プログラミング演習３（Python）⑩ GUI ツールキット tkinter を使う

Python では、操作できる画面を作るときは、標準ライブラリの GUI ツールキット「tkinter」を使う。この tkinter を使うとウィンドウの上にボタンやラベルを並べて操作ができる画面を作ることができる。

ここでは、まずプログラムを実行するとおみくじの結果を表示するプログラムを作り、それをウィンドウ上のボタンを押したら、そのウィンドウにおみくじの結果が出るように改良する。

１０－１．おみくじプログラム

py10-1.py　おみくじの結果を表示する

```
001  import random
002  omikuji = [”大吉”,”中吉”,”小吉”,”吉”,”末吉”,”凶”]
003  print(random.choice(omikuji))
```

（解説）

１行目　random モジュールをインポート

２行目　表示するおみくじの結果の文字をリストに入れる

３行目　リストからランダムに要素を一つ選択して返す random.choice()でおみくじの結果を表示する

（実行結果）

```
PS C:\Users\[redacted]\Desktop\py> py py10-1.py
凶
```

１０－２．GUI ツールキットを使って、ボタンを押して結果を表示する

py10-2.py　学習用データを読み込んで表示する

```
001  import tkinter as tk
002  import random
003  def omikuji():
004      omikujikekka = ["大吉","中吉","小吉","吉","末吉","凶"]
005      lbl.configure(text=random.choice(omikujikekka))
006  gamen = tk.Tk()
007  gamen.geometry("150x100")
008  lbl=tk.Label(text="LABEL")
009  btn=tk.Button(text="PUSH",command=omikuji)
010  lbl.pack()
011  btn.pack()
```

```
012 tk.mainloop()
```

（解説）

１行目　tkinter をインポートして tk で使えるようにする

２行目　random をインポートして乱数が使えるようにする

３～５行目　おみくじの結果を得る omikuji 関数を設定する

　４行目　omikujikekka リストの要素を設定する

　５行目　ラベルの文字を omikujikekka リストの要素をランダムで選んだ一つにする

６，７行目　150×100 のサイズのウィンドウを作る。※x は半角英数字の x(エックス)

　　ここでは gamen 変数に入れて処理をしているが、root 変数が使われることが多い。

８行目　lbl というラベルを作り、LABEL という文字が表示されるように設定する

９行目　btn というボタンを作り、PUSH という文字が表示されるようにし、

　　ボタンを押したら omikuji 関数が実行されるよう設定する

10 行目　lbl という名前のラベルを配置する

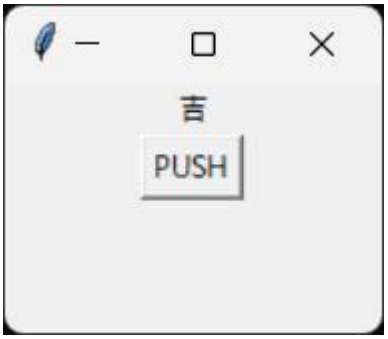
11 行目　btn という名前のボタンを配置する

12 行目　lbl と btn を配置した gamen ウィンドウを動かす

（実行結果）

```
PS C:\Users\[redacted]\Desktop\py> py py10-2.py
```

PUSH ボタンを押すと



（参考）森 巧尚 著「Python1 年生(第２版)　体験してわかる！会話でまなべる！プログラミングのしくみ」（株式会社 翔泳社）

10－3　値を入力して、結果を返す

tkinter では Entry と呼ばれるテキストボックスから、入力した文字を取得することができる。

また、ラベルやボタンなども含め、pack だけでなく、grid や place などを使っても配置できる。

py10-3.py　10 進数を入力して、ボタンを押すと、2 進数に変換して表示する

```
001 import tkinter as tk
002 def ju2ni():
003   jussin = int(ent10.get())
004   ent2.delete(0, tk.END)
005   nisin=""
006   nisinout = ""
007   while jussin >= 1:
008     nisin= str(int(jussin) % 2) + nisin
009     jussin = int(jussin) // 2
010   ent2.insert(0,nisin)
011 gamen = tk.Tk()
012 gamen.title('10 進から 2 進へ')
013 gamen.geometry("260x100")
014 gamen.configure(bg="skyblue")
015 lbltitle=tk.Label(text="10 進数を入力してください(1 以降の整数)")
016 lbltitle.pack(anchor=tk.W)
017 lbl1 = tk.Label(gamen,text="10 進数")
018 lbl1.place(x=5,y=30)
019 ent10 = tk.Entry(gamen, width=5, font=16, justify="right")
020 ent10.place(x=50,y=30)
021 btn=tk.Button(gamen, text="PUSH",command=ju2ni)
022 btn.place(x=180,y=30)
023 lbl2=tk.Label(gamen, text="2 進数")
024 lbl2.place(x=5,y=60)
025 ent2 = tk.Entry(gamen, width=16, font=16, justify="right")
026 ent2.place(x=50,y=60)
027 tk.mainloop()
```

（解説）

1 行目　tkinter をインポートして tk で使えるようにする

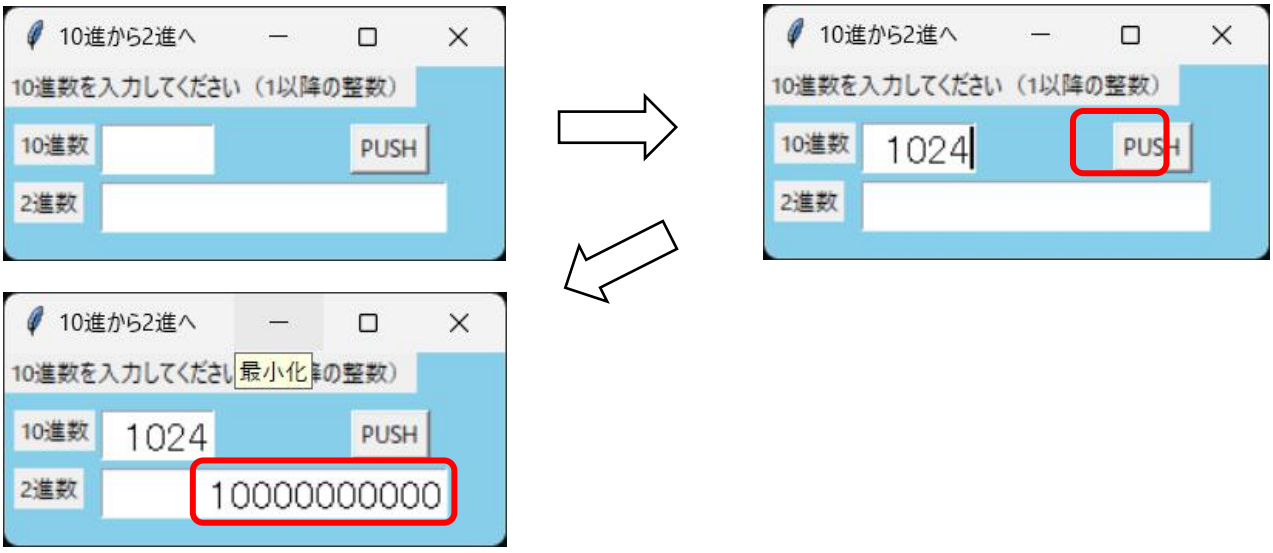
2～10 行目　10 進数から 2 進数に変換する ju2ni 関数

3 行目　ent10 に入力した文字を取得する

４行目　ent2 を初期化する
５、６行目　変数の定義、初期化　nisin,nisinout は文字列で処理する
７～９行目　１０進数から２進数に変換する繰り返し処理（２で割って余りを拾う）
８行目　２で割った余りを、前の文字列の前に入れる
９行目　切り捨て除算を jussin に代入する
１０行目　結果を表示する ent2 に nisin の値を挿入する
１１～２７行目　ウィンドウの定義
１１行目　ウィンドウを gamen とする
１２行目　タイトルバーにタイトル「10 進から 2 進へ」を表示させる
１３行目　幅２６０，高さ１００とする
１４行目　背景色を skyblue とする
１５行目　lbltitle というラベルを作り、「10 進数を入力してください（1 以降の整数）」と文字が表示されるように設定する
１６行目　lbltitle を左揃えで配置する
１７行目　lbl1 というラベルを作り、10 進数と表示させる
１８行目　lbl1 をｘ＝5、ｙ＝30 の位置に配置する
１９行目　ent10 というテキストボックスを gamen に作り、幅を５、文字の大きさを１６、文字を右揃えにする
２０行目　ent10 を x=50 と y=30 に配置する
２１行目　btn というボタンを gamen に作り、PUSH という文字を表示させ、クリックしたら ju2ni 関数が実行されるようにする
２２行目　btn を x=180 y=30 に配置する
２３行目　lbl2 というラベルを作り、2 進数と表示させる
２４行目　lbl2 をｘ＝5、ｙ＝60 の位置に配置する
２５行目　ent2 というテキストボックスを gamen に作り、幅を１６、文字の大きさを１６、文字を右揃えにする
２６行目　ent2 を x=50 と y=60 に配置する
２７行目　ラベルなどを配置した gamen ウィンドウを動かす

（実行結果）

```
PS C:\Users\[redacted]\Desktop\py> py py10-3.py
```



10－4　キャンバスを使う

　tkinter で作ったウィンドウには、入力を受け付けるボタン、文字列を表示するラベルなどのほかに、グラフィックを描画するキャンバスという部品を配置することができる。今回は、キャンバスを使ってプログラムを作成していく。

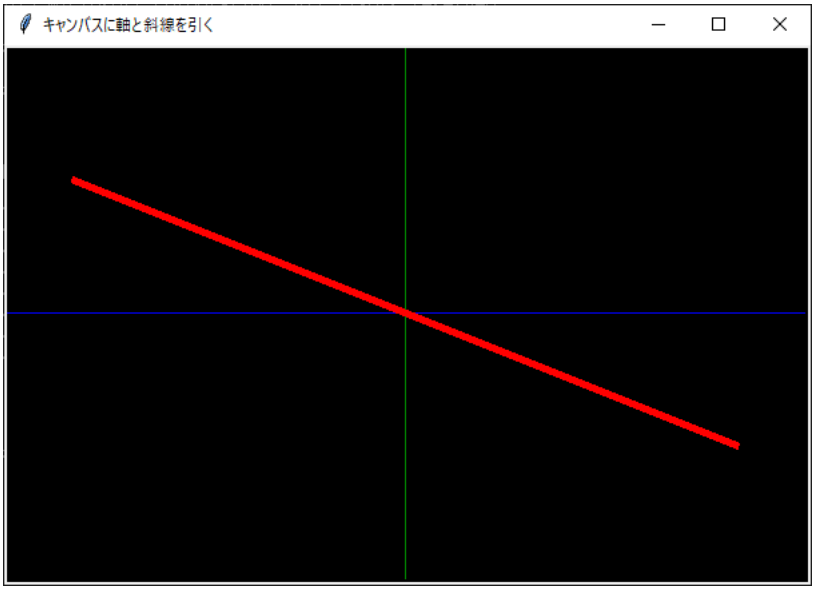
座標について

　コンピュータの画面の原点も、個々のウィンドウの原点も左上が原点になる。横方向がx軸、縦方向がy軸となる。y軸は数学などとは逆で、値が大きくなると下方向に進んでいく。

py10-4.py　キャンバスを配置して、軸と線を引く

```
001 import tkinter as tk
002 gamen = tk.Tk()
003 gamen.title("キャンバスに軸と斜線を引く")
004 cvs = tk.Canvas(width=600, height=400,bg = "black")
005 cvs.create_line(300,0,300,400,fill="green")
006 cvs.create_line(0,200,600,200,fill="blue")
007 cvs.create_line(50,100,550,300,fill="red",width=5)
008 cvs.pack()
009 tk.mainloop()
```

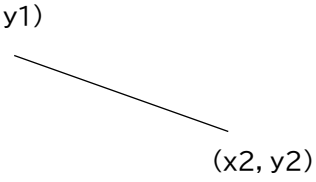
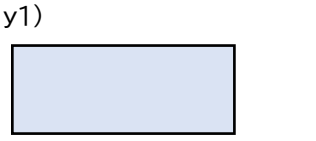
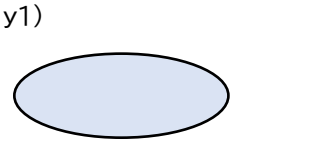
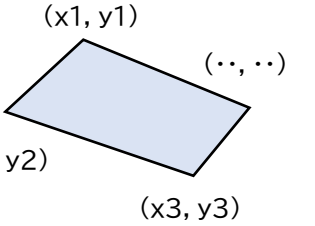
（実行結果）

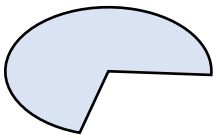


10-5 キャンバスにいろいろな図形を描く

キャンバスに図形を描く命令は、下表のとおりである。

表10－5 図形の描画命令

図形	描画命令	描画イメージ
線	create_line(x1,y1,x2,y2,fill=色,width=線の太さ) ・座標の引数は[x1,y1,x2,y2]のように配列でも指定できる。 ・三点以上をまとめて指定し、それらを線で結べる。 ・三点以上指定して smooth=True という引数を加えると曲線になる	
矩形	create_rectangle(x1,y1,x2,y2,fill=塗る色、outline=周りの色,width=線の太さ)	
楕円	create_oval(x1,y1,x2,y2,fill=塗る色,outline=周りの線の色,width=線の太さ) ・(x1,y1)を左上角、(x2,y2)を右上角とした矩形の中に入る楕円を描く。	
多角形	create_polygon([x1,y1,x2,y2,x3,y3,‥,‥], fill=塗る色,outline=周りの線の色,width=線の太さ) ・複数の点を指定する	

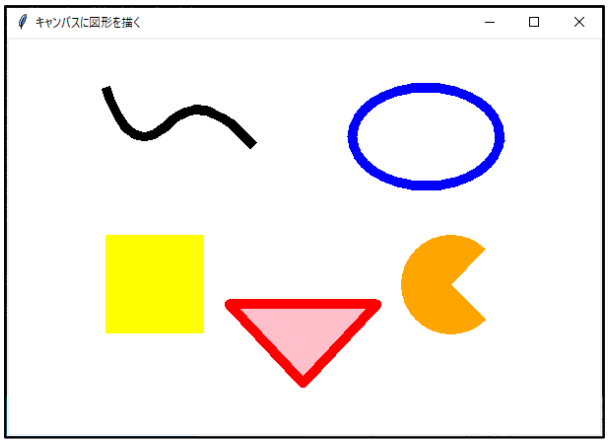
扇形 （円弧）	create_arc(x1,y1,x2,y2, fill=塗る色,outline=周りの線の色,start=開始角,extent=何度開くか，style=tkinter.形状 ・角度は度(degree)の値で指定する。 ・style=は省略可。指定するなら ARC,CHORD,PIESLICE のいずれかを記す。 ・ARC は弧、CHORD は弦の意味。PIESLICE は一部を切り取ったパイの形。	(x1, y1)  (x2, y2)

※引数の fill が塗りつぶしの色、outline が図形の枠線の色、width が線の太さを指定し、塗りつぶしの色を設定しない場合は、線だけで図形が描かれる。

py10-5.py　キャンバスにいろいろな図形を描く

```
001 import tkinter as tk
002 gamen = tk.Tk()
003 gamen.title("キャンバスに図形を描く")
004 cvs = tk.Canvas(width=600, height=400, bg="white")
005 cvs.create_line(100, 50, 125, 125, 200, 50, 250, 110, smooth=True, width=10)
006 cvs.create_rectangle(100, 200, 200, 300, fill="yellow", width=0)
007 cvs.create_oval(350, 50, 500, 150, outline="blue", width=10)
008 cvs.create_polygon(300, 350, 225, 270, 375, 270, fill="pink", outline="red", width=10)
009 cvs.create_arc(400, 200, 500, 300, start=45, extent=270, fill="orange", outline="")
010 cvs.pack()
011 tk.mainloop()
```

（実行結果）



10-6　キャンバスに文字を表示する

キャンバスに文字列を表示するには、下記のように記述する

cvs.create_text(x座標, y 座標, text ＝ 文字列 , font=(フォントの種類,サイズ))

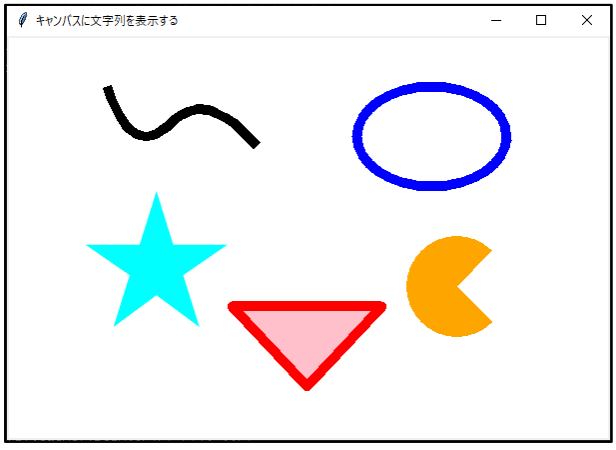
py10-6.py　キャンバスに文字列を表示する

```
001  import tkinter as tk
002  gamen = tk.Tk()
003  gamen.title("キャンバスに文字列を表示する")
004  cvs = tk.Canvas(width=600, height=400, bg="white")
005  cvs.create_line(100, 50, 125, 125, 200, 50, 250, 110, smooth=True, width=10)
006  cvs.create_text(150, 225, text="★", fill="yellow", font=("MSゴシック",120))
007  cvs.create_oval(350, 50, 500, 150, outline="blue", width=10)
008  cvs.create_polygon(300, 350, 225, 270, 375, 270, fill="pink", outline="red", width=10)
009  cvs.create_arc(400, 200, 500, 300, start=45, extent=270, fill="orange", outline="")
010  cvs.pack()
011  tk.mainloop()
```

（解説）

6行目　create_text を使って、フォントは MS ゴシック、サイズは 120 の「★」の文字を表示させている。

（実行結果）



10-7 配列を使って色を扱う

　ゲーム制作で規則正しくブロックを配置するときなどには配列を扱う。ここでは、図形の矩形を連続して配置して、配列で定義された色を配色していく。

py10-7.py　配列で定義した色を配色する

```
001 import tkinter as tk
002 gamen = tk.Tk()
003 gamen.title("配列で色を定義する")
004 cvs = tk.Canvas(width=700, height=100)
005 rainbow=["red","orange","yellow","green","blue","indigo","violet"]
006 for i in range(7):
007     X=i*100
008     cvs.create_rectangle(X,0,X+100,100,fill=rainbow[i],width=0)
009 cvs.pack()
010 tk.mainloop()
```

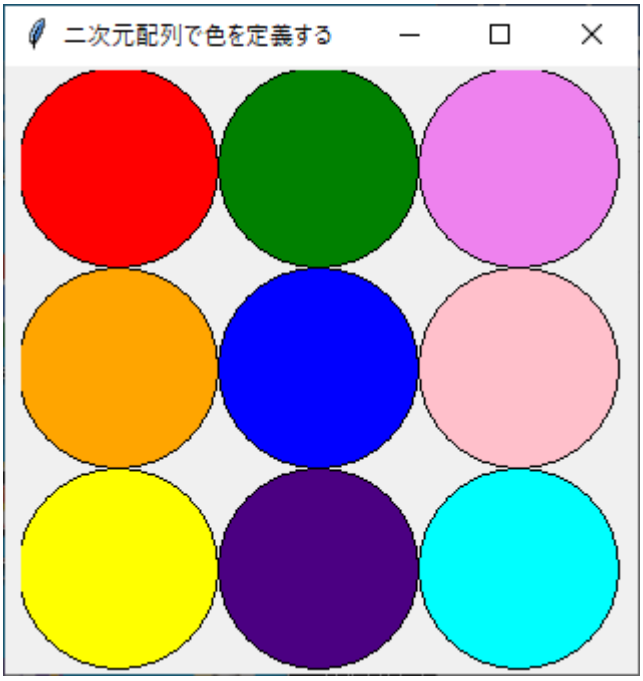
（実行結果）

py10-7-2.py　二次元配列で定義した色を配色する

```
001 import tkinter as tk
002 gamen = tk.Tk()
003 gamen.title("二次元配列で色を定義する")
004 cvs = tk.Canvas(width=300, height=300)
005 color=[
006     ["red","orange","yellow"],
007     ["green","blue","indigo"],
008     ["violet","pink","cyan"]
009 ]
010 for j in range(3):
011     for i in range(3):
012         X=i*100
013         Y=j*100
```

```
014             cvs.create_oval(X,Y,X+100,Y+100,fill=color[i][j])
015 cvs.pack()
016 tk.mainloop()
```

（実行結果）



10-8　画像ファイルを表示する

画像ファイルをプログラムファイルと同じ場所にある pic フォルダに入れ、

py10-8.py　画像ファイルを表示する

```
001 import tkinter as tk
002 gamen = tk.Tk()
003 gamen.title("キャンパスに画像を表示する")
004 cvs = tk.Canvas(width=600, height=400)
005 img = tk.PhotoImage(file='img/img10-8-1.png')
006 cvs.create_image(300,200,image=img)
007 img2 = tk.PhotoImage(file='img/img10-8-2.png')
008 cvs.create_image(300,200,image=img2)
009 cvs.pack()
010 tk.mainloop()
```

(説明)

5行目　PhotoImage で変数 img に背景の画像(img フォルダの img10-8-1.png)を読み込む

6行目　img に読み込んだ画像を(300,200)が画像の中心になるように表示する

7、8行目　5, 6行目と同じ要領で顔の画像 img10-8-2.png ファイルを読み込んで、表示する

(実行結果)

10-9　リアルタイム処理

　ゲームのプログラムは、常に入力を受け付け、画面を書き換えながら描画する。背景がスクロールしたり、ギャラクターが動き続けたりなどの、時間軸に沿って続いていく処理は、リアルタイム処理と呼ぶ。ここでは Python でリアルタイム処理を行う方法を説明する。
　tkinter で

py10-9.py　数をカウントする

```
001 import tkinter as tk
002 n = 0
003 def count():
004     global n
005     n = n + 1
006     cvs.delete("all")
007     cvs.create_text(100, 75, text=n, font=("System", 80))
008     gamen.after(1000, count)
009 gamen = tk.Tk()
010 gamen.title("数を表示する")
011 cvs = tk.Canvas(width=200, height=150)
012 cvs.pack()
013 count()
014 tk.mainloop()
```

（解説）
2行目　グローバル変数nの定義
4行目　グローバル変数nを関数内で変更する
　　グローバル変数は、プログラムが終了するまで保持され、関数内で宣言したローカル変数の値は、それを宣言した関数を呼び出すたびに初期化される。つまり変数nを関数count()内で宣言すると、読み出すたびにnは初期化され、数をカウントすることができなくなる。
5行目　nの値を1増やす
6行目　キャンバスに描いたものを削除する。 こうすることで何もない状態で数字が表示できるとともに、上書きせずにすべてを消しておくことで、処理が重たくなることがなくなる。
7行目　nの値をこの行の設定で表示する。
8行目　指定時間を1000ミリ秒＝1秒として、1秒後に count()を呼び出している。こうすることで、延々と1秒間隔でnを1ずつ増やして表示し続けるようになる。

（実行結果）

10-10　イベント

　ユーザーがキーボードのキーを押したり、マウスを操作したりすることをイベントという。たとえば、ウィンドウをクリックすると、ウィンドウに対してクリックイベントが発生する。キーボードのキーを押すとキーイベントが発生する。

　tkinter で作ったウィンドウでは bind()という命令でイベントを受け取る。bind()は、次のように記述する。

ウィンドウのオブジェクト変数.bind(イベントの種類, 呼び出す関数)

bind()で取得できるイベントは下表のとおりである。

表10－10　bind()で取得できるイベント

イベント	イベントの内容
<Motion>	マウスポインタを動かした
<Button>あるいは<ButtonPress>	マウスボタンを押した
<ButtonRelease>	マウスボタンを離した
<Key>あるいは<KeyPress>	キーを押した
<KeyRelease>	キーを離した

py10-10.py　マウスポインタの座標を表示する

```
001 import tkinter as tk
002 FNT = ("System", 40)
003 def move(e):
004     cvs.delete("all")
005     s = "({}, {})".format(e.x, e.y)
006     cvs.create_text(e.x, e.y, text=s, font=FNT)
007 gamen = tk.Tk()
008 gamen.title("マウスポインタの座標")
009 gamen.bind("<Motion>", move)
```

```
010 cvs = tk.Canvas(width=600, height=400)
011 cvs.pack()
012 tk.mainloop()
```

（解説）
3行目　マウスを動かしたときに呼び出す関数 move()
　4 行目　キャンバスに描いたものを削除
　5行目　format()という命令で、ポインタの座標を文字列にして変数sに代入している。この命令は、format の前の文字列の{}の部分を引数の値に置き換えるものである。
　6行目　変数sに入れた文字列を表示する
9行目　マウスポインタを動かしたら、move()関数を呼び出す

（実行結果）

py10-10-2.py　マウスポインタの座標を表示する２（複数のイベントを使う）

```
001 import tkinter as tk
002 FNT = ("System", 40)
003 def move(e):
004     cvs.delete("all")
005     s = "({}, {})".format(e.x, e.y)
006     cvs.create_text(e.x, e.y, text=s, font=FNT)
007 def move2(e):
008     cvs.delete("all")
009     s = "({}, {})".format(e.x, e.y)
010     cvs.create_text(300, 200, text=s, font=FNT)
011 gamen = tk.Tk()
012 gamen.title("マウスポインタの座標")
013 gamen.bind("<Motion>", move)
014 gamen.bind("<Button>", move2)
```

```
015 cvs = tk.Canvas(width=600, height=400)
016 cvs.pack()
017 tk.mainloop()
```

（解説）

7～10行目　move2 関数:マウスの座標をウィンドウの真ん中で表示する

14行目　クリックしたら、move2 関数を呼び出す

py10-10-3.py　マウスポインタを追いかける

```
001 import tkinter as tk
002 mx = 400
003 my = 300
004 def move(e):
005     global mx, my
006     mx = e.x
007     my = e.y
008 cx = 400
009 cy = 300
010 cr = 30
011 def main():
012     global cx, cy
013     if cy>my: cy -= 10
014     if cy<my: cy += 10
015     if cx>mx: cx -= 10
016     if cx<mx: cx += 10
017     cvs.delete("all")
018     cvs.create_oval(cx-cr, cy-cr, cx+cr, cy+cr, fill="red", outline="pink")
019     gamen.after(30, main)
020 gamen = tk.Tk()
021 gamen.title("ポインタを追い掛ける図形")
022 gamen.resizable(False, False)
023 gamen.bind("<Motion>", move)
024 cvs = tk.Canvas(width=400, height=300, bg="black")
025 cvs.pack()
026 main()
027 tk.mainloop()
```

（解説）

2, 3行目　mx, myをグローバル変数として定義する
6、7行目　mx, myにマウスポインタの座標を代入する
8～10 行目　円の中心のx座標をcx, y座標をcyとし、半径を30とする
11行目　main関数
　12行目　グローバル変数cx、cyを変更できるようにする。
　13～16行目　マウスポインタの位置と円の中心を比較して、円の座標がマウスの座標より小さければ大きくなるように動かし、逆に大きければ小さくなるように動かす。これにより、円がマウスポインタに近づくようになる。
　18行目　変数cx、cy、crを使って、円を描く。
　19行目　30 ミリ秒＝0. 03秒後に11行目 main 関数を呼び出す
22行目　resizable()命令でウィンドウの大きさを変えられないようにする。
23行目　マウスを動かしたら move 関数を呼び出し、マウスポインタの座標を更新する

（実行結果）

　キーの値は bind()命令を使って知ることができる。<Key>イベントを受け取る関数には引数を設け、def pkey(e)と関数を定義した場合、e.keycode や e.keysym が押されたキーの値になる。
　キーコードとシンボルで押されたキーが判定でき、キーコードはキーを表す数値で、Windows と Mac で値が異なるキーがある。シンボルはキー名を表す文字列で、Windows と Mac で共通である。

表10－10－3　キーのシンボル

キー	イベントの内容
0～9	0～9
A～Z	a～z
↑↓←→	Up,Down,Left,Right
[ ]	Space

Enter	Return
Shift(左右)	Shift_L,Shift_R
Esc	Escape

py10-10-4.py　押されたキーの値を表示する

```
001 import tkinter as tk
002 FNT = ("System", 25)
003 def pkey(e):
004     cvs.delete("all")
005     cvs.create_text(150, 70, text="コード="+str(e.keycode), font=FNT)
006     cvs.create_text(150, 130, text="シンボル="+e.keysym, font=FNT)
007 gamen = tk.Tk()
008 gamen.title("キーの値")
009 gamen.bind("<Key>", pkey)
010 cvs = tk.Canvas(width=300, height=200)
011 cvs.pack()
012 tk.mainloop()
```

（解説）

3行目　pkey(e)関数　キーを押したときに呼び出す

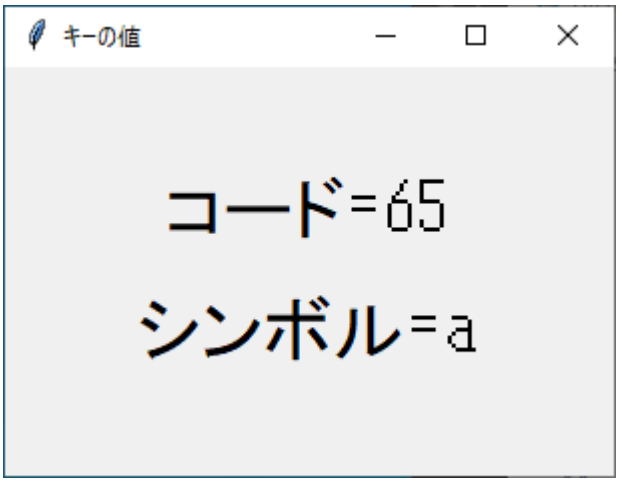
4行目　キャンバスに描いたものを削除

5行目　keycode の値の表示

6行目　keysym の値を表示

11行目　キーが押されたら4行目の pkey 関数を呼び出す

（実行結果）



取得したキーの値を使って、文字列を表示させたり、背景色を変えたりする。ここでは、1～8のキーと、色の英単語と色を対応させて表示する。

表10－10－4　キーと色の対応表

キー	色
1	Red
2	Green
3	Blue
4	Cyan

キー	色
5	Magenta
6	Yellow
7	White
8	Black

py10-10-5.py　押されたキーに合わせて、文字列を表示させ、背景色を変更する

```
001 import tkinter as tk
002 FNT = ("System", 50,"bold")
003 COLOR = ["red","green","blue","cyan","magenta","yellow","white","black"]
004 def pkey(e):
005     k = e.keysym
006     if "1"<=k and k<="8":
007         c = int(k)-1
008         cvs.delete("all")
009         cvs["bg"] = COLOR[c]
010         cvs.create_text(150, 100, text=COLOR[c], fill="gray", font=FNT)
011 gamen = tk.Tk()
012 gamen.title("1～8 キーを押そう")
013 gamen.bind("<Key>", pkey)
014 cvs = tk.Canvas(width=300, height=200)
015 cvs.pack()
016 tk.mainloop()
```

（解説）

3行目　配列で色の英単語を定義

4行目　キーを押したときに呼び出す関数 pkey

5行目　変数kにキーシンボルを代入

6行目　1から8までのキーが押されたら、7～10行目の処理をする

7行目　kの値を整数にして1を引いた値をcに代入（※配列の添え字は0から始まる）

8行目　キャンバスで描いたものを削除

9行目　キャンパスの背景色を変更

10行目　英単語を表示

13行目　キーが押されたとき、pkey 関数を呼び出す

（実行結果）

10-11　ヒットチェック①(円)

　ゲームでよく使うヒットチェック(物体同士が接触しているか判断するアルゴリズム)について説明する。ヒットチェックは当たり判定、衝突判定、接触判定と呼ばれることもある。

　たとえば、円 A(x1,y1)と円 B(x2,y2)があり、下図のように接触していた場合、三平方の定理から

　　d**2＝(x2－x1)**2＋(y2－y1)**2

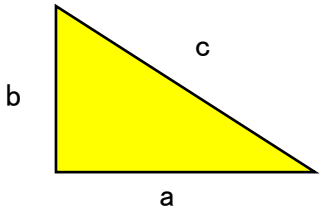
となり、

　　d＝math.sqrt((x2-x1)**2+(y2－y1)**2))

または

　　d＝((x2-x1)**2+(y2－y1)**2))**(0.5)

となる。



$c^2 = a^2 + b^2$

図 10-11-1-1　三平方の定理

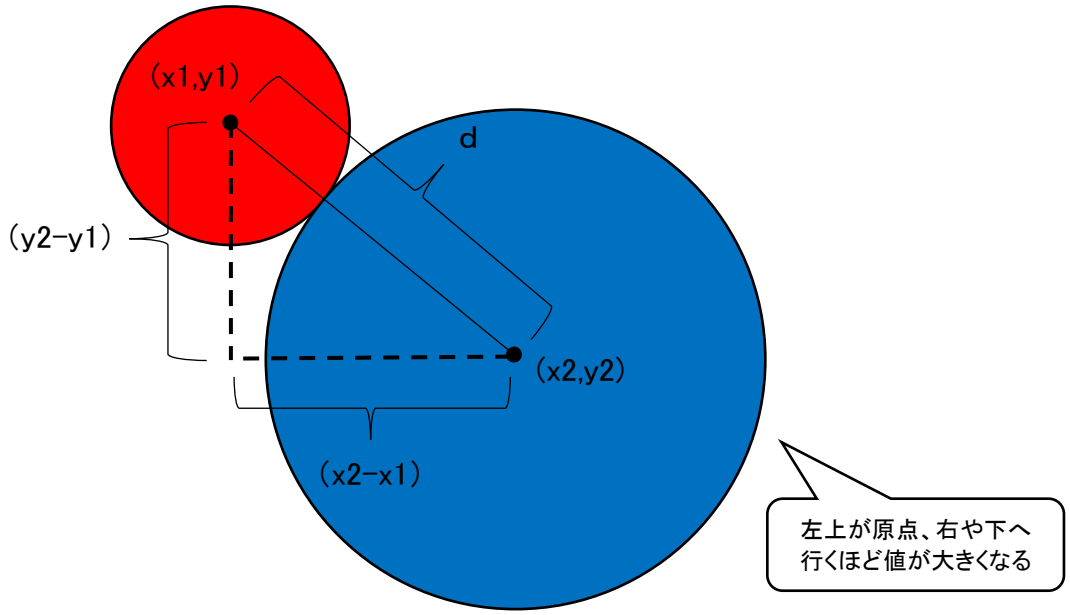
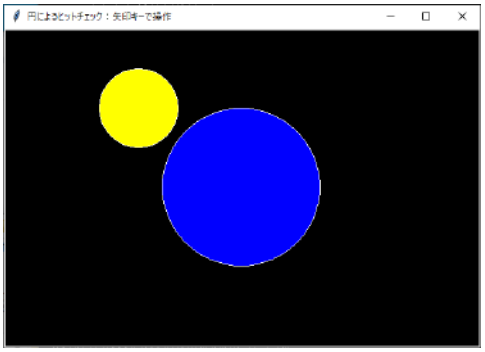


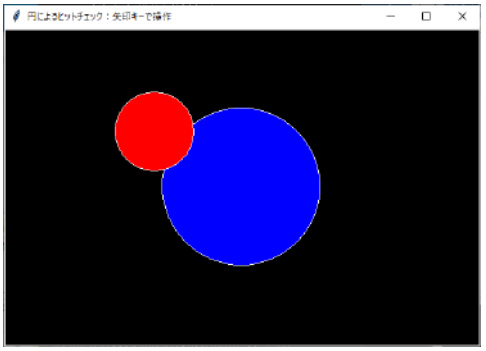
図 10-11-1-2　円の外周が触れ合う例

py10-11-1.py　２つの円が接触したら、捜査している円の塗り色を変える

```
001 import tkinter as tk
002 import math
003 x1 = 50
004 y1 = 50
005 r1 = 50
006 x2 = 300
007 y2 = 200
008 r2 = 100
009 def pkey(e):
010     global x1, y1
011     if e.keysym=="Up": y1 -= 10
012     if e.keysym=="Down": y1 += 10
013     if e.keysym=="Left": x1 -= 10
014     if e.keysym=="Right": x1 += 10
015     d = math.sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))
016     col = "yellow"
017     if d<=r1+r2: col = "red"
018     cvs.delete("RED_CIRCLE")
019     cvs.create_oval(x1-r1, y1-
020 r1, x1+r1, y1+r1, fill=col, outline="white", tag="RED_CIRCLE")
021 gamen = tk.Tk()
022 gamen.title("円によるヒットチェック：矢印キーで操作")
023 gamen.bind("<Key>", pkey)
024 cvs = tk.Canvas(width=600, height=400, bg="black")
025 cvs.pack()
026 cvs.create_oval(x1-r1, y1-
027 r1, x1+r1, y1+r1, fill="yellow", outline="white", tag="RED_CIRCLE")
028 cvs.create_oval(x2-r2, y2-r2, x2+r2, y2+r2, fill="blue", outline="white")
029 tk.mainloop()
```

（実行結果）


円によるヒットチェック：矢印キーで操作


円によるヒットチェック：矢印キーで操作

矩形によるヒットチェックをする場合は、下図のように矩形A(幅w1, 高さh1)、矩形B(幅w2, 高さh2)として、矩形Aの中心座標を(x_1, y_1)、矩形Bの中心座標を(x_2, y_2)とした場合、x_2-x_1の絶対値が$w_1/2+w_2/2$ よりも小さくて、かつ、y_2-y_1の絶対値が$h_1/2+h_2/2$ よりも小さい場合に矩形が重なる。絶対値は abs()を使って求められるので、

abs(x2-x1)<=(w1+w2)/2　and　abs(y2-y1)<=(h1+h2)/2

と表現できる。

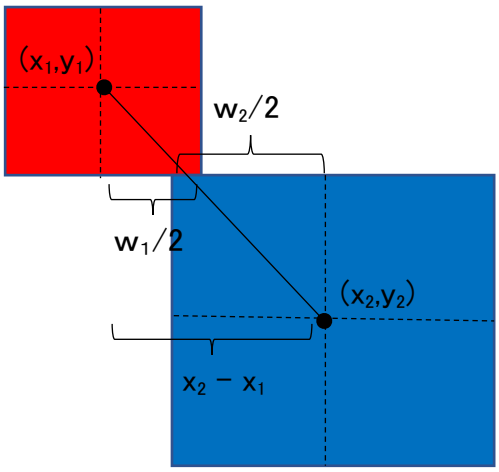


図 10-11-2-1　x座標の差が小さくなる場合

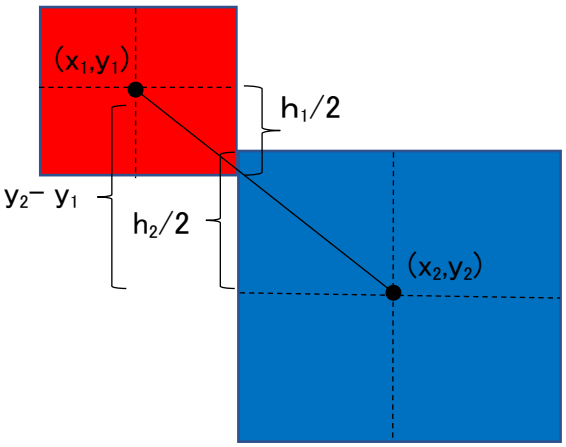


図 10-11-2-2　y座標の差が小さくなる場合

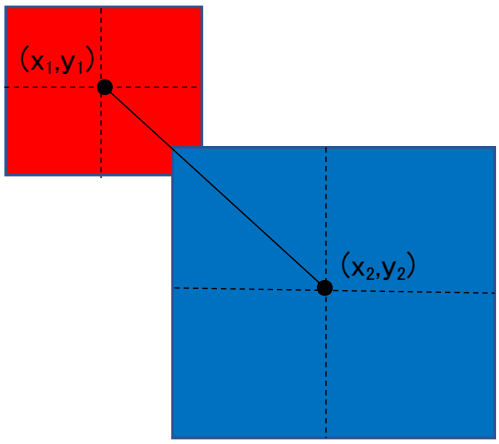


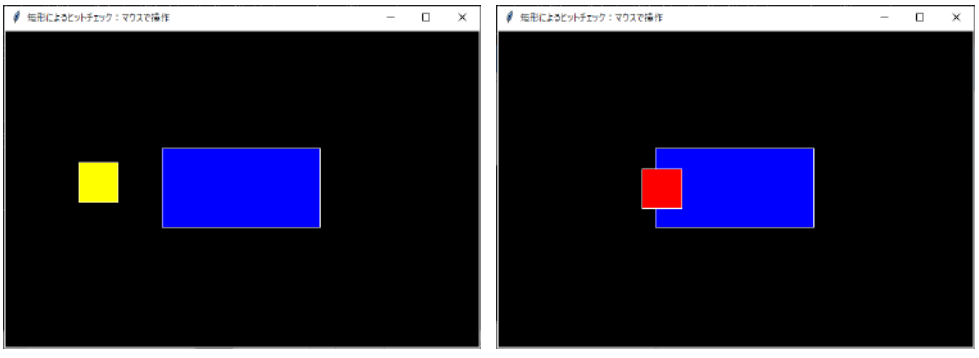
図 10-11-2-3　両方小さくなる場合

py10-11-2.py　２つの矩形が接触したら、操作している矩形の塗り色を変える。

```
001 import tkinter as tk
002 import math
003 x1 = 100
004 y1 = 100
```

```
005 w1 = 50
006 h1 = 50
007 x2 = 300
008 y2 = 200
009 w2 = 200
010 h2 = 100
011 def move(e):
012     global x1, y1
013     x1 = e.x
014     y1 = e.y
015     col = "yellow"
016     if abs(x2-x1)<=(w1+w2)/2 and abs(y2-y1)<=(h1+h2)/2:
017         col = "red"
018     cvs.delete("RED_RECT")
019     cvs.create_rectangle(x1-w1/2, y1-h1/2, x1+w1/2, y1+h1/2, fill=col, outline="white", tag="RED_RECT")
020 gamen = tk.Tk()
021 gamen.title("矩形によるヒットチェック：マウスで操作")
022 gamen.bind("<Motion>", move)
023 cvs = tk.Canvas(width=600, height=400, bg="black")
024 cvs.pack()
025 cvs.create_rectangle(x1-w1/2, y1-
026 h1/2, x1+w1/2, y1+h1/2, fill="red", outline="white", tag="RED_RECT")
027 cvs.create_rectangle(x2-w2/2, y2-h2/2, x2+w2/2, y2+h2/2, fill="blue", outline="white")
028 tk.mainloop()
```

（実行結果）



（参考書籍） 廣瀬豪「Python ではじめるゲーム制作超入門」（インプレス）

10-12 アニメーションをさせる

最終的にモグラたたきゲームを作る準備として、モグラをアニメーションをさせる仕組みを作る。また、メインループの中にシーンを設定して、シーンを切り替えさせる。

py10-12.py モグラの顔を穴から出したり引っ込めたりする

```
001 import tkinter as tk
002 scene = "タイトル"
003 mogu_a = 0
004 def pkey(e):
005     global scene
006     if e.keysym=="space":
007         scene = "ゲーム"
008     if e.keysym=="Return":
009         scene = "タイトル"
010 def main():
011     global mogu_a
012     cvs.delete("all")
013     cvs.create_image(300, 200, image=bg)
014     if scene=="タイトル":
015         cvs.create_image(300, 200, image=ilst)
016       cvs.create_text(300, 80, text="もぐらたたきゲーム", font=("System",50), fill="Black")
017         cvs.create_text(300, 300, text="press [SPACE] key",
    font=("System",30), fill="Black")
018     if scene=="ゲーム":
019         mogu_a = mogu_a + 1
020         cvs.create_image(300, 200, image=mogu[mogu_a%7])
021     gamen.after(100, main)
022 gamen = tk.Tk()
023 gamen.bind("<Key>", pkey)
024 cvs = tk.Canvas(width=600, height=400)
025 cvs.pack()
026 ilst = tk.PhotoImage(file="img/opillust.png")
027 bg = tk.PhotoImage(file="img/bg.png")
028 mogu = [
029     tk.PhotoImage(file="img/mpic0.png"),
030     tk.PhotoImage(file="img/mpic1.png"),
```

```
031     tk.PhotoImage(file="img/mpic2.png"),
032     tk.PhotoImage(file="img/mpic3.png"),
033     tk.PhotoImage(file="img/mpic2.png"),
034     tk.PhotoImage(file="img/mpic1.png"),
035     tk.PhotoImage(file="img/mpic0.png")
036 ]
037 main()
038 tk.mainloop()
039
```

（解説）

4～9行目　キー入力で帰って来るキーシンボルで条件式造り、シーンを切り替える

10～32行目　メインループ

　　14～17行目　タイトルのシーン：タイトルの文字やイラストを表示させる

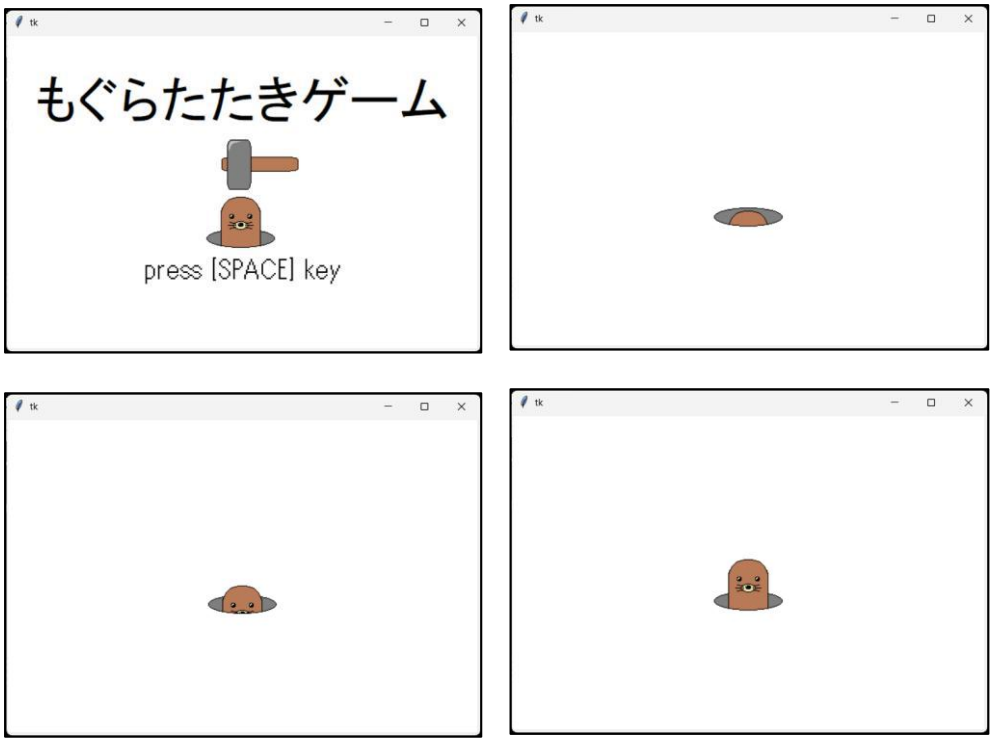
　　18～20行目　ゲームのシーン：ここではモグラをアニメーションさせるだけ

26行目　最初のページに表示させるイラストを ilst に設定

27行目　背景の画像をbgに設定

28～36行目　モグラの顔の画像を配列 mogu に入れる

（完成例）



10-13　もぐらたたきゲームを作る

　いままで勉強したことを使ってゲームを作りなさい。

（完成例）