラムを実行した時間が表示される

【datetime関数のおさらい】

現在日時を表示するときはdatetime.now()と書く。

```
import datetime
現在日時 = datetime.datetime.now()
print(現在日時)
```

現在時刻を呼び出すクラス

```
2020-06-06 13:57:15.279526
>>>
```

bind()命令を理解する

イベントを受け取るにはbind()命令を使う。キーイベントを取得するプログラムを作ってみよう。

・bind()命令の書き方

bind(“ <イベント> ” , イベント発生時に実行する関数名)

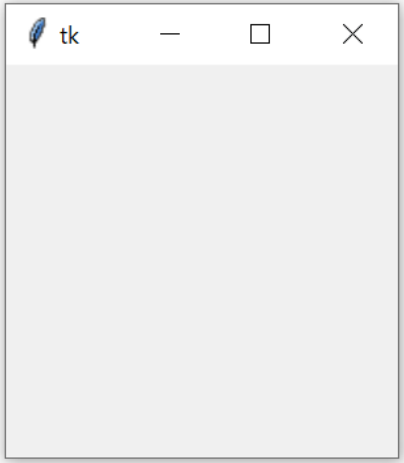
scratchの〇キーが押されたときブロックと同じ

<イベント>	イベントの内容
<KeyPress>	キーを押した
<Key>	
<KeyRelease>	キーを離した
<Motion>	マウスポインタを動かした
<ButtonPress>	マウスボタンをクリックした
<Button>	

eventを表す引数eは引数を使用しない場合でも記載する必要がある。

```
import tkinter
def key_down(e):
    print("キーが押されました")
root = tkinter.Tk()
root.bind("<KeyPress>",key_down)
root.mainloop()
```

```
キーが押されました
キーが押されました
キーが押されました
キーが押されました
キーが押されました
```



bind()命令を理解する

押されたキーのキーコードを表示するプログラムを作る。

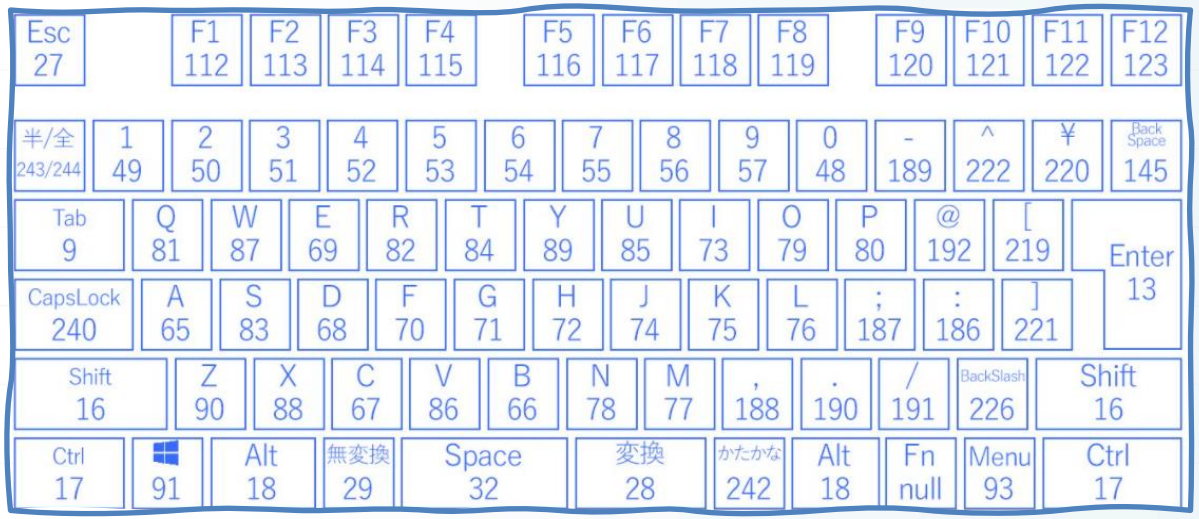
```
import tkinter  ・・・・tkinterモジュールの呼出し
def key_down(e): ・・・ key_down関数を呼び出した時に実行する関数を定義
    key = e.keycode ・・・ 押されたキーのコードを変数「key」に代入
    print("押されたボタンのキーコードは",key,"です。")
                                        ・・・ キーコードの表示
root = tkinter.Tk()  ・・・ ウィンドウのオブジェクトを作成
root.title("キーコードを取得") ・・・ ウィンドウのタイトルを表示する
root.bind("<KeyPress>",key_down)
                    ・・・ bind()命令でキーを押した時に実行する関数を指定する
root.mainloop()  ・ ・ ・ ウィンドウを表示する
```

bind()命令を理解する

「Run Module」またはF5キーを押してプログラムを実行する。

```
押されたボタンのキーコードは 49 です。
押されたボタンのキーコードは 50 です。
押されたボタンのキーコードは 51 です。
押されたボタンのキーコードは 52 です。
押されたボタンのキーコードは 53 です。
押されたボタンのキーコードは 54 です。
押されたボタンのキーコードは 55 です。
押されたボタンのキーコードは 56 です。
押されたボタンのキーコードは 57 です。
押されたボタンのキーコードは 48 です。
```

押されたキーのキーコードがシェルウィンドウに表示される。



マウス入力を組み込む②

次にマウスのボタンをクリックしたときに「１」をマウスのボタンを離したときに「0」と表示するプログラムを追加する。

```
import tkinter  ・・・・tkinterモジュールの呼出し

mouse_x = 0     ・・・・マウスポインタのx座標
mouse_y = 0     ・・・・マウスポインタのy座標

def mouse_move(e):  ・・・・マウスを動かした時に実行する関数
    global mouse_x,mouse_y ・・・・グローバル変数として扱うと宣言
    mouse_x = e.x  ・・・mouse_xにマウスポインタのx座標を代入
    mouse_y = e.y  ・・・mouse_yにマウスポインタのy座標を代入

def mouse_press(e):  ・・・・マウスボタンをクリックした時に実行する関数
    global mouse_c   ・・・・グローバル変数として扱うと宣言
    mouse_c = 1      ・・・・mouse_cに1を代入

def mouse_release(e): ・・・・マウスボタンを離した時に実行する関数
    global mouse_c    ・・・・グローバル変数として扱うと宣言
    mouse_c = 0       ・・・・mouse_cに0を代入
```

次のページに続く

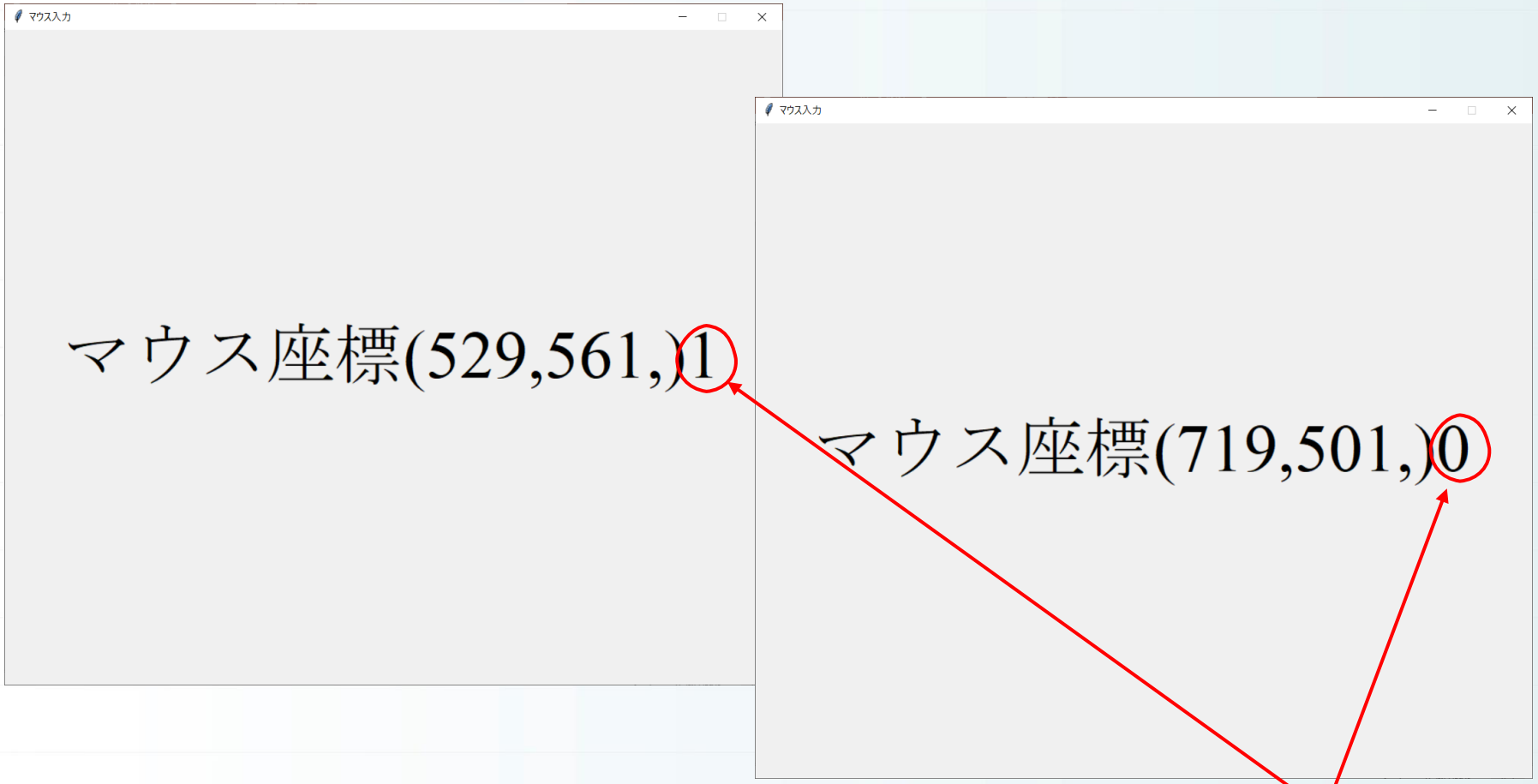
マウス入力を組み込む②

```
def game_main():    ・・・・リアルタイム処理を行う関数
    fnt = ("Times New Roman",60) ・・・・フォントを指定する変数
    txt = "マウス座標({},{},){}".format(mouse_x,mouse_y,mouse_c)
                                                ・・表示する文字列
    cvs.delete("test") ・・文字列の削除
    cvs.create_text(456,384,text=txt,fill = "black",font = fnt,tag
="test") ・・・・キャンバスに文字列を表示する
    root.after(100,game_main) ・・・0.1秒後に「game_main」関数を実行

root = tkinter.Tk()  ・・・・ウィンドウのオブジェクトを作る
root.title("マウス入力")  ・・・・タイトルを指定
root.resizable(False,False) ・・・ウィンドウサイズを変更できないようにする
root.bind("<Motion>",mouse_move)・・マウスが動いた時に実行する関数を指定
root.bind("<ButtonPress>",mouse_press)・マウスをクリックした時に実行する関数を指定
root.bind("<ButtonRelease>",mouse_release)
                        ・・・・マウスボタンを離した時に実行する関数を指定
cvs = tkinter.Canvas(root,width = 912,height=768)
                                        ・・・キャンバスの部品を作る
cvs.pack()  ・・・キャンバスを配置する
game_main()  ・・・メインの処理を行う関数を呼び出す
root.mainloop()  ・・・ウィンドウを表示
```

マウス入力を組み込む②

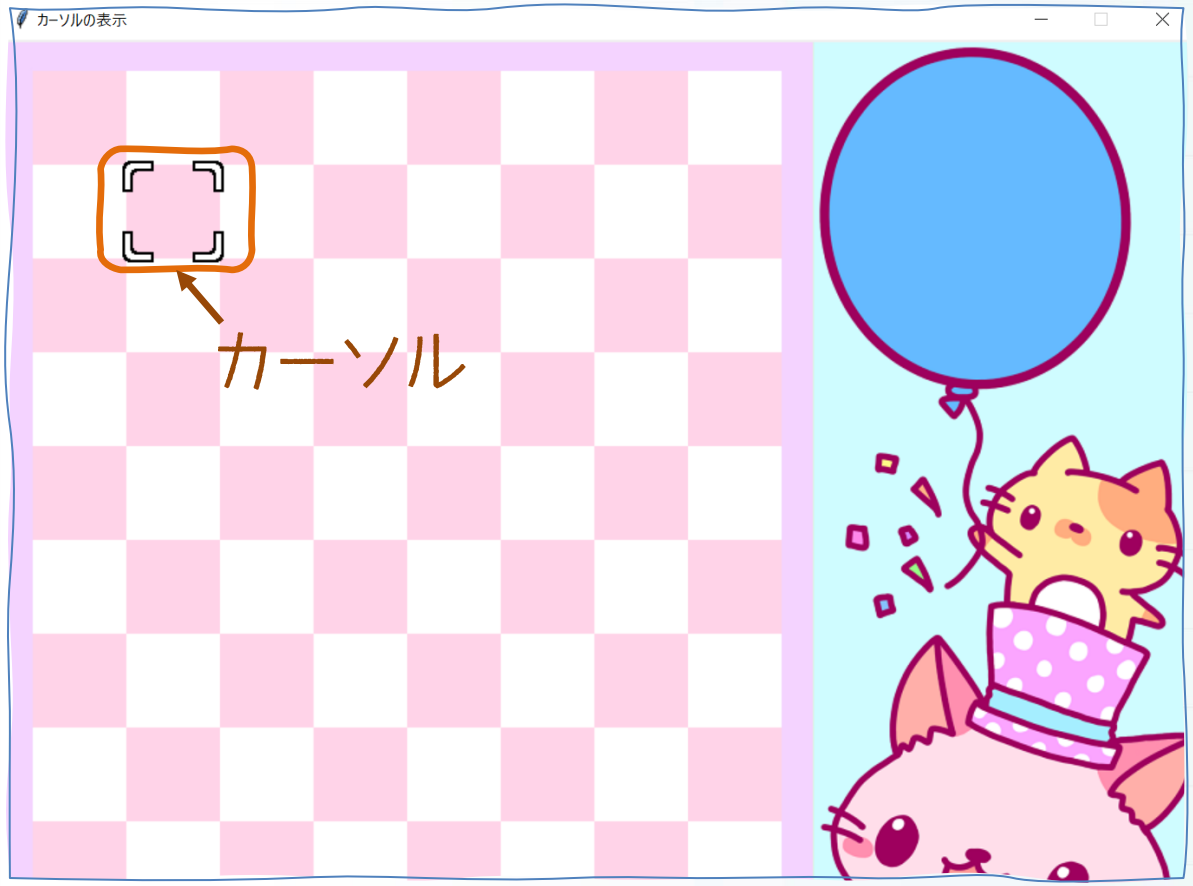
「Run Module」またはF5キーを押してプログラムを実行する。



マウスボタンをクリックする数字が書き換わることを確認しよう。

ゲーム用カーソルを作る

マウスポインタの現在位置を示す小さな目印のことをカーソルという。取得したマウスの座標値からゲーム用のカーソルを操作できるようにする。



ゲーム用カーソルを作る

```
import tkinter ・・・・tkinterモジュールの呼出し

mouse_x=0 ・・・・マウスポインタのx座標
mouse_y=0 ・・・・マウスポインタのy座標

def mouse_move(e):    ・・・・マウスを動かした時に実行する関数
    global mouse_x,mouse_y    ・・・・グローバル変数として扱うと宣言
    mouse_x = e.x    ・・・mouse_xにマウスポインタのx座標を代入
    mouse_y = e.y    ・・・mouse_yにマウスポインタのy座標を代入

def game_main():・・・・リアルタイム処理を行う関数
    global mouse_x,mouse_y ・・・・グローバル変数として扱うと宣言
    cvs.delete("CURSOR") ・・・・カーソルを消す
    cvs.create_image(mouse_x,mouse_y,image=cursor,tag="CURSOR")
                                        ・・・・新たな位置にカーソルを表示する
    root.after(100,game_main)  ・・・0.1秒後に「game_main」関数を実行

root = tkinter.Tk()  ・・・・ウィンドウのオブジェクトを作る
root.title("カーソルの表示")・・・・タイトルを指定
root.resizable(False,False) ・・・ウィンドウサイズを変更できないようにする
root.bind("<Motion>",mouse_move)
                                        ・・マウスを動かした時に実行する関数を指定
cvs = tkinter.Canvas(root,width=912,height=768)
                                ・・・キャンバスの部品を作る
```

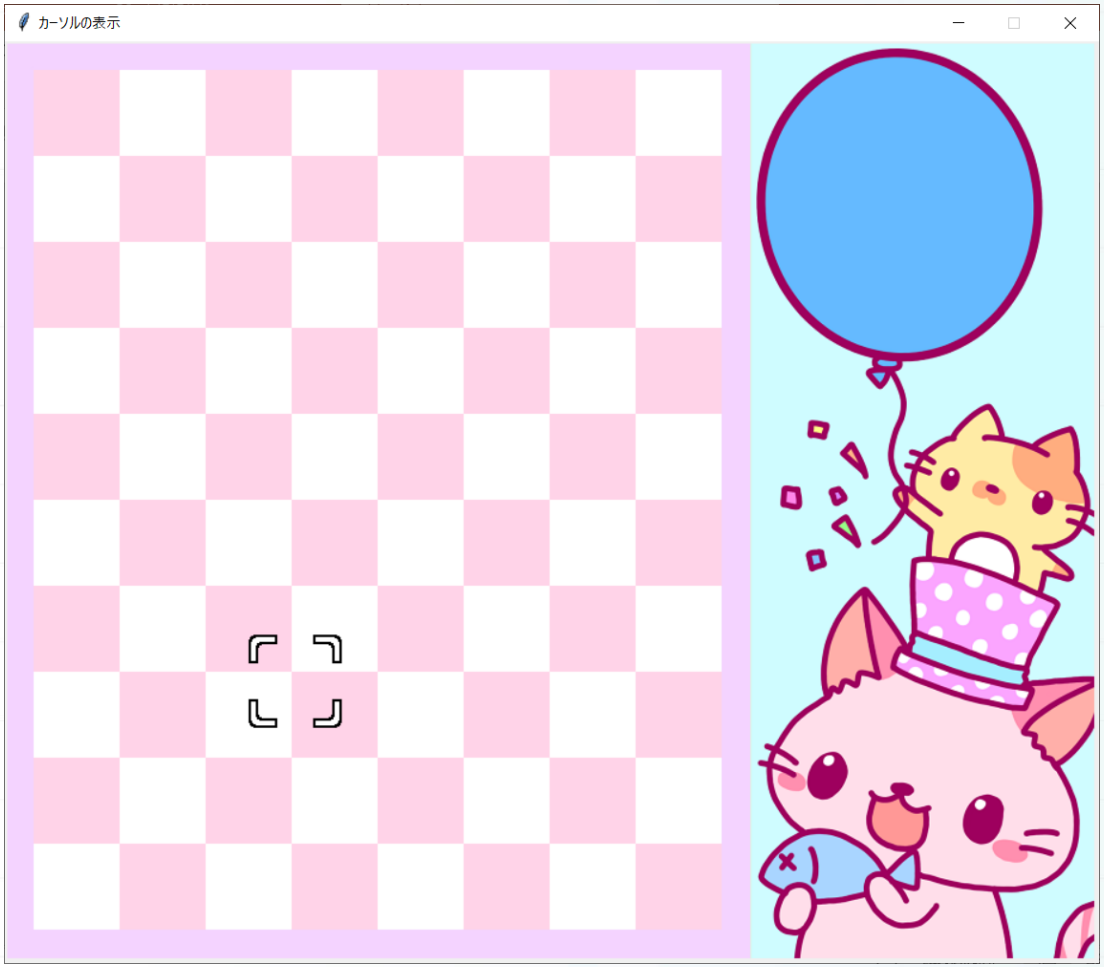
次のページに続く

ゲーム用カーソルを作る

```
cvs.pack() ・・・キャンバスを配置する
bg= tkinter.PhotoImage(file="neko_bg.png")・・・背景画像の読み込み
cursor=tkinter.PhotoImage(file="neko_cursor.png")
                                        ・・・カーソル画像の読み込み
cvs.create_image(456,384,image=bg) ・・・キャンバス上に背景を描く
game_main() ・・・メインの処理を行う関数を呼び出す
root.mainloop() ・・・ウィンドウを表示
```

ゲーム用カーソルを作る

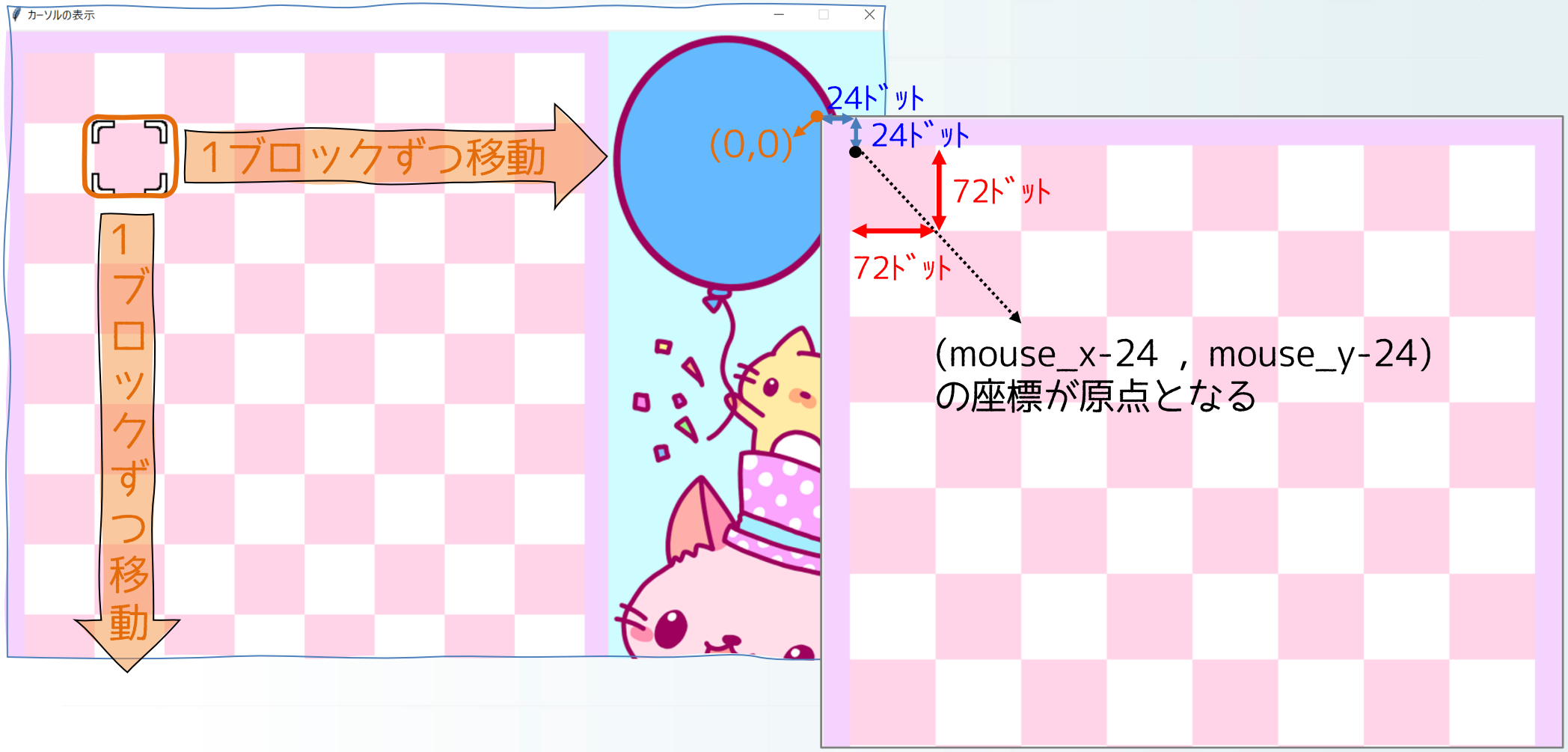
「Run Module」またはF5キーを押してプログラムを実行する。



マウスを動かすとカーソルの位置が変わることを確認しよう。

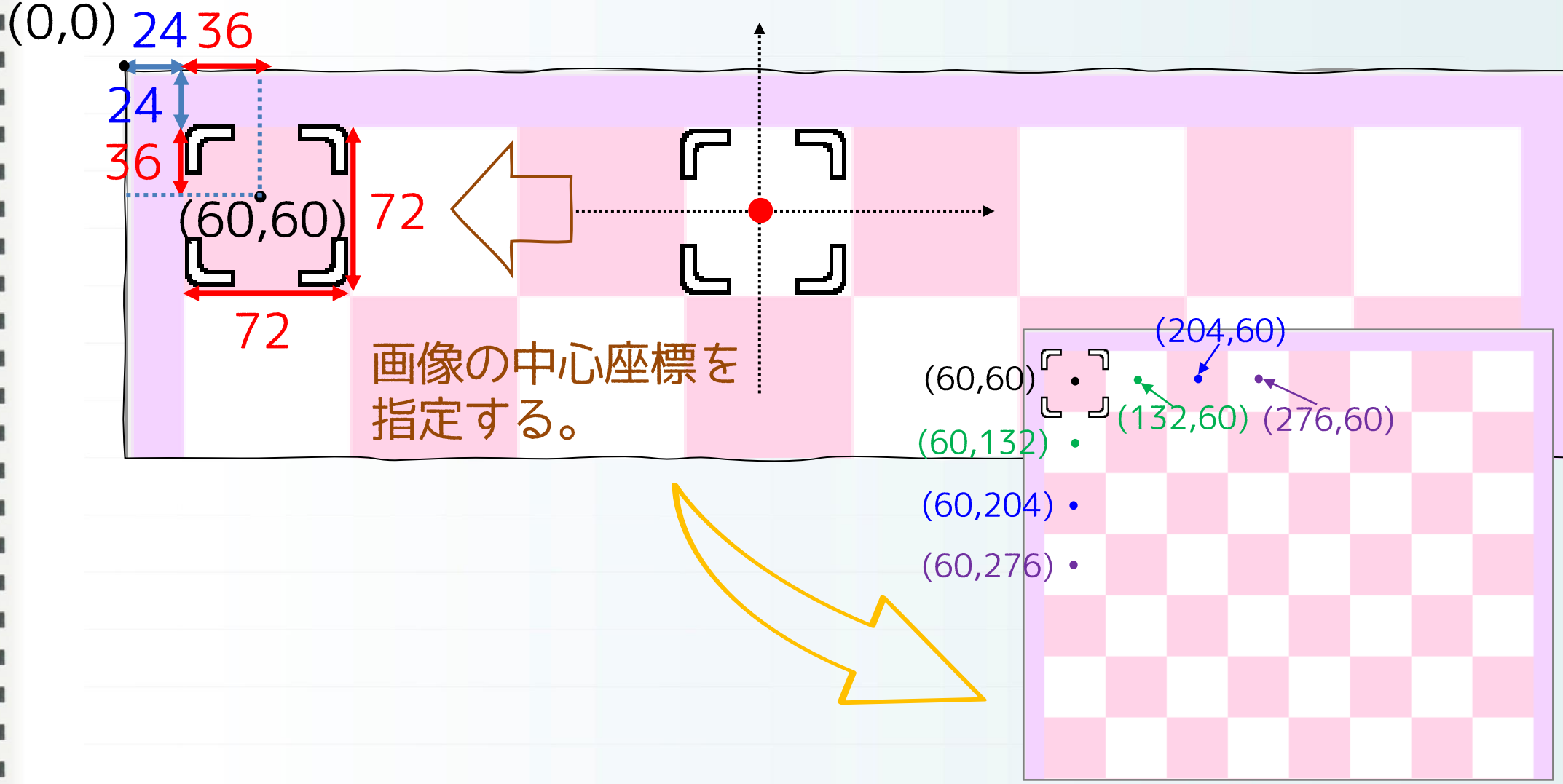
ゲーム用カーソルを作る

カーソルの移動できる範囲をピンク色のブロックと白色のブロックに限定してみよう。移動範囲を限定するにはマウスポインタの座標からカーソルの位置を計算する必要がある。



ゲーム用カーソルを作る

カーソルの位置はカーソル画像の中心座標を指定する必要がある。

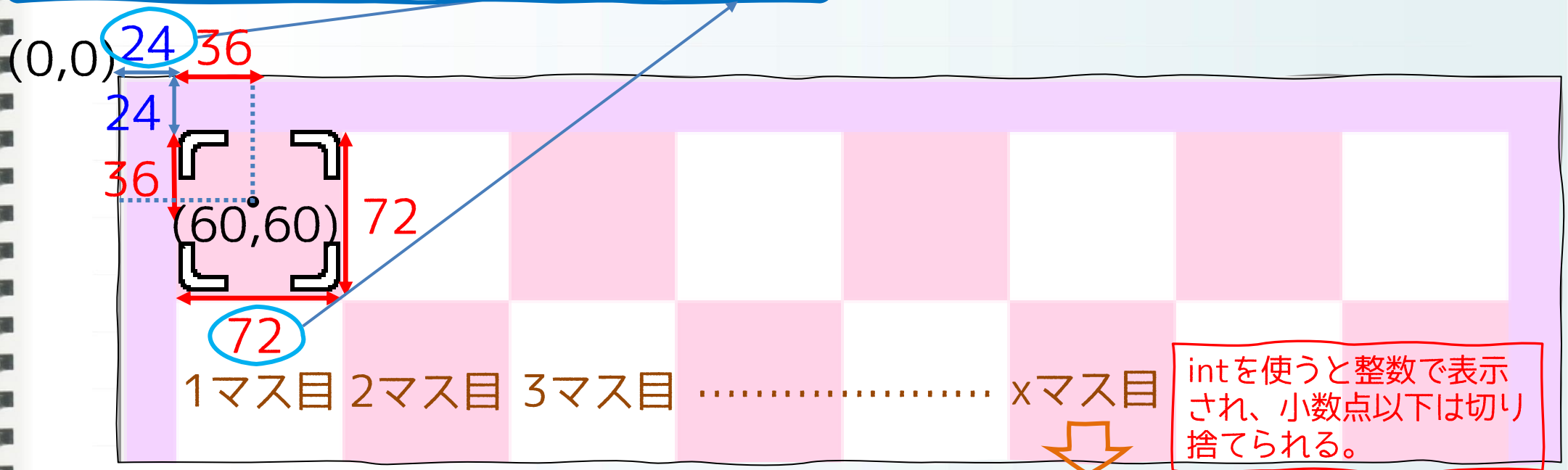


ゲーム用カーソルを作る

1マス分づつ、移動させるために1マス分の座標を格納する変数を作る。

カーソル画像をセットする位置(座標)を(cursor_x, cursor_y)と定義する。
変数「cursor_x」に何マス目かを求める計算式を代入する。

```
cursor_x = int((mouse_x-24)/72)
```

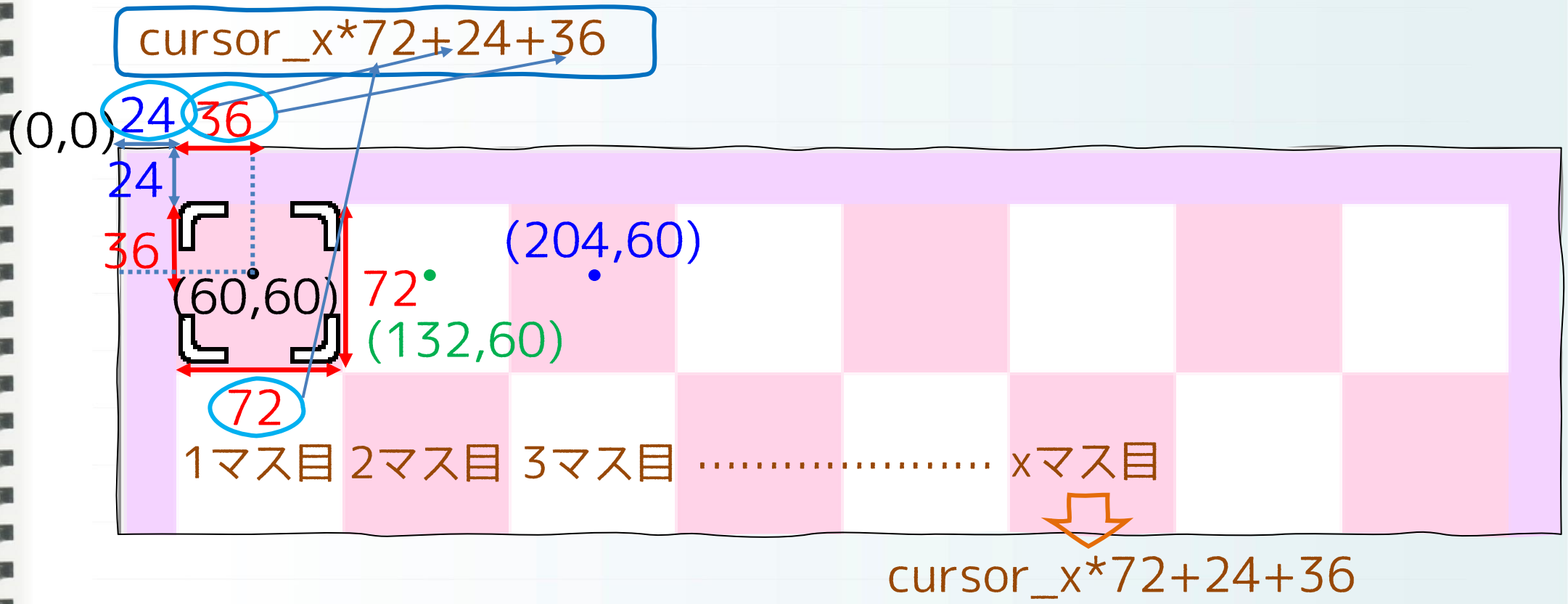



```
x = int((mouse_x-24)/72)
```

【例1】x座標が100のとき： (100-24)÷ 72⇒1.05 →1マス目
【例2】x座標が150のとき： (150-24)÷ 72⇒1.75 →1マス目

ゲーム用カーソルを作る

何マス目かを求めることができたら、カーソルの位置を計算することができる。



【例1】1マス目のときのx座標： 1× 72+24+36⇒132
【例2】2マス目のときのx座標： 2× 72+24+36⇒204

ゲーム用カーソルを作る

```
import tkinter ・・・・tkinterモジュールの呼出し
cursor_x=0 ・・・・カーソルの横方向の位置(左から何マス目にあるか)
cursor_y=0 ・・・・カーソルの縦方向の位置(上から何マス目にあるか)
mouse_x=0 ・・・・マウスポインタのx座標
mouse_y=0 ・・・・マウスポインタのy座標

def mouse_move(e): ・・・・マウスを動かした時に実行する関数
    global mouse_x,mouse_y ・・・・グローバル変数として扱うと宣言
    mouse_x = e.x ・・・mouse_xにマウスポインタのx座標を代入
    mouse_y = e.y ・・・mouse_yにマウスポインタのy座標を代入

def game_main(): ・・・・リアルタイム処理を行う関数
    global cursor_x,cursor_y ・・・・グローバル変数として扱うと宣言
    cursor_x = int((mouse_x-24)/72) ・・x座標の値から何マス目かを計算
    cursor_y = int((mouse_y-24)/72) ・・y座標の値から何マス目かを計算
    cvs.delete("CURSOR") ・・・・カーソルを消す
    cvs.create_image(cursor_x*72+36+24,cursor_y*72+36+
    24,image=cursor,tag="CURSOR") ・・新たな位置にカーソルを表示する
    root.after(100,game_main) ・・・0.1秒後に「game_main」関数を実行
```

次のページに続く

ゲーム用カーソルを作る

```
root = tkinter.Tk()　・・・・ウィンドウのオブジェクトを作る
root.title("カーソルの表示")・・・・タイトルを指定
root.resizable(False,False)・・・ウィンドウサイズを変更できないようにする
root.bind("<Motion>",mouse_move)
                                ・・・・マウスを動かした時に実行する関数を指定
cvs = tkinter.Canvas(root,width=912,height=768)
cvs.pack()　・・・キャンバスを配置する　　・・・キャンバスの部品を作る

bg= tkinter.PhotoImage(file="neko_bg.png")・・・背景画像の読み込み
cursor=tkinter.PhotoImage(file="neko_cursor.png")
                                ・・・カーソル画像の読み込み
cvs.create_image(456,384,image=bg)・・・キャンバス上に背景を描く
game_main()・・・メインの処理を行う関数を呼び出す
root.mainloop()・・・ウィンドウを表示
```

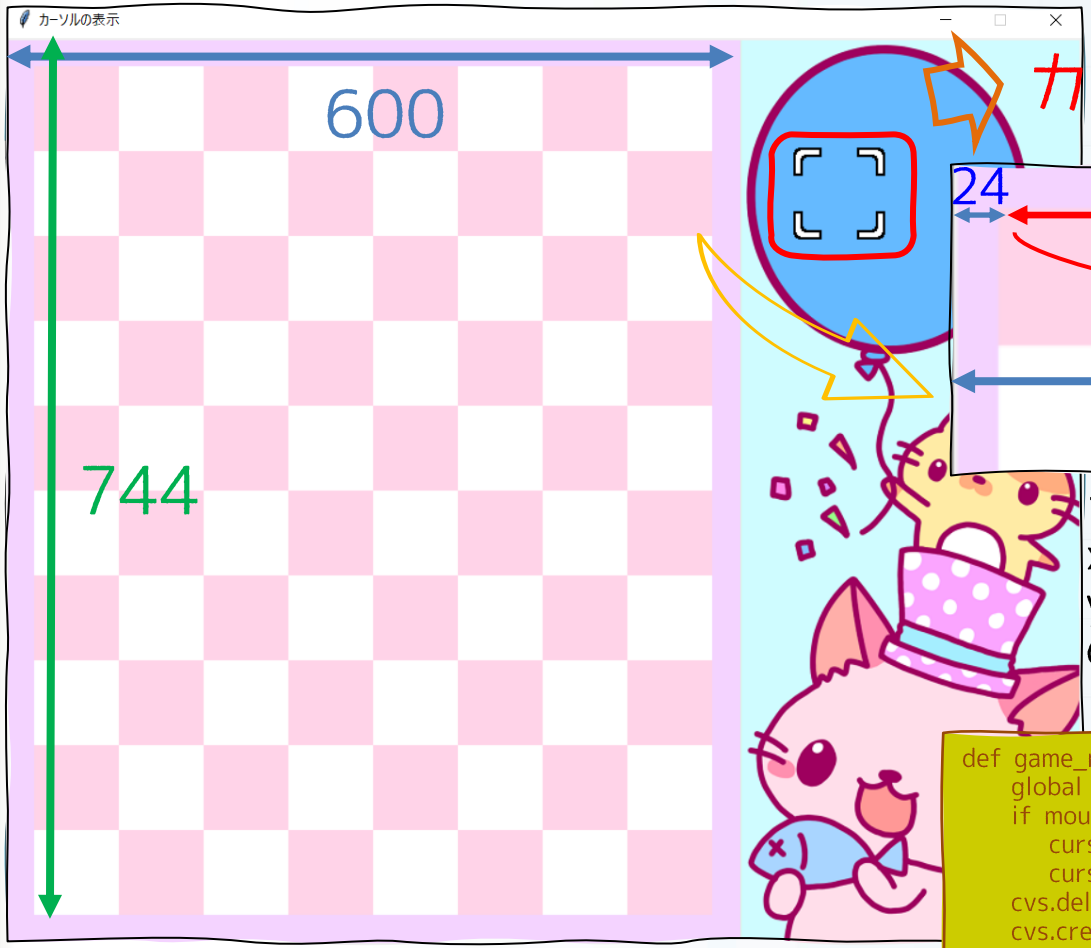
ゲーム用カーソルを作る

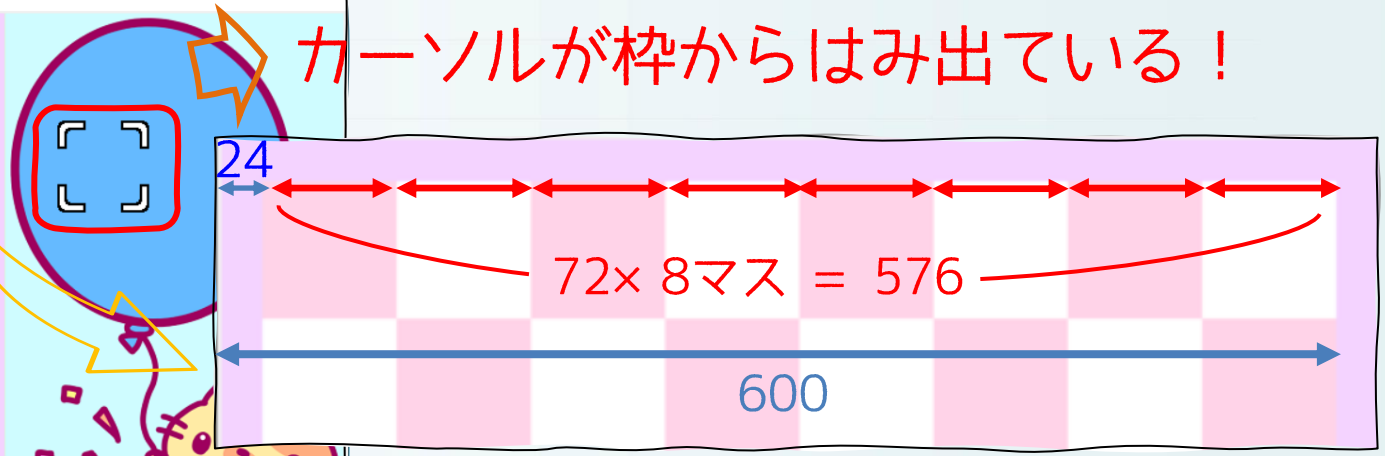
「Run Module」またはF5キーを押してプログラムを実行する。

カーソルがマス目にそって動くことを確認しよう。

ゲーム用カーソルを作る

カーソルを1マスづつ、移動させることができた。しかし、一番端までくるとはみ出してしまう！？
枠からはみ出さないように条件を追加する。





カーソルが画面からはみ出ないようにするには、
x座標＜600ピクセル(72ピクセル× 8マス+24ピクセル)
y座標＜744ピクセル(72ピクセル× 10マス+24ピクセル)
のときにgame_main()関数が実行されるようにする。

```
def game_main():
    global cursor_x,cursor_y
    if mouse_x < 72*8+24 and mouse_y<72*10+24:
        cursor_x = int((mouse_x-24)/72)
        cursor_y = int((mouse_y-24)/72)
    cvs.delete("CURSOR")
    cvs.create_image(cursor_x*72+36+24,cursor_y*72+36+24,image=cursor,tag="CURSOR")
    root.after(100,game_main)
```

ゲーム用カーソルを作る

```
import tkinter ・・・・tkinterモジュールの呼出し
cursor_x=0 ・・・・カーソルの横方向の位置(左から何マス目にあるか)
cursor_y=0 ・・・・カーソルの縦方向の位置(上から何マス目にあるか)
mouse_x=0 ・・・・マウスポインタのx座標
mouse_y=0 ・・・・マウスポインタのy座標

def mouse_move(e): ・・・・マウスを動かした時に実行する関数
    global mouse_x,mouse_y ・・・・グローバル変数として扱うと宣言
    mouse_x = e.x ・・・mouse_xにマウスポインタのx座標を代入
    mouse_y = e.y ・・・mouse_yにマウスポインタのy座標を代入

def game_main(): ・・・・リアルタイム処理を行う関数
    global cursor_x,cursor_y ・・・・グローバル変数として扱うと宣言
    if mouse_x < 72*8+24 and mouse_y<72*10+24: ・・追加
        cursor_x = int((mouse_x-24)/72) ・・x座標の値から何マス目かを計算
        cursor_y = int((mouse_y-24)/72) ・・y座標の値から何マス目かを計算
    cvs.delete("CURSOR") ・・・・カーソルを消す
    cvs.create_image(cursor_x*72+36+24,cursor_y*72+36+
    24,image=cursor,tag="CURSOR") ・・新たな位置にカーソルを表示する
    root.after(100,game_main) ・・・0.1秒後に「game_main」関数を実行
```

次のページに続く

ゲーム用カーソルを作る

```
root = tkinter.Tk()　・・・・ウィンドウのオブジェクトを作る
root.title("カーソルの表示")・・・・タイトルを指定
root.resizable(False,False)・・・ウィンドウサイズを変更できないようにする
root.bind("<Motion>",mouse_move)
                                ・・・・マウスを動かした時に実行する関数を指定
cvs = tkinter.Canvas(root,width=912,height=768)
cvs.pack()　・・・キャンバスを配置する　　・・・キャンバスの部品を作る

bg= tkinter.PhotoImage(file="neko_bg.png")　・・・背景画像の読み込み
cursor=tkinter.PhotoImage(file="neko_cursor.png")
                                ・・・カーソル画像の読み込み
cvs.create_image(456,384,image=bg)　・・・キャンバス上に背景を描く
game_main()　・・・メインの処理を行う関数を呼び出す
root.mainloop()・・・ウィンドウを表示
```

ゲーム用カーソルを作る

「Run Module」またはF5キーを押してプログラムを実行する。

復習＆チャレンジ

ここまで習ったことをScratchでもできるかチャレンジしてみよう。
その過程でScratchでできること、Pythonでないとできないことを整理してみよう。

メモ

プログラミング教室のテクノロ

なまえ：