プログラミング教室のテクノロ

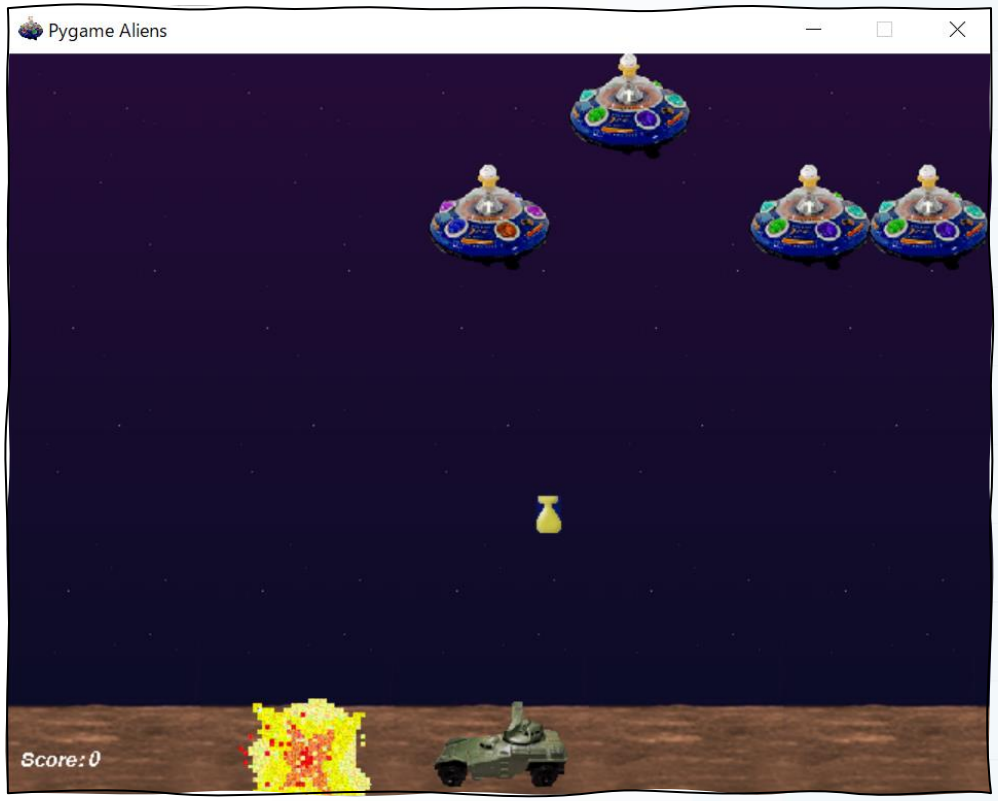
Pythonの道

pygameの使い方②

もくじ

・pygameの使い方(①の続き)

pygameを使って本格的なゲーム開発を学ぶ



キーで絵を動かす

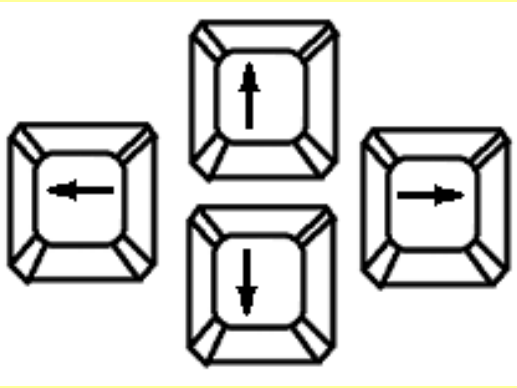
絵を動かせるようになったので、キー操作で絵を動かしてみよう。まずはどのキーが押されているかを調べる必要がある。

どのキーが押されているかを調べる
キー変数 = pg.key.get_pressed()

返ってきたキー変数には「今どのキーが押されているか」がわかる情報が入っているので、その中を調べる。例えば、「右キーが押されたか」は「キー変数[pg.K_RIGHT]」の値がTrueかFalseであるかを調べ、Trueなら押されていることになる。

```
Key = pg.Key.get_pressed()

if Key[pg.K_RIGHT]:
    右キーが押された
if Key[pg.K_LEFT]:
    左キーが押された
```



キーで絵を動かす

「左右のキーが押されたかを調べるプログラム」を作ってみよう。押されたキーを取得して、押されたのがpg.K_RIGHTなら「右」、pg.K_LEFTならば「左」と表示する。

```
import pygame as pg,sys  #ゲームの準備をする
pg.init()
screen = pg.display.set_mode((800,600)) ・・・復習1
while True:  #この下をずっとループする
    screen.fill("white")  #画面を初期化する
    key = pg.key.get_pressed() #ユーザからの入力を調べる
    if key[pg.K_RIGHT]:  ・・・ 右キーが押された時の処理
        print("右")
    if key[pg.K_LEFT]:   ・・・ 左キーが押された時の処理
        print("左")
    pg.display.update()  #画面を表示する
```

次のページに続く

キーで絵を動かす

```
    pg.time.Clock().tick(60)
    for event in pg.event.get(): #閉じるボタンが押されたら終了する
        if event.type == pg.QUIT:
            pg.quit()
            sys.exit()
```

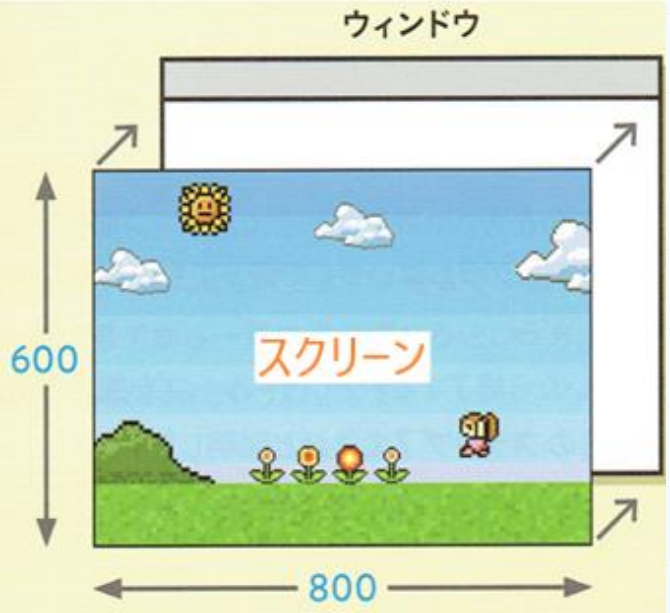
「Run Module」またはF5キーを押してプログラムを実行する。

```
pygame 2.0.1 (SDL 2.0.14, Python 3.8.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
右
右
右
左
左
```

復習1: スクリーンを作る

ライブラリ名.display.set_mode((〇,〇))を使って、ゲームを描くためのスクリーンを作る。tkinterモジュールのキャンバス命令と同じもの。

```
スクリーン ＝ pg.display.set_mode((800,600))
```



スクリーンを定義した後、
スクリーン.fill()命令で画面を初期化したり、
py.draw.rect(スクリーン,〇〇,△△)命令で
画面を表示したりすることができる。

変数「スクリーン」に横800ピクセル、縦600ピクセルのスクリーンを作る命令
tkinterモジュールのキャンバス命令と同じ

```
キャンバス ＝ tkinter.Canvas(ウィンドウ , width ＝ 800 , height ＝ 600)
```

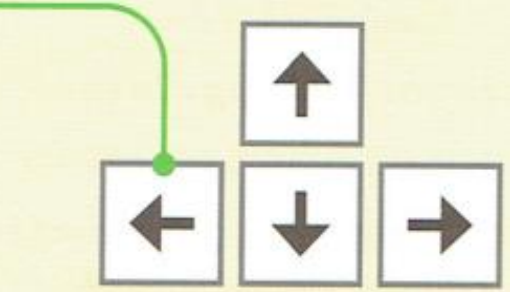
キーで絵を動かす

「今どのキーが押されているか」が分かるようになったところで、これを使って画像を動かす。右キーが押されたら右に、左キーが押されたら左に画像を移動させる。

```
vx = 0 移動量を表す変数vx
key = pg.key.get_pressed()

if key[pg.K_RIGHT]:
    vx = 5

if key[pg.K_LEFT]:
    vx = -5
```



myrect.x + vx

キーで絵を動かす

「左右キーでキャラクタが動くプログラム」を作ってみよう。

```
#ゲームの準備をする
import pygame as pg, sys
pg.init()
screen = pg.display.set_mode((800, 600))
imageR = pg.image.load("playerR.png") ・・・ 画像ファイルの読み込み
myrect = pg.Rect(0, 0, 0, 0) ・・・復習1
#この下をずっとループする
while True:
    #画面を初期化する
    screen.fill("WHITE")
    vx = 0
```

次のページに続く

キーで絵を動かす

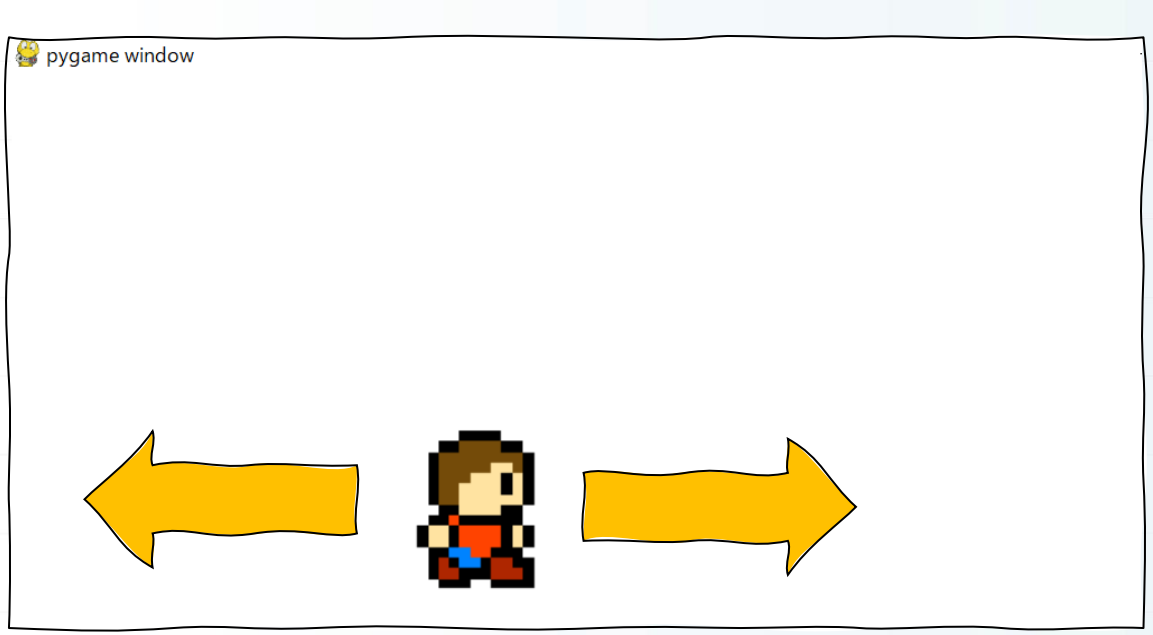
```
#ユーザーからの入力を調べる
key = pg.key.get_pressed()
#絵を描いたり、判定したりする
if key[pg.K_RIGHT]:    ・・・ 右向き矢印キーが押された時、vxに5を
                              代入する
    vx = 5
if key[pg.K_LEFT]:     ・・・ 左向き矢印キーが押された時、vxに5を
                              代入する
    vx = -5
myrect.x = myrect.x + vx    ・・・ キャラクタのX座標を変更する
screen.blit(imageR, myrect)  ・・・ 復習2
#画面を表示する
pg.display.update()  ・・・ 復習3
pg.time.Clock().tick(60) ・・・ 復習3
```

次のページに続く

キーで絵を動かす

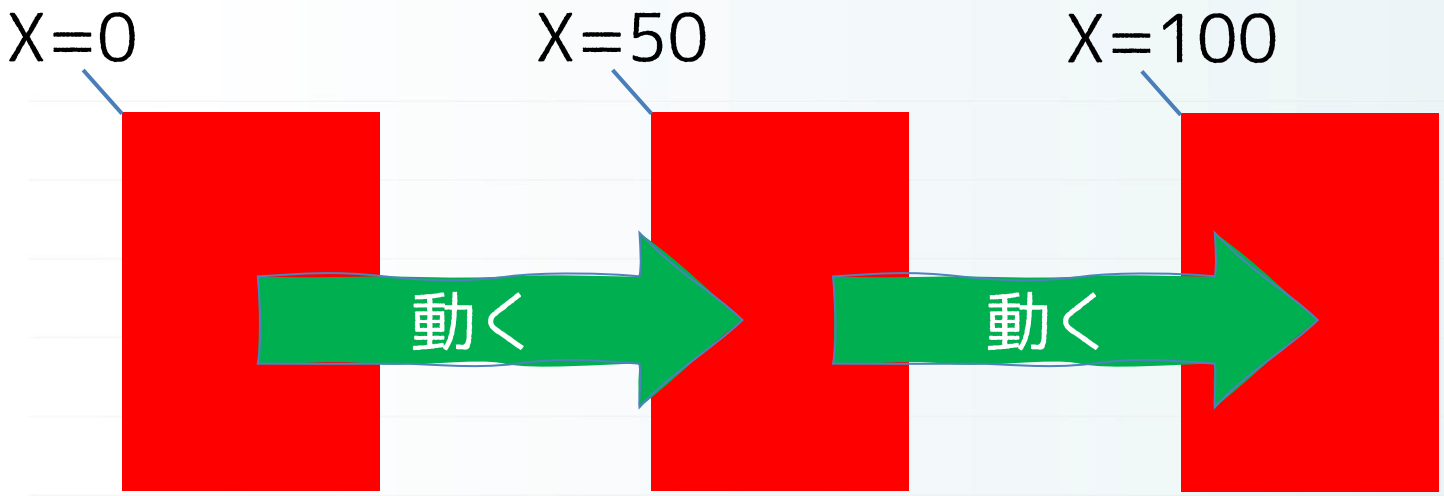
```
#閉じるボタンが押されたら、終了する
for event in pg.event.get():
    if event.type == pg.QUIT:
        pg.quit()
        sys.exit()
```

「Run Module」またはF5キーを押してプログラムを実行する。



復習1:グラフィックスを動かす

止まっている絵を動かす。絵を動かすためにプログラムをループさせて、X座標、Y座標を移動させていく。



ゲームの中では、ただキャラクタを移動させるだけでなく、「壁や敵と衝突したか」を調べる必要がある。この「X座標、Y座標、幅、高さ」をひとまとめにしたものが「Rect」命令になる。

位置と大きさをまとめて扱う
変数 ＝ pg.Rect(X, Y, 幅 , 高さ)

Rectの中の引数には、変数に「.」と「X,Y,width,height」をつける。

復習2: スクリーンに画像を描く

スクリーンに画像を描く場合、image.load関数を使って画像を読み込む。次にblit関数を使って読み込んだ画像を描画する。

画像を読み込む
画像変数=pg.image.load(“ 画像ファイルパス”)

読み込んだ画像を描画する
スクリーン.blit(画像変数,(X,Y))
※blit 転送する

復習3: time.Clock().tick()命令

処理速度を調整するためにtime.Clock.tick()命令を使う

```
pg.display.update()
pg.time.Clock().tick(60)
```

pg.time.Clock().tick(60)と記載することで1秒間に60回以下のスピードにすることができる。

スピードを調整する
pg.time.Clock().tick()
1秒間にこの回数以下のスピードにする

左に進むときに絵を反転させる

右へ動かすときは問題ないが、左に動かすと後ろ向きに動いているように見える。そこで「左に動くとき」は左向きの絵で移動させる。transform.flip関数を使うと左右反転の絵を作ることができる。

画像を上下左右に反転させる

画像変数 ＝ pg.transform.flip(画像変数,左右反転,上下反転)

※flip ひっくり返る

キーで絵を動かす

「左右キー」で絵が反転するプログラムを作る。

```
#ゲームの準備をする
import pygame as pg, sys
pg.init()
screen = pg.display.set_mode((800, 600))
imageR = pg.image.load("playerR.png")
imageL = pg.transform.flip(imageR,True,False) ・・・画像を左右に反転させる
myrect = pg.Rect(0,0,0,0) ・・・画像の座標を管理する変数
right_flag = True ・・・右向き矢印キーが押されたかを判断する変数
#この下をずっとループする
while True:
    #画面を初期化する
    screen.fill("WHITE")
    vx = 0
```

次のページに続く

キーで絵を動かす

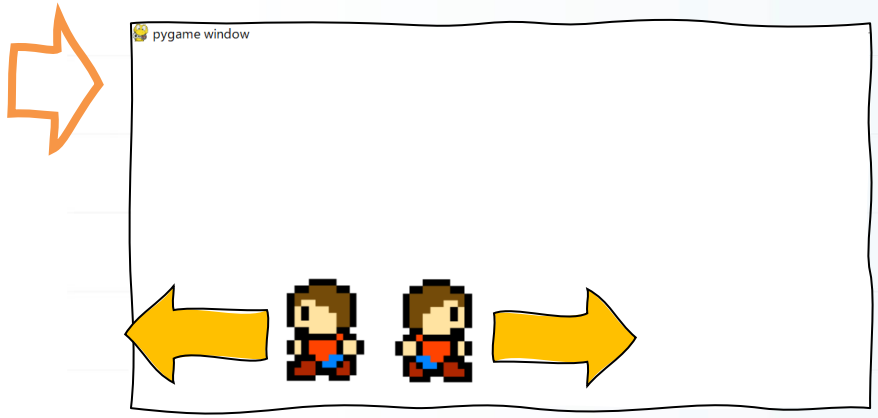
```
#ユーザーからの入力を調べる
key = pg.key.get_pressed()
#絵を描いたり、判定したりする
if key[pg.K_RIGHT]:
    vx = 5
    right_flag = True   ・・・ライトフラグをON（True）にする
if key[pg.K_LEFT]:
    vx = -5
    right_flag = False   ・・・ライトフラグをOFF（False）にする
myrect.x = myrect.x + vx
if right_flag == True :
    screen.blit(imageR,myrect)
else:
    screen.blit(imageL,myrect)
```

次のページに続く

キーで絵を動かす

```
#画面を表示する
pg.display.update()
pg.time.Clock().tick(60)
#閉じるボタンが押されたら、終了する
for event in pg.event.get():
    if event.type == pg.QUIT:
        pg.quit()
        sys.exit()
```

「Run Module」またはF5キーを押してプログラムを実行する。



マウスで絵を動かす

キーボードの次はマウスで絵を動かす。マウスが押されたかどうかを調べるにはmouse.get_pressed関数を使う。この関数を使うとマウスのボタンが今、押されているかどうかが分かる。さらにmouse.get_pos関数を使うとマウスがどこを指しているかを調べることができる。

マウスが押されたかを調べる

```
マウス変数 = pg.mouse.get_pressed()
(mx,my) = pg.mouse.get_pos()
```

「マウスのどのボタンが押されているか」を調べる「マウス変数 = pg.mouse.get_pressed()」関数では、左ボタンがおされるとマウス変数[0]、右ボタンが押されるとマウス変数[2]が入る。

また、「マウスが画面のどこを指しているか」を調べる「(mx,my) = pg.mouse.get_pos()」関数では、マウスが指している位置がmx,myの２つの変数に入る。

マウスで絵を動かす

マウスを押した位置を表示するプログラムを作る。

```
#ゲームの準備をする
import pygame as pg, sys
pg.init()
screen = pg.display.set_mode((800, 600))
#この下をずっとループする
while True:
    #画面を初期化する
    screen.fill("WHITE")
    #ユーザーからの入力を調べる
    mdown = pg.mouse.get_pressed()
    (mx, my) = pg.mouse.get_pos()
```

・・・マウスのどのボタンが押されたかを調べる関数

・・・マウスが画面のどこを指しているかを調べる関数

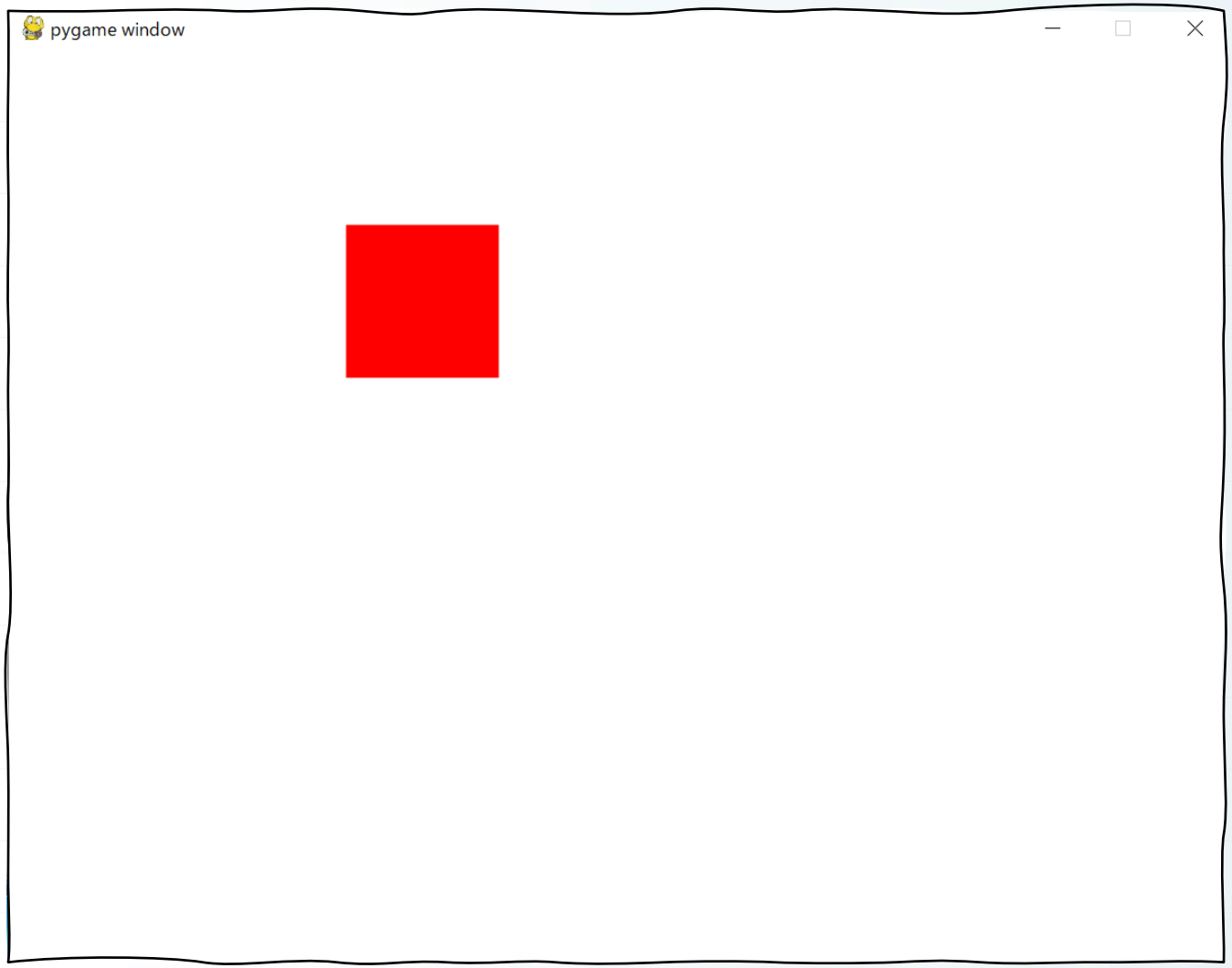
次のページに続く

マウスで絵を動かす

```
#絵を描いたり、判定したりする
if mdown[0]: ・・・マウスの左ボタンがおされたときの処理
    pg.draw.rect(screen, "RED", (mx-50, my-50, 100, 100))
#画面を表示する
pg.display.update()
pg.time.Clock().tick(60)
#閉じるボタンが押されたら、終了する
for event in pg.event.get():
    if event.type == pg.QUIT:
        pg.quit()
        sys.exit()
```

マウスで絵を動かす

「Run Module」またはF5キーを押してプログラムを実行する。

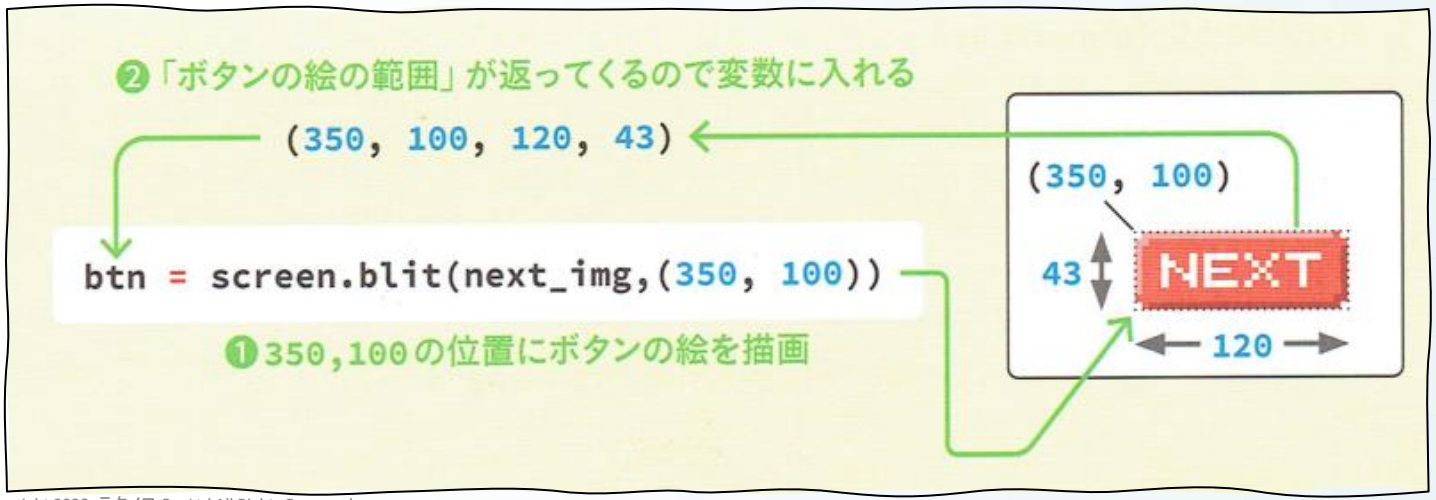


ボタンを作ろう

マウスで押した位置が、ボタンの中なら押したと判断する。
画像を描くときに使うblit関数には、表示した画像の範囲(X,Y,幅,高さ)を返すという機能がついている。つまり、ボタンの絵を描けば、自動的に「ボタンの絵の範囲」も手に入れることができる。

画像を描画して、その範囲を取得する
画像の範囲 ＝ screen.blit(画像変数,(X,Y))

「blit」とは 画像処理などで大量のデータをいちどに転送する操作のこと。
日本語では「貼りつける」ぐらいの意味。

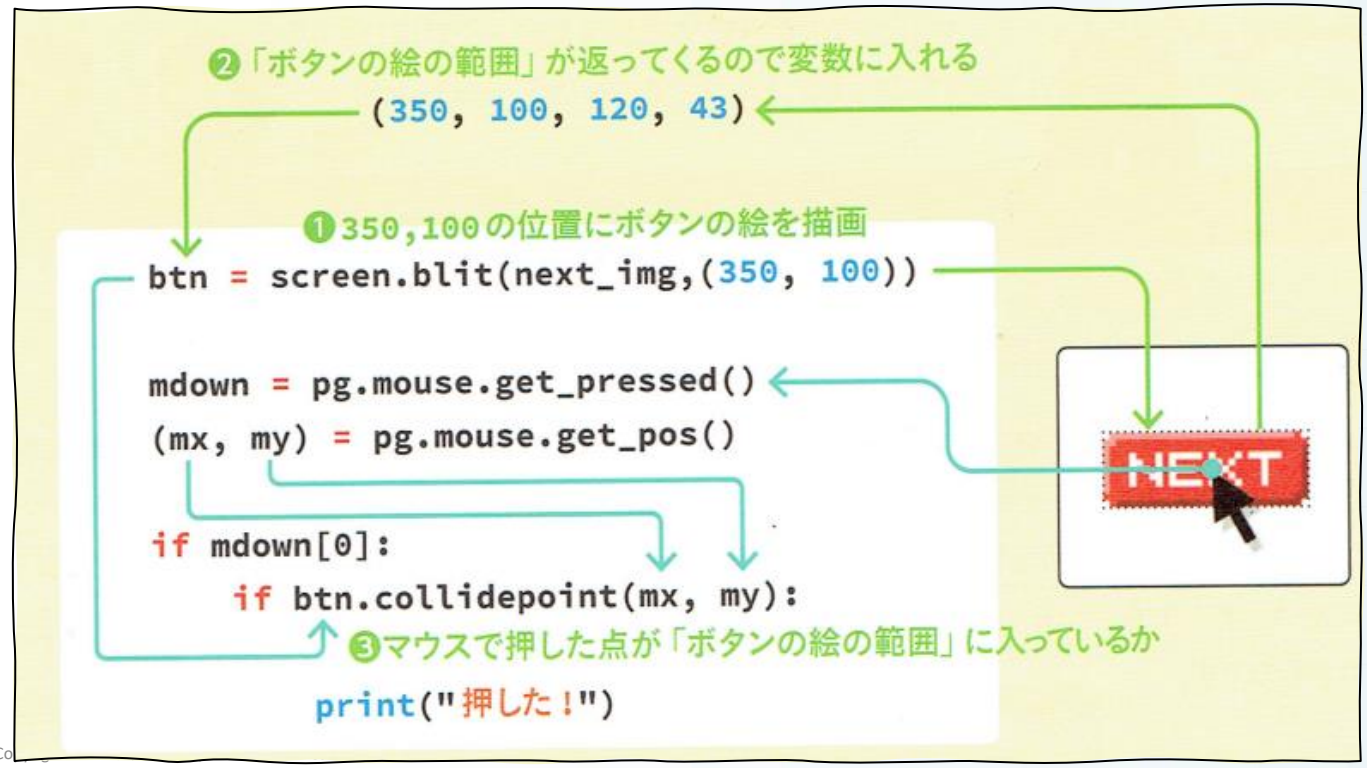


ボタンを作ろう

さらにRect.collidepoint関数を使うと「ある点が、ある範囲内に入っているか」を調べることができる。つまり、「マウスで押した点が、ボタンの範囲内に入っているか」を調べることができる。

座標が描写範囲内にあるかを調べる
ある範囲.collidepoint(X,Y)

collidepoint:衝突点



ボタンを作ろう

ここまでの知識を使ってボタンをマウスで押したか調べるプログラムを作る。

```
#ゲームの準備をする
import pygame as pg, sys
pg.init()
screen = pg.display.set_mode((800, 600))
next_img = pg.image.load("nextbtn.png") ・・・画像の読み込み
#この下をずっとループする
while True:
    #画面を初期化する
    screen.fill("WHITE") ・・・画面を真っ白に塗りつぶす
    btn = screen.blit(next_img,(350, 200)) ・・・画像を描画しその範囲を取得する
    #ユーザーからの入力を調べる
    mdown = pg.mouse.get_pressed() ・マウスのどのボタンが押されたかを調べる
    (mx, my) = pg.mouse.get_pos() ・マウスが画面のどこを指しているかを調べる
```


次のページに続く

ボタンを作ろう

```
#絵を描いたり、判定したりする
if mdown[0]: ・・・マウスの左ボタンがおされたときの処理
    if btn.collidepoint(mx, my): ・・・座標が描写範囲内にある場合
        print("押した!")
else:
    print("押していない")
#画面を表示する
pg.display.update()
pg.time.Clock().tick(1)
#閉じるボタンが押されたら、終了する
for event in pg.event.get():
    if event.type == pg.QUIT:
        pg.quit()
        sys.exit()
```

ボタンを作ろう

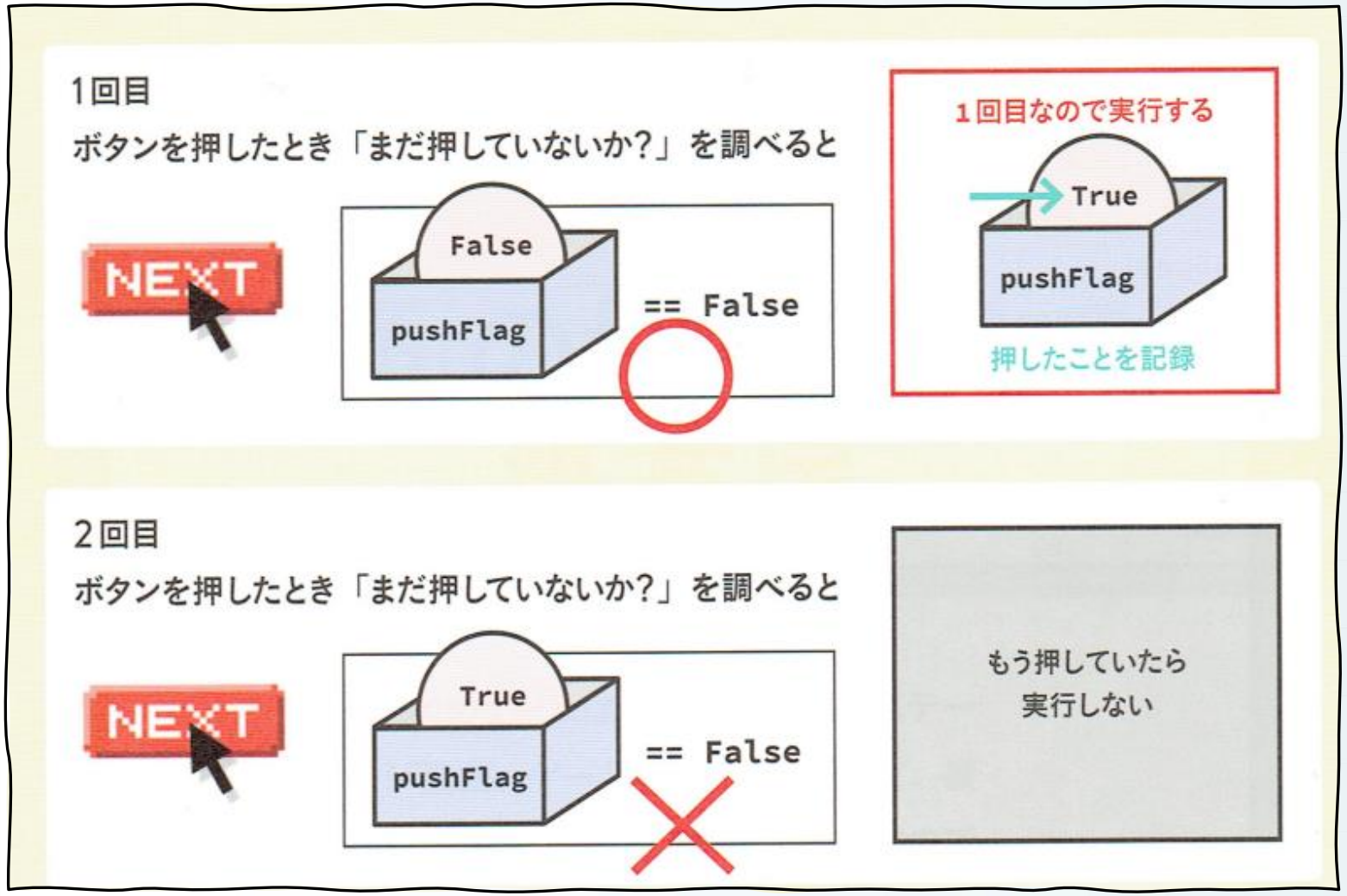
「Run Module」またはF5キーを押してプログラムを実行する。

```
pygame 2.0.1
Hello from the
押してない
押した
押してない
押してない
押してない
押した
押した
```

ボタンの範囲をクリックすると「押した」と表示される

押したら1回だけ実行するボタン

ボタンを押したことは分かったけど、少し押しただけでも何回も「押した！」と実行され続けてしまう。そこで「押したら1回だけ実行するボタン」に修正する。



押したら1回だけ実行するボタン

押したら1回だけ実行するボタンをプログラムする

```
#ゲームの準備をする
import pygame as pg, sys
pg.init()
screen = pg.display.set_mode((800, 600))
next_img = pg.image.load("nextbtn.png") ・・・画像の読み込み
#この下をずっとループする
while True:
    #画面を初期化する
    screen.fill("WHITE") ・・・画面を真っ白に塗りつぶす
    btn = screen.blit(next_img,(350, 200)) ・・・画像を描画しその範囲を
                                               取得する
    #ユーザーからの入力を調べる
    mdown = pg.mouse.get_pressed() ・マウスのどのボタンが押されたかを調べる
    (mx, my) = pg.mouse.get_pos() ・マウスが画面のどこを指しているかを調べる
```

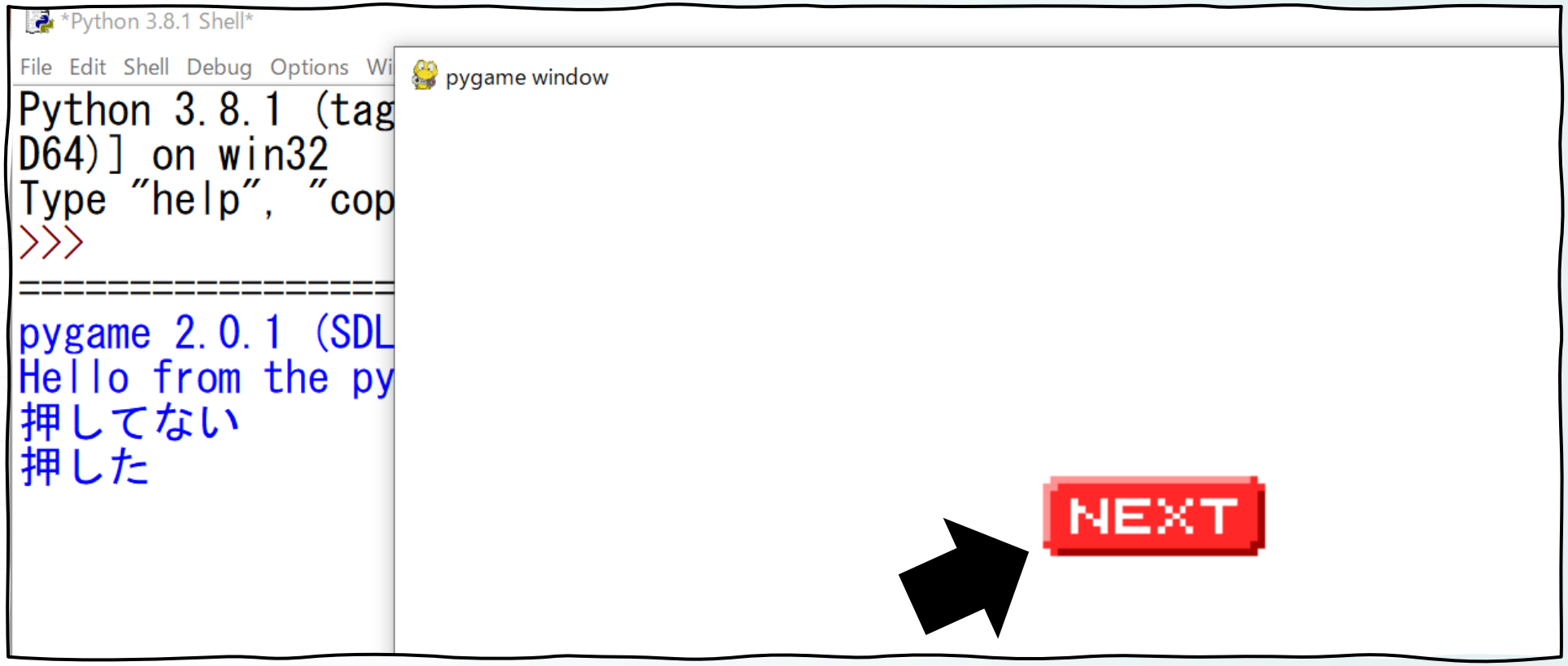

次のページに続く

押したら1回だけ実行するボタン

```
#絵を描いたり、判定したりする
if mdown[0]:  ・・・マウスの左ボタンがおされたときの処理
    if btn.collidepoint(mx, my) and pushFlag == False:
        print("押した!")
        pushFlag= True
else:
    print("押していない")
    pushFlag = False
#画面を表示する
pg.display.update()
pg.time.Clock().tick(1)
#閉じるボタンが押されたら、終了する
for event in pg.event.get():
    if event.type == pg.QUIT:
        pg.quit()
        sys.exit()
```

ボタンを作ろう

「Run Module」またはF5キーを押してプログラムを実行する。



ボタンの範囲をクリックすると「押した」と表示される。一回だけしか、おせないが、ボタンを離すともう一回、押せることを確認しよう。

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：