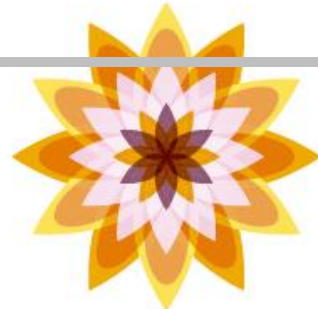
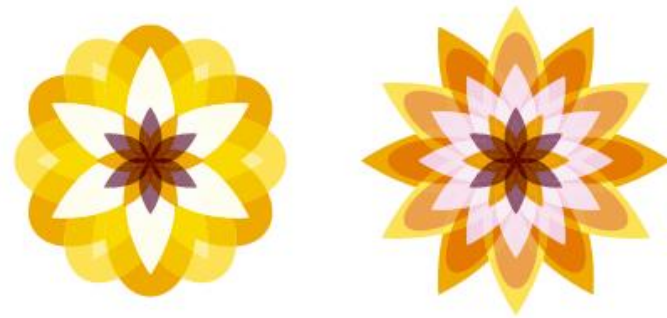


*Chapter 08*

# 블록 낙하 게임 만들기



# 1. 게임 사양 고려하기

- 본격적으로 게임을 개발합니다.
- 가장 먼저 게임 규칙, 화면 구성, 처리 흐름과 같은 사양을 고려합니다.

## ■ 게임 규칙

- 이 게임은 낙하하는 블록을 고양이 캐릭터로 디자인했습니다.
- ① 화면을 클릭한 위치에 고양이(블록)를 1 개 배치합니다. 배치하는 고양이는 화면 오른쪽 위에 표시되며, 랜덤으로 바뀝니다.
- ② 고양이를 배치하면 화면 위에서 여러 고양이가 떨어집니다.
- ③ 떨어진 고양이는 바닥 위치에 아무것도 없으면 화면 아래쪽부터 위쪽으로 쌓입니다.
- ④ 가로, 세로, 대각선 중 한 방향에 같은 색의 고양이가 3개 이상 모이면 없앨 수 있습니다.
- ⑤ 한 줄이라도 맨 위의 화면에 닿으면 게임 끝입니다.

# 1. 게임 사양 고려하기

## ■ 화면 구성

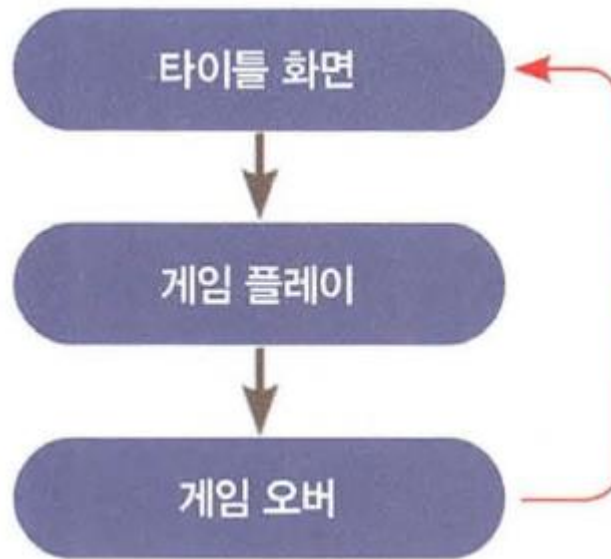
- 실제 게임 개발에서는 스크래치 (scratch) 등으로 화면을 스케치하며 게임 화면을 상상해보지만 예제에서는 완성된 화면을 먼저 확인하겠습니다.



# 1. 게임 사양 고려하기

## ■ 처리 흐름

- 다음과 같은 흐름으로 처리를 수행합니다.



- 타이틀 화면에서 게임 난이도를 Easy, Normal, Hard 중 선택할 수 있습니다.
- 스테이지를 클리어하는 것이 아니라 최고 점수 갱신을 목표로 플레이하는 게임입니다.

## ■ 개발 순서

- 어느 정도 규모가 있는 게임을 개발할 때는 위와 같이 게임 사양을 먼저 고려합니다.
- 사양을 고려함으로써 게임을 완성시키기 위해 어떤 처리를 해야 할지 파악할 수 있습니다.
- 컴퓨터나 수학에서 문제를 해결하는 순서를 구체화한 것을 알고리즘(algorithm)이라고 부릅니다.
- 블록 낙하 퍼즐에서는 크게 2가지 알고리즘이 필요합니다.
- '블록 낙하 알고리즘'과 '블록 연결 판정 알고리즘'입니다
- 다양한 처리를 차근차근 이해할 수 있도록 전체 프로그램을 부분으로 나누어서 프로그램을 만들어 갑니다.

# 1. 게임 사양 고려하기

## ■ 이미지 리소스에 관해

- 다양한 이미지 파일을 사용합니다.
- 깃헙 페이지에서 이미지를 다운로드한 다음 프로그램과 같은 폴더에 넣으면 됩니다.



## 2. 마우스 입력 조합하기

- 본격적인 게임 프로그래밍을 시작합니다.
- 이 게임은 마우스로 조작하므로 먼저 마우스의 움직임과 버튼 클릭을 판정하는 프로그램을 입력해 마우스 동작을 숫자 값으로 얻는 방법을 학습합니다.
- 파이썬에서의 마우스 입력
  - 앞 장에서 키 입력(키 이벤트)을 학습했습니다.
  - 마우스도 마찬가지로 `bind()` 명령과 이벤트를 받아 처리하는 함수를 정의해서 입력받습니다.
  - 구체적으로는 마우스 포인터 좌표를 대입하는 변수, 버튼 클릭 여부를 판정하는 변수를 준비합니다.
  - 마우스 이벤트 발생 시 실행할 함수를 정의하고 이들 변수에 이벤트에서 얻은 값을 대입합니다.

## 2. 마우스 입력 조합하기

### ■ 파이썬에서의 마우스 입력

- 마우스 입력을 처리하는 프로그램을 만들어 보겠습니다.
- 예제 ex0802\_1.py (※ 마우스 이벤트 처리 부분을 굵게 표시)

```
1 import tkinter
```

```
2
```

```
3 mouse_x = 0
```

```
4 mouse_y = 0
```

```
5 mouse_c = 0
```

```
6
```

```
7 def mouse_move(e) :
```

```
8     global mouse_x, mouse_y
```

```
9     mouse_x = e.x
```

```
10    mouse_y = e.y
```

```
11
```

tkinter 모듈 импорт

마우스 포인터의 X 좌표

마우스 포인터의 Y 좌표

마우스 포인터 클릭 여부 변수

마우스 포인터 이동 시 실행할 함수

mouse\_x, mouse\_y를 전역 변수로 선언

mouse\_x에 마우스 포인터의 X좌표 대입

mouse\_y에 마우스 포인터의 Y좌표 대입



## 2. 마우스 입력 조합하기

### ■ 파이썬에서의 마우스 입력

- 예제 ex0802\_1.py (※ 마우스 이벤트 처리 부분을 굵게 표시)

12	<code>def mouse_press(e) :</code>	마우스 버튼 클릭 시 실행할 함수
13	<code>    global mouse_c</code>	mouse_c를 전역 변수로 선언
14	<code>    mouse_c = 1</code>	mouse_c에 1 대입
15		
16	<code>def mouse_release(e) :</code>	마우스 버튼 클릭 후 해제 시 실행할 함수
17	<code>    global mouse_c</code>	mouse_c를 전역 변수로 선언
18	<code>    mouse_c = 0</code>	mouse_c에 0 대입
19		
20	<code>def game_main( ) :</code>	실시간 처리 수행 함수
21	<code>    fnt = ("Times New Roman", 30)</code>	폰트 지정 변수
22	<code>    txt = "mouse ({}, {})".format(mouse_x, mouse_y, mouse_c)</code>	표시할 문자열(마우스용 변수값)

## 2. 마우스 입력 조합하기

### ■ 파이썬에서의 마우스 입력

- 예제 ex0802\_1.py (※ 마우스 이벤트 처리 부분을 굵게 표시)

```
23     cvs.delete("TEST")
24     cvs.create_text(456, 384, text=txt,
25                     fill="black", font=fnt, tag="TEST")
26     root.after(100, game_main)
27
28 root = tkinter.Tk( )
29 root.title("마우스 입력")
30 root.resizable(False, False)
31 root.bind("<Motion>", mouse_move)
root.bind("<ButtonPress>", mouse_press)
```

문자열 삭제

캔버스에 문자열 표시

0.1초 후 함수 재실행

윈도우 객체 생성

윈도우 제목 지정

윈도우 크기 변경 불가 설정

**마우스 포인터 이동시 실행할  
함수 지정**

**마우스 버튼 클릭시 실행할 함  
수 지정**

## 2. 마우스 입력 조합하기

### ■ 파이썬에서의 마우스 입력

- 예제 ex0802\_1.py (※ 마우스 이벤트 처리 부분을 굵게 표시)

```
32 root.bind("<ButtonRelease>",  
mouse_release)
```

마우스 버튼 클릭 후 해제 시  
실행할 함수 지정

```
33 cvs = tkinter.Canvas(root, width=912,  
height=768)
```

캔버스 컴포넌트 생성

```
34 cvs.pack( )
```

캔버스 컴포넌트 배치

```
35 game_main( )
```

실시간 처리 수행 함수 호출

```
36 root.mainloop( )
```

윈도우 표시


- 이 프로그램을 실행하면 마우스 포인터 좌표가 표시되고 버튼을 클릭하면 가장 오른쪽 숫자 값이 0에서 1로 바뀝니다.
- 마우스 포인터를 움직이거나 버튼을 클릭하면서 값 변화를 확인합니다.

## 2. 마우스 입력 조합하기

### ■ 파이썬에서의 마우스 입력

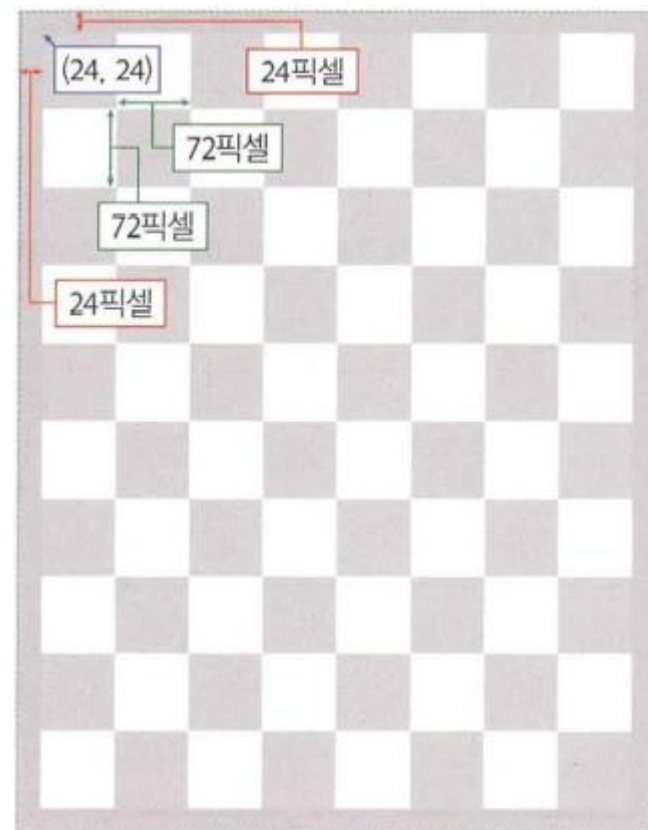
- 예제 ex0802\_1.py (※ 마우스 이벤트 처리 부분을 굵게 표시)
- 7~18번 행에서 마우스 포인터를 움직일 때 실행할 함수, 마우스 버튼을 클릭할 때 실행할 함수, 마우스 버튼을 클릭한 후 떼었을 때 실행하는 함수를 각각 정의합니다.
- 포인터의 좌표는 함수의 인수로 받는 이벤트 변수(이 프로그램에서는 'e')에 9, 10번 행과 같이 '.x'와 '.y'를 붙여서 얻습니다.
- 버튼을 클릭했을 때와 클릭한 후 떼었을 때는 mouse\_c에 1과 0을 대입함으로써 이 변수가 1이라면 버튼을 클릭한 것으로 판정할 수 있게 합니다.
- mouse\_c와 같은 변수 사용 방법을 플래그(flag)라고 하며, 클릭했을 때 값을 1로 하는 것을 '플래그를 올린다', 클릭 후 떼었을 때 그 값을 0으로 하는 것을 '플래그를 내린다'라고 표현합니다.
- 22번 행의 format( ) 명령은 문자열 내의 {}를 변수값으로 대체합니다.
- {}의 수(변수 개수)는 임의로 입력할 수 있습니다.

```
txt = "mouse({}, {}){}".format(mouse_x, mouse_y, mouse_c)
```



### 3. 게임용 커서 표시하기

- 다음으로 얻은 마우스 좌표값을 사용해 게임용 커서를 조작할 수 있도록 하겠습니다.
- 게임 화면 사이즈 설정하기
  - 고양이(블록)를 낙하할 영역의 크기를 다음과 같이 설정합니다.
  - 고양이를 배열할 영역은 가로 8칸, 세로 10칸으로 합니다.



### 3. 게임용 커서 표시하기

#### ■ 게임 화면 사이즈 설정하기

- 마우스 포인터 좌표에서 커서의 위치를 계산하는 프로그램을 확인하겠습니다.
- 마우스 버튼 판정은 여기서는 불필요하므로 생략합니다.
- 예제 ex0803\_1.py (※ 커서 위치 계산 및 표시 부분 굵게 표시)

```
1 import tkinter
```

```
2
```

```
3 cursor_x = 0
```

```
4 cursor_y = 0
```

```
5 mouse_x = 0
```

```
6 mouse_y = 0
```

```
7
```

```
8 def mouse_move(e) :
```

```
9     global mouse_x, mouse_y
```

```
10    mouse_x = e.x
```

```
11    mouse_y = e.y
```

tkinter 모듈 импорт

커서의 X 좌표

커서의 Y 좌표

마우스 포인터의 X 좌표

마우스 포인터의 Y 좌표

마우스 포인터 이동 시 실행할 함수

mouse\_x, mouse\_y를 전역 변수로 선언

mouse\_x에 마우스 포인터의 X좌표 대입

mouse\_y에 마우스 포인터의 Y좌표 대입

### 3. 게임용 커서 표시하기

#### ■ 게임 화면 사이즈 설정하기

- 예제 ex0803\_1.py (※ 커서 위치 계산 및 표시 부분 굵게 표시)

```
13 def game_main( ) :  
14     global cursor_x, cursor_y  
  
15     if 24 <= mouse_x and mouse_x  
       < 24+72*8 and 24 <= mouse_y and  
       mouse_y < 24+72*10 :  
16         cursor_x = int((mouse_x - 24)  
                           / 72)  
17         cursor_y = int((mouse_y -24) /  
                           72)  
18         cvs.delete("CURSOR")  
19         cvs.create_image(cursor_x*72+60,  
                           cursor_y*72+60, image=cursor,  
                           tag="CURSOR")  
20         root.after(100, game_main)
```

실시간 처리 수행 함수

cursor\_x, cursor\_y를 전역 변수로 선언

마우스 포인터 좌표가 게임 영역 내에 있으면

포인터의 X 좌표에서 커서의 X 좌표 계산

포인터의 Y 좌표에서 커서의 Y 좌표 계산

커서 삭제

새로운 위치에 커서 표시

0.1초 후 함수 재실행

### 3. 게임용 커서 표시하기

#### ■ 게임 화면 사이즈 설정하기

- 예제 ex0803\_1.py (※ 커서 위치 계산 및 표시 부분 굵게 표시)

21		
22	<code>root = tkinter.Tk( )</code>	윈도우 객체 생성
23	<code>root.title("커서 표시")</code>	윈도우 제목 지정
24	<code>root.resizable(False, False)</code>	윈도우 크기 변경 불가 설정
25	<code>root.bind("&lt;Motion&gt;", mouse_move)</code>	마우스 포인터 이동 시 실행할 함수 지정
26	<code>cvs = tkinter.Canvas(root, width=912, height=768)</code>	캔버스 컴포넌트 생성
27	<code>cvs.pack( )</code>	캔버스 컴포넌트 배치
28		



### 3. 게임용 커서 표시하기

#### ■ 게임 화면 사이즈 설정하기

- 예제 ex0803\_1.py (※ 커서 위치 계산 및 표시 부분 굵게 표시)

```
29 bg = tkinter.PhotoImage(file="neko_bg.png")
30 cursor =
    tkinter.PhotoImage(file="neko_cursor.png")
31 cvs.create_image(456, 384, image=bg)
32 game_main( )
33 root.mainloop( )
```

배경 이미지 로딩

커서 이미지 로딩

캔버스에 배경 그리기

실시간 처리 수행 함수 호출

윈도우 표시

### 3. 게임용 커서 표시하기

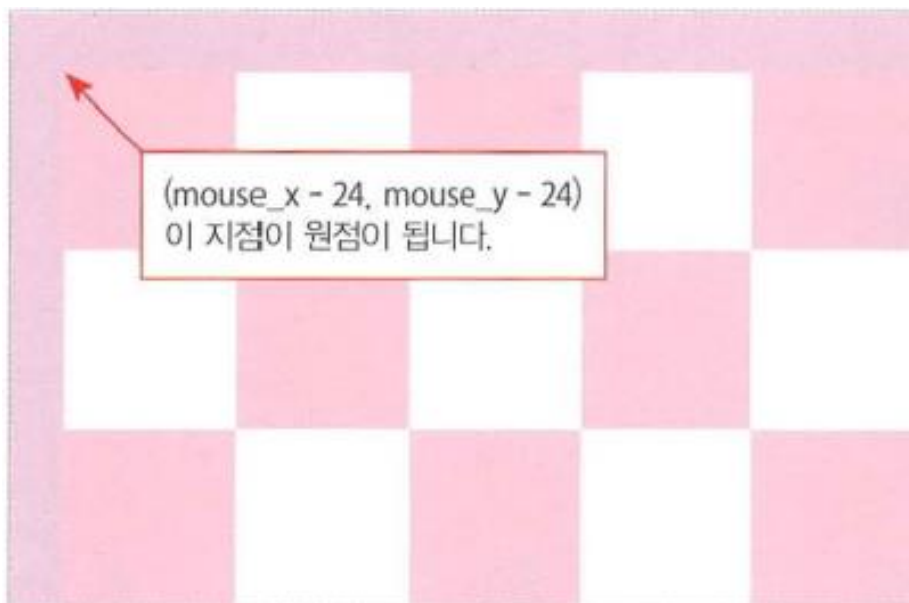
#### ■ 게임 화면 사이즈 설정하기

- 16~17번 행에서 마우스 포인터 좌표값을 사용해 게임 보드에서 커서의 위치를 계산합니다.

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

```
cursor_y = int((mouse_y - 24) / 72)
```

- 이 계산식에서 사용하는 숫자값이 어떤 값인지를 확인해 보겠습니다.
- 24는 여백 부분의 픽셀 수, 72는 1칸의 크기입니다.
- $\text{mouse\_x} - 24$ ,  $\text{mouse\_y} - 24$ 는 다음 그림에서 칸의 왼쪽 모서리를 원점으로 하기 위한 뉘셈입니다.



### 3. 게임용 커서 표시하기

#### ■ 게임 화면 사이즈 설정하기

- '(mouse\_x - 24)를 72로 나눈 몫이 왼쪽으로부터 몇 번째 칸인가?', '(mouse\_y - 24)를 72로 나눈 몫이 위로부터 몇 번째 칸인가?'라는 값이 됩니다.
- 나눗셈한 값의 소수점 이하를 int( ) 명령으로 버리고 몫을 구합니다.
- 커서를 그리는 위치 (캔버스 상 좌표) 지정 부분도 확인합니다.
- 19번 행과 같이 'cursor\_x \* 72 + 60, cursor\_y \* 72 + 60'이라는 식으로 커서의 위치로부터 캔버스 상의 좌표를 구합니다.
- 이 계산식의 의미는 1칸의 크기가 72픽셀이므로 72를 더하고, create\_image( ) 명령의 좌표는 이미지의 중심이므로 여백의 24픽셀과 위치 절반 크기인 36픽셀을 합한 60픽셀을 더합니다.

## 4. 위치 데이터 관리하기

- 이번 절에서는 격자(게임판)에 늘어놓을 고양이(블록)를 리스트로 관리하는 프로그램을 작성해보겠습니다.

- 2차원 리스트로 관리하기

- 격자는 가로 8칸, 세로 10칸으로 되어 있습니다.
- 칸에는 아무것도 들어있지 않거나, 6종류의 고양이 중 하나가 들어갑니다.
- 이런 게임 화면은 앞 장에서 학습한 것처럼 2차원 리스트를 사용해 관리합니다.
- 이번 프로그램에서는 리스트의 엘리먼트 값(데이터)을 다음과 같이 정의합니다.
- 리스트 이름은 `neko`이며, 2차원 리스트 `neko[y][x]`에 데이터를 넣고 꺼냅니다.

neko[y][x]	0	1	2	3	4	5	6	7
	표시 없음							

- ❖ 7번 발자국 이미지는 떨어진 고양이를 지웁니다.

## 4. 위치 데이터 관리하기

### ■ 2차원 리스트로 관리하기

- 8 x 10칸을 2차원 리스트로 정의하고 고양이 이미지를 표시하는 프로그램을 작성하겠습니다.
- 이미지는 여러 장이므로 1차원 리스트로 관리합니다.
- 동작 확인 후, 리스트에 이미지를 로딩하는 방법을 설명하겠습니다.
- 예제 ex0804\_1.py (※ 2차원 리스트와 고양이 표시 부분은 굵게 표시)

```
1 import tkinter
```

```
2
```

```
3 neko = [
```

```
4     [1, 0, 0, 0, 0, 0, 7, 7],
```

```
5     [0, 2, 0, 0, 0, 0, 7, 7],
```

```
6     [0, 0, 3, 0, 0, 0, 0, 0],
```

```
7     [0, 0, 0, 4, 0, 0, 0, 0],
```

```
8     [0, 0, 0, 0, 5, 0, 0, 0],
```

```
9     [0, 0, 0, 0, 0, 6, 0, 0],
```

tkinter 모듈 임포트

위치를 관리할 2차원 리스트

## 4. 위치 데이터 관리하기

### ■ 2차원 리스트로 관리하기

- 예제 ex0804\_1.py (※ 2차원 리스트와 고양이 표시 부분은 굵게 표시)

```
10     [0, 0, 0, 0, 0, 0, 0, 0],
11     [0, 0, 0, 0, 0, 0, 0, 0],
12     [0, 0, 0, 0, 0, 0, 0, 0],
13     [0, 0, 1, 2, 3, 4, 5, 6]
14 ]
15
16 def draw_neko( ) :
17     for y in range(10) :
18         for x in range(8) :
19             if neko[y][x] > 0:
20                 cvs.create_image(x*72+60,
21                                 y*72+60, image=img_neko[neko[y][x]])
```

고양이 표시 함수

반복: y는 0부터 9까지 1씩 증가

반복: x는 0부터 7까지 1씩 증가

리스트 엘리먼트 값이 0보다 크면

고양이 이미지 표시

## 4. 위치 데이터 관리하기

### ■ 2차원 리스트로 관리하기

- 예제 ex0804\_1.py (※ 2차원 리스트와 고양이 표시 부분은 굵게 표시)

```
22 root = tkinter.Tk( )
23 root.title("2차원 리스트로 위치 관리함")
24 root.resizable(False, False)
25 cvs = tkinter.Canvas(root, width=912,
26 height=768)
27
28 bg =
29     tkinter.PhotoImage(file="neko_bg.png")
30 img_neko = [
31     None,
32     tkinter.PhotoImage(file="neko1.png"),
33     tkinter.PhotoImage(file="neko2.png"),
```

윈도우 객체 생성

윈도우 제목 지정

윈도우 크기 변경 불가 설정

캔버스 컴포넌트 생성

캔버스 컴포넌트 배치

배경 이미지 로딩

리스트로 여러 고양이 이미지 관리  
img\_neko[0]은 아무것도 없는 값

## 4. 위치 데이터 관리하기

### ■ 2차원 리스트로 관리하기

- 예제 ex0804\_1.py (※ 2차원 리스트와 고양이 표시 부분은 굵게 표시)

```
33     tkinter.PhotoImage(file="neko3.png"),
34     tkinter.PhotoImage(file="neko4.png"),
35     tkinter.PhotoImage(file="neko5.png"),
36     tkinter.PhotoImage(file="neko6.png"),
37     tkinter.PhotoImage
38     (file="neko_niku.png")
39 ]
40 cvs.create_image(456, 384, image=bg)
41 draw_neko( )
42 root.mainloop( )
```

캔버스에 배경 그리기  
고양이 표시 함수 호출  
윈도우 표시



## 4. 위치 데이터 관리하기

### ■ 2차원 리스트로 관리하기

- 예제 ex0804\_1.py (※ 2차원 리스트와 고양이 표시 부분은 굵게 표시)
- 이 프로그램을 실행하면 2차원 리스트로 정의한 대로 고양이와 발자국 이미지가 표시됩니다.



## 4. 위치 데이터 관리하기

### ■ 2차원 리스트로 관리하기

- 예제 ex0804\_1.py (※ 2차원 리스트와 고양이 표시 부분은 굵게 표시)
- 3~14번 행이 칸 위의 고양이를 관리하는 1차원 리스트입니다.
- 리스트 값과 그림의 이미지 번호를 대조해 보기 바랍니다.
- 16~20번 행에 정의한 draw\_neko( ) 함수로 고양이와 발자국 이미지를 표시합니다.
- 20번 행 create\_image( ) 명령의 인수 좌표는 'x \* 72 + 60, y \* 72 + 60'이며, 칸의 중심 위치를 지정합니다.
- 29~38번 행이 이미지를 로딩하는 리스트로 0번째 엘리먼트를 None으로 지정합니다.
- 파이썬에서는 '아무것도 존재하지 않는 상태'를 None으로 표시합니다.
- `neko[y][x]` 값이 0인 곳에는 아무것도 표시하지 않는 이미지가 없으므로 이 프로그램에서는 '이미지 불필요'라는 의미로 None을 사용합니다.

## 5. 블록 낙하 알고리즘

- 이 절에서는 고양이(블록) 낙하 처리를 추가합니다.
- 리스트 값 확인하기
  - 고양이가 존재하는 칸의 한 칸 아래 위치가 비어 있는 경우 고양이를 빈 칸으로 이동시키면 한 칸 낙하한 것이 됩니다.
  - 이 처리를 모든 칸에 적용하면 화면 전체의 고양이를 떨어뜨릴 수 있습니다.
  - 한 칸씩 떨어뜨리기 위해서는 아래쪽 칸부터 위쪽 칸 방향으로 확인해 나가야 합니다.
  - 일단 프로그램의 동작을 확인한 후 그림과 함께 설명하겠습니다.

## 5. 블록 낙하 알고리즘

### ■ 리스트 값 확인하기

- 예제 ex0805\_1.py (※ 고양이 낙하 관련 처리는 굵게 표시)

```
1 import tkinter
```

tkinter 모듈 импорт

```
2
```

```
3 neko = [
```

위치를 관리할 2차원 리스트

```
4     [1, 0, 0, 0, 0, 0, 1, 2],
```

```
5     [0, 2, 0, 0, 0, 0, 3, 4],
```

```
6     [0, 0, 3, 0, 0, 0, 0, 0],
```

```
7     [0, 0, 0, 4, 0, 0, 0, 0],
```

```
8     [0, 0, 0, 0, 5, 0, 0, 0],
```

```
9     [0, 0, 0, 0, 0, 6, 0, 0],
```

```
10    [0, 0, 0, 0, 0, 0, 0, 0],
```

```
11    [0, 0, 0, 0, 0, 0, 0, 0],
```

```
12    [0, 0, 0, 0, 0, 0, 0, 0],
```

```
13    [0, 0, 1, 2, 3, 4, 0, 0],
```

## 5. 블록 낙하 알고리즘

### ■ 리스트 값 확인하기

#### ■ 예제 ex0805\_1.py

```
14 ]
15
16 def draw_neko( ) :
17     for y in range(10) :
18         for x in range(8) :
19             if neko[y][x] > 0 :
20                 cvs.create_image(x*72+60,
21                                 y*72+60, image=img_neko[neko[y][x],
22                                 tag="NEKO")
21
```

고양이 표시 함수

반복: y는 0부터 9까지 1씩 증가

반복: x는 0부터 7까지 1씩 증가

리스트 엘리먼트 값이 0보다 크면

고양이 이미지 표시

## 5. 블록 낙하 알고리즘

### ■ 리스트 값 확인하기

■ 예제 ex0805\_1.py

```
22 def drop_neko( ) :
23     for y in range(8, -1, -1) :
24         for x in range(8) :
25             if neko[y][x] != 0 and
neko[y+1][x] == 0 :
26                 neko[y+1][x] = neko[y][x]
27                 neko[y][x] = 0
28
29 def game_main( ) :
30     drop_neko( )
31     cvs.delete("NEKO")
32     draw_neko( )
33     root.after(100, game_main)
```

고양이 낙하 함수

반복: y는 8부터 0까지 1씩 감소

반복: x는 0부터 7까지 1씩 증가

고양이가 있는 칸의 아래 칸이 비었다면

빈 칸에 고양이를 넣음

원래 고양이가 있던 칸을 비움

메인처리(실시간 처리) 수행 함수

고양이 낙하 함수 호출

고양이 이미지 삭제

고양이 표시

0.1초 후 메인 함수 재호출

## 5. 블록 낙하 알고리즘

### ■ 리스트 값 확인하기

#### ■ 예제 ex0805\_1.py

```
34
35 root = tkinter.Tk( )
36 root.title("고양이 낙하시키기")
37 root.resizable(False, False)
38 cvs = tkinter.Canvas(root, width=912,
39                        height=768)
40
41 bg = tkinter.PhotoImage(file="neko_bg.png")
42 img_neko = [
43     None,
44     tkinter.PhotoImage(file="neko1.png"),
45     tkinter.PhotoImage(file="neko2.png"),
```

윈도우 객체 생성

윈도우 제목 지정

윈도우 크기 변경 불가 설정

캔버스 컴포넌트 생성

캔버스 컴포넌트 배치

배경 이미지 로딩

여러 고양이 이미지 관리

## 5. 블록 낙하 알고리즘

### ■ 리스트 값 확인하기

#### ■ 예제 ex0805\_1.py

```
46     tkinter.PhotoImage(file="neko3.png"),
47     tkinter.PhotoImage(file="neko4.png"),
48     tkinter.PhotoImage(file="neko5.png"),
49     tkinter.PhotoImage(file="neko6.png"),
50     tkinter.PhotoImage(file="neko_niku.png"),
51 ]
52
53 cvs.create_image(456, 384, image=bg)
54 game_main( )
55 root.mainloop( )
```

캔버스에 배경 그리기  
고양이 표시 함수 호출  
윈도우 표시

- 이 프로그램을 실행하면 고양이가 떨어집니다.
- 여러 차례 실행하면서 동작을 확인해보기 바랍니다.



### ■ 리스트 값 확인하기

- 예제 ex0805\_1.py
- 22~27 번 행에 정의한 `drop_neko( )`가 고양이를 떨어뜨리는 함수입니다.
- 이 함수는 `neko[y][x]` 위치에 고양이가 있고 그 아래 위치인 `neko[y+1][x]`의 값이 0이면 (즉 빈칸이라면), `neko[y+1][x]`에 `neko[y][x]`의 값을 넣고, `neko [y][x]`의 값을 0으로 만듭니다.
- 이로써 고양이를 한 칸 아래로 이동시킵니다.
- 이 처리는 2중 반복 `for` 구문으로 아래 칸부터 위 칸 방향으로 수행해야 합니다.
- 그림으로 나타내면 다음과 같습니다.

## 5. 블록 낙하 알고리즘

### ■ 리스트 값 확인하기

- 그림으로 나타내면 다음과 같습니다.



### ■ 리스트 값 확인하기

- 가장 아래 행은 확인하지 않아도 되므로 1행부터 확인합니다.
- 2중 반복 for 구문에서 y의 값이 8인 경우 x의 값은 0→1→2→3→4→5→6→7로 변화하면서 옆으로 확인을 수행하며, 떨어뜨려야 할 고양이가 있다면 한 층 낙하시킵니다.
- 다음으로 y 값이 7이 되고 2행을 확인합니다.
- 이 과정을 y 값이 0, x 값이 7이 될 때까지 2중 반복을 수행합니다.
- 만약 이를 위에서 아래 방향으로 수행하면 1행의 고양이가 2행으로 이동하고 2행으로 이동한 그 고양이가 3행으로 이동하는 과정이 한꺼번에 수행되면서 단체적이 아니라 한 번에 맨 아래로 떨어져 버립니다.
- `drop_neko()` 함수를 30번 행과 같이 실시간 처리 중 호출함으로써 고양이가 자동으로 떨어지도록 합니다.

## 6. 클릭해서 블록 떨어뜨리기

■ 이번 절에서는 클릭한 위치에 고양이를 놓는 처리를 추가합니다.

■ 블록 위치와 낙하

- 2절부터 5절까지 작성했던 프로그램을 조합해 클릭한 칸에 고양이를 놓을 수 있게 합니다.
- 칸에 놓은 고양이는 자동으로 낙하합니다.
- 예제 `ex0806_1.py`

```
1 import tkinter
2 import random
3
4 cursor_x = 0
5 cursor_y = 0
6 mouse_x = 0
7 mouse_y = 0
8 mouse_c = 0
```

```
tkinter 모듈 импорт
random 모듈 импорт

커서의 X 좌표
커서의 Y 좌표
마우스 포인터의 X 좌표
마우스 포인터의 Y 좌표
마우스 포인터 클릭 판정 변수
```

## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

#### ■ 예제 ex0806\_1.py

```
9
10 def mouse_move(e) :
11     global mouse_x, mouse_y
12     mouse_x = e.x
13     mouse_y = e.y
14
15 def mouse_press(e) :
16     global mouse_c
17     mouse_c = 1
18
19 neko = [
20     [0, 0, 0, 0, 0, 0, 0, 0],
```

마우스 포인터 이동 시 실행할 함수  
mouse\_x, mouse\_y를 전역 변수로 선언  
mouse\_x에 마우스 포인터의 X좌표 대입  
mouse\_y에 마우스 포인터의 Y좌표 대입

마우스 버튼 클릭 시 실행할 함수  
mouse\_c를 전역 변수로 선언  
mouse\_c에 1 대입

칸을 관리할 2차원 리스트

## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

#### ■ 예제 ex0806\_1.py

```
21 [0, 0, 0, 0, 0, 0, 0, 0],  
22 [0, 0, 0, 0, 0, 0, 0, 0],  
23 [0, 0, 0, 0, 0, 0, 0, 0],  
24 [0, 0, 0, 0, 0, 0, 0, 0],  
25 [0, 0, 0, 0, 0, 0, 0, 0],  
26 [0, 0, 0, 0, 0, 0, 0, 0],  
27 [0, 0, 0, 0, 0, 0, 0, 0],  
28 [0, 0, 0, 0, 0, 0, 0, 0],  
29 [0, 0, 0, 0, 0, 0, 0, 0],  
30 ]  
31
```

## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

```
32 def draw_neko( ) :
33     for y in range(10) :
34         for x in range(8) :
35             if neko[y][x] > 0 :
36                 cvs.create_image(x*72+60,
y*72+60, image=img_neko[neko[y][x],
tag="NEKO")
37
38 def drop_neko( ) :
39     for y in range(8, -1, -1) :
40         for x in range(8) :
41             if neko[y][x] != 0 and
neko[y+1][x] == 0 :
42                 neko[y+1][x] = neko[y][x]
43                 neko[y][x] = 0
```

고양이 표시 함수

반복: y는 0부터 9까지 1씩 증가

반복: x는 0부터 7까지 1씩 증가

리스트 엘리먼트 값이 0보다 크면

고양이 이미지 표시

고양이 낙하 함수

반복: y는 8부터 0까지 1씩 감소

반복: x는 0부터 7까지 1씩 증가

고양이가 있는 칸의 아래 칸이 비  
었다면

빈 칸에 고양이를 넣음

원래 고양이가 있던 칸을 비움

## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

```
44
45 def game_main( ) :
46     global cursor_x, cursor_y
47
48     drop_neko( )
49     if 24 <= mouse_x and mouse_x
50     < 24+72*8 and 24 <= mouse_y and
51     mouse_y < 24+72*10 :
52         cursor_x = int((mouse_x - 24) / 72)
53         cursor_y = int((mouse_y - 24) / 72)
54
55         if mouse_c == 0 :
56             mouse_c = 0
57             neko[cursor_y][cursor_x] =
58             random.randint(1, 6)
```

메인처리(실시간 처리) 수행 함수  
cursor\_x, cursor\_y를 전역 변수로 선언

고양이 낙하 함수 호출

마우스 포인터 좌표가 게임 영역 내에 있으면

포인터의 X 좌표에서 커서의 X 좌표 계산

포인터의 Y 좌표에서 커서의 Y 좌표 계산

마우스 포인터를 클릭했다면

클릭 플래그 해제

커서 위치 칸에 무작위로 고양이 배치



## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

```
54     cvs.delete("CURSOR")
55     cvs.create_image(cursor_x*72+60,
56     cursor_y*72+60, image=cursor,
57     tag="CURSOR")
58     cvs.delete("NEKO")
59     draw_neko( )
60     root.after(100, game_main)
61
62 root = tkinter.Tk( )
63 root.title("클릭해서 고양이 놓기")
64 root.resizable(False, False)
65 root.bind("<Motion>", mouse_move)
66
67 root.bind("<ButtonPress>", mouse_press)
```

커서 삭제

새로운 위치에 커서 표시

고양이 이미지 삭제

고양이 표시

0.1초 후 메인 함수 재호출

윈도우 객체 생성

윈도우 제목 지정

윈도우 크기 변경 불가 설정

마우스 포인터 이동시 실행할 함수 지정

마우스 버튼 클릭시 실행할 함수 지정

## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

```
65 cvs = tkinter.Canvas(root, width=912, height=768)
66 cvs.pack( )
67
68 bg = tkinter.PhotoImage(file="neko_bg.png")
69 cursor =
   tkinter.PhotoImage(file="neko_cursor.png")
70 img_neko = [
71     None,
72     tkinter.PhotoImage(file="neko1.png"),
73     tkinter.PhotoImage(file="neko2.png"),
74     tkinter.PhotoImage(file="neko3.png"),
75     tkinter.PhotoImage(file="neko4.png"),
```

캔버스 컴포넌트 생성

캔버스 컴포넌트 배치

배경 이미지 로딩

커서 이미지 로딩

여러 고양이 이미지 관리

## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

#### ■ 예제 ex0806\_1.py

```
76     tkinter.PhotoImage(file="neko5.png"),
77     tkinter.PhotoImage(file="neko6.png"),
78     tkinter.PhotoImage(file="neko_niku.png")
79 ]
80
81 cvs.create_image(456, 384, image=bg)
82 game_main( )
83 root.mainloop( )
```

캔버스에 배경 그리기  
고양이 표시 함수 호출  
윈도우 표시

- 이 프로그램을 실행한 뒤 마우스로 커서를 이동한 후에 클릭하면 그 칸에 고양이를 놓을 수 있습니다.
- 칸에 놓인 고양이는 아래로 떨어집니다.

## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

- 예제 ex0806\_1.py
- 48~53번 행에서 커서 이동과 클릭한 칸에 고양이를 놓는 처리를 수행합니다.
- 이 부분의 프로그램은 다음과 같습니다.

```
if 24 <= mouse_x and mouse_x < 24+72*8 and 24 <= mouse_y and mouse_y < 24+72*10:  
    cursor_x = int((mouse_x-24)/72)  
    cursor_y = int((mouse_y-24)/72)  
    if mouse_c == 1:  
        mouse_c = 0  
        neko[cursor_y][cursor_x] = random.randint(1, 6)
```

- 파란 태두리의 if 구문에서 마우스 포인터가 칸 위에 있는지 판정합니다.
- 칸 위에 있다면 커서의 가로 방향 위치와 세로 방향 위치를 계산합니다.

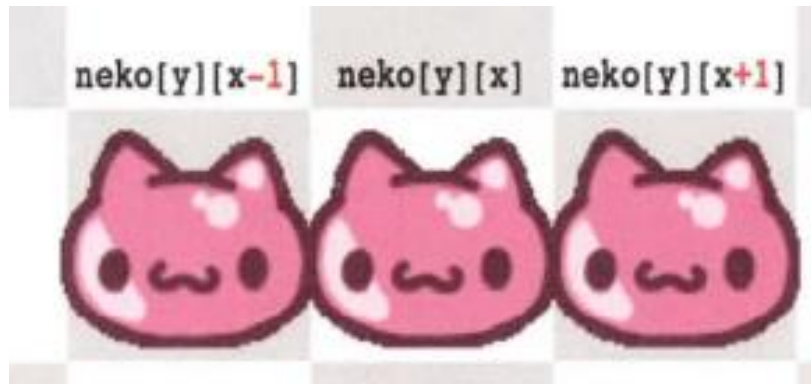
## 6. 클릭해서 블록 떨어뜨리기

### ■ 블록 위치와 낙하

- 빨간 태두리의 if 구문에서 마우스 버튼을 클릭했는지 판정하고 고양이를 놓습니다.
- 이때 `mouse_c` 값에 0을 대입하므로 클릭할 때마다 고양이를 놓습니다.
- 시험 삼아 `'mouse_c = 0'` 부분을 삭제하거나 주석 처리한 뒤 프로그램을 실행합니다.
- 그러면 클릭했을 때 고양이가 계속 놓입니다.
- 고양이를 놓을 때 `'mouse_c = 0'`를 함으로써 클릭 시 들려 있던 플래그를 내라는 것이 핵심입니다.
- 2절의 마우스 입력 프로그램 `ex0802_1.py`에 있는 '마우스 버튼에서 손을 뗄 때 실행하는 함수'를 넣을 필요는 없습니다.
- 이후 프로그램에서도 마우스 버튼에서 손을 뗄 때의 처리는 하지 않습니다.
- 여기에서 설명한 if 구문 안에 또 다른 if 구문을 넣는 것은 소프트웨어 개발 시 매우 자주 사용하므로 이 구조를 잘 이해해 두도록 합시다.

## 7. 블록 모임 판정 알고리즘

- 낙하 퍼즐에서 블록이 모였는지 판정하는 알고리즘이 필요합니다.
- 이를 어떻게 프로그래밍하면 좋을지 생각해 봅시다.
- 블록이 나란히 있는지 판정할 수 있는 방법은 여러가지가 있지만, 여기에서는 프로그래밍 초보자가 이해하기 쉬운 판정 방법을 설명하겠습니다.
- 3개가 나란히 늘어선 상태 확인하기
  - 가로로 3개가 나란히 늘어선 상태를 판정하는 방법을 생각해 봅시다.
  - 다음 그림과 같이 가운데 위치한 고양이 `neko[y][x]`의 값이 왼쪽 고양이 `neko[y][x-1]`, 오른쪽 고양이 `neko[y][x+1]`의 값과 일치한다면 가로로 3개가 늘어선 상태임을 알 수 있습니다.



## 7. 블록 모임 판정 알고리즘

### ■ 3개가 나란히 늘어선 상태 확인하기

- 이 판정을 2중 반복 for 구문으로 모든 칸에 적용하면 가로로 3개 이어진 장소를 판정할 수 있습니다.
- 이 방법으로 판정하는 프로그램을 작성해 보겠습니다.
- 고양이를 낙하하는 처리는 생략하고 동작 확인이 쉽도록, 클릭한 위치에 분홍 혹은 하늘 색 고양이 중 하나가 떨어지도록 했습니다.
- 풍선에 표시한 '테스트'라는 문자를 클릭해서 3개가 이어진 고양이가 있다면 발자국 이미지로 바꿉니다.
- 이 프로그램에서는 2차원 리스트 `neko`를 `append( )` 명령으로 준비합니다.
- `append( )` 명령에 대해서는 동작 확인 후 설명하겠습니다.

## 7. 블록 모임 판정 알고리즘

### ■ 3개가 나란히 늘어선 상태 확인하기

- 예제 ex0807\_1.py (※ 고양이 낙하 관련 처리는 굵게 표시)

```
1 import tkinter
2 import random
```

```
3
```

```
4 cursor_x = 0
```

```
5 cursor_y = 0
```

```
6 mouse_x = 0
```

```
7 mouse_y = 0
```

```
8 mouse_c = 0
```

```
9
```

```
10 def mouse_move(e) :
```

```
11     global mouse_x, mouse_y
```

```
12     mouse_x = e.x
```

```
13     mouse_y = e.y
```

tkinter 모듈 импорт

random 모듈 импорт

커서 가로 방향 위치(왼쪽부터 몇 번째 칸인지)

커서 세로 방향 위치(위쪽부터 몇 번째 칸인지)

마우스 포인터 X 좌표

마우스 포인터 Y 좌표

마우스 포인터 클릭 판정 변수

마우스 포인터 이동 시 실행할 함수

mouse\_x, mouse\_y를 전역변수로 선언

mouse\_x에 마우스 포인터의 X좌표 대입

mouse\_y에 마우스 포인터의 Y좌표 대입



## 7. 블록 모임 판정 알고리즘

```
14
15 def mouse_press(e) :
16     global mouse_c
17     mouse_c = 1
18
19     neko = [ ]
20     for i in range(10) :
21         neko.append([0, 0, 0, 0, 0, 0, 0, 0])
22
23     def draw_neko( ) :
24         for y in range(10) :
25             for x in range(8) :
26                 if neko[y][x] > 0 :
27                     cvs.create_image(x*72+60,
y*72+60, image=img_neko[neko[y][x]],
tag="NEKO")
```

클릭  
마우스 버튼 클릭 시 실행할 함수  
mouse\_c를 전역변수로 선언  
mouse\_c에 1을 대입

칸을 관리할 2차원 리스트

append( ) 명령으로 리스트 초기화

고양이 표시 함수

반복: y는 0부터 9까지 1씩 증가

반복: x는 0부터 7까지 1씩 증가

리스트 엘리먼트 값이 0보다 크면

고양이 이미지 표시

```

28
29 def yoko_neko( ) :
30     for y in range(10) :
31         for x in range(1, 7) :
32             if neko[y][x] > 0 :
33                 if neko[y][x-1] == neko[y][x]
and neko[y][x+1] == neko[y][x] :
34                     neko[y][x-1] = 7
35                     neko[y][x] = 7
36                     neko[y][x+1] = 7
37
38 def game_main( ) :
39     global cursor_x, cursor_y, mouse_c
40     if 660 <= mouse_x and mouse_x
< 840 and 100 <= mouse_y and
mouse_y < 160 and mouse_c == 1 :
41         mouse_c = 0
42         yoko_neko( )

```

국:

고양이가 가로로 3개 놓였는지 확인  
하는 함수

반복: y는 0부터 9까지 1씩 증가

반복: x는 1부터 6까지 1씩 증가

칸에 고양이가 놓여 있고

좌우에 같은 고양이가 놓였다면

해당 칸을 발자국으로 변경

해당 칸을 발자국으로 변경

해당 칸을 발자국으로 변경

메인처리(실시간 처리) 수행 함수

전역 변수 선언

풍선의 "테스트"라는 문자를 클릭  
하면

클릭 플래그 해제

가로 놓임 확인 함수 실행

## 7. 블록 모임 판정 알고리즘

```
43     if 24 <= mouse_x and mouse_x  
44         < 24+72*8 and 24 <= mouse_y and  
45         mouse_y < 24+72*10 :
```

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

```
45         cursor_y = int((mouse_y - 24) / 72)
```

```
46         if mouse_c == 1 :
```

```
47             mouse_c = 0
```

```
48             neko[cursor_y][cursor_x] =  
49             random.randint(1, 2)
```

```
49         cvs.delete("CURSOR")
```

```
50         cvs.create_image(cursor_x*72+60,  
51         cursor_y*72+60, image=cursor,  
52         tag="CURSOR")
```

```
51         cvs.delete("NEKO")
```

```
52         draw_neko( )
```

```
53         root.after(100, game_main)
```

클릭

마우스 포인터 좌표가 격자 위라면

마우스 X좌표에서 커서 가로위치  
계산

마우스 Y좌표에서 커서 세로위치  
계산

마우스를 클릭했다면

클릭 플래그 해제

커서 위치 칸에 무작위로 고양이  
배치

커서 삭제

새로운 위치에 커서 표시

고양이 이미지 삭제

고양이 표시

0.1초 후 메인 함수 호출

## 7. 블록 모임 판정 알고리즘

### ■ 3개가 나란히 늘어선 상태 확인하기

- 예제 ex0807\_1.py (※ 고양이 낙하 관련 처리는 굵게 표시)

```
54
55 root = tkinter.Tk( )
56 root.title("가로로 3개가 나란히 놓였는가?")
57 root.resizable(False, False)
58 root.bind("<Motion>", mouse_move)
59 root.bind("<ButtonPress>", mouse_press)
60 cvs = tkinter.Canvas(root, width=912,
61                       height=768)
62
63 bg =
64   tkinter.PhotoImage(file="neko_bg.png")
65
66 cursor =
67   tkinter.PhotoImage(file="neko_cursor.png")
```

윈도우 객체 생성

윈도우 제목 지정

윈도우 크기 변경 불가 설정

마우스 이동 시 실행할 함수 지정

마우스 클릭 시 실행할 함수 지정

캔버스 컴포넌트 생성

캔버스 컴포넌트 배치

배경 이미지 로딩

커서 이미지 로딩

## 7. 블록 모임 판정 알고리즘

### ■ 3개가 나란히 늘어선 상태 확인하기

- 예제 ex0807\_1.py (※ 고양이 낙하 관련 처리는 굵게 표시)

```
65 img_neko = [  
66     None,  
67     tkinter.PhotoImage(file="neko_1.png")  
68     tkinter.PhotoImage(file="neko_2.png")  
69     tkinter.PhotoImage(file="neko_3.png")  
70     tkinter.PhotoImage(file="neko_4.png")  
71     tkinter.PhotoImage(file="neko_5.png")  
72     tkinter.PhotoImage(file="neko_6.png")  
73     tkinter.PhotoImage(file=  
74         "neko_niku.png")  
75 ]
```

리스트로 여러 고양이 이미지 관리  
첫번째 요소는 아무것도 없는 값임

## 7. 블록 모임 판정 알고리즘

### ■ 3개가 나란히 늘어선 상태 확인하기

- 예제 ex0807\_1.py (※ 고양이 낙하 관련 처리는 굵게 표시)

76	<code>cvs.create_image(456, 384, image=bg)</code>	캔버스에 배경 그리기
77	<code>cvs.create_rectangle(660, 100, 840, 160, fill="white")</code>	풍선 내 테두리 그리기
78	<code>cvs.create_text(750, 130, text="테스트", fill="red", font=("Times New Roman", 30))</code>	테스트 문자 표시
79	<code>game_main( )</code>	메인 처리 수행 함수 호출
80	<code>root.mainloop( )</code>	윈도우 표시

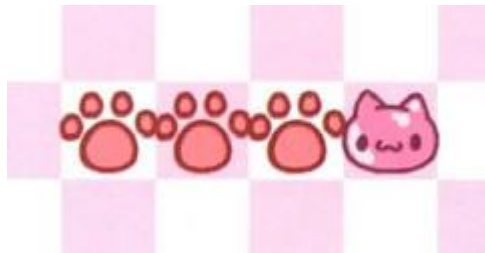
- 이 프로그램을 실행하고 칸을 클릭해서 예를 들면, 다음과 같이 하늘색 고양이 3개를 나란히 놓습니다.



## 7. 블록 모임 판정 알고리즘

### ■ 3개가 나란히 늘어선 상태 확인하기

- 그리고 풍선 안의 테스트 문자를 클릭하면 하늘색 고양이 발자국으로 바뀝니다.



- 29~36번 행이 가로로 3개 연결된 것을 판정하는 `yoko_neko( )` 함수입니다.
- 2중 반복 `for` 구문으로 칸 전체를 확인하고 같은 고양이가 가로로 3개 놓여 있으면, 해당 칸(리스트 엘리먼트)을 발자국 값인 7로 바꿉니다.
- 40~42번 행에서 테스트 문자를 클릭할 때 이 함수를 호출합니다.
- `yoko_neko( )` 함수의 2중 반복에서 `x`의 범위는 `range(1, 7)`로 지정해 1부터 6까지의 값을 사용합니다.
- 확인해야 할 리스트를 `neko[y][x-1]`, `neko[y][x]`, `neko[y][x+1]`으로 하기 때문입니다.
- 만약 `range(0, 8)`로 지정해 `x` 값 범위를 0부터 7까지로 하면 0과 7인 경우 범위를 벗어난 것(out of index, `neko[y][-1]`, `neko[y][8]`)을 확인하게 되므로 에러가 발생합니다.

### ■ append( ) 명령

- 앞의 ex0806\_1.py에서는 2차원 리스트 `neko`를 `[0, 0, 0, 0, 0, 0, 0, 0]` 리스트 10개를 이어서 썼습니다.
- 보기에는 면하지만 프로그램 입장에서는 약간 낭비라고 할 수 있습니다.
- 여기에서는 리스트에 엘리먼트를 추가하는 `append( )` 명령을 사용해 간략하게 만들 수 있습니다.
- 19번 행에서 `neko = [ ]`로 빈 리스트를 준비한 뒤, 20 ~21번 행에서 반복 `for` 구문과 `append( )` 명령으로 `[0, 0, 0, 0, 0, 0, 0, 0]`을 10개 행만큼 추가합니다.



## 7. 블록 모임 판정 알고리즘

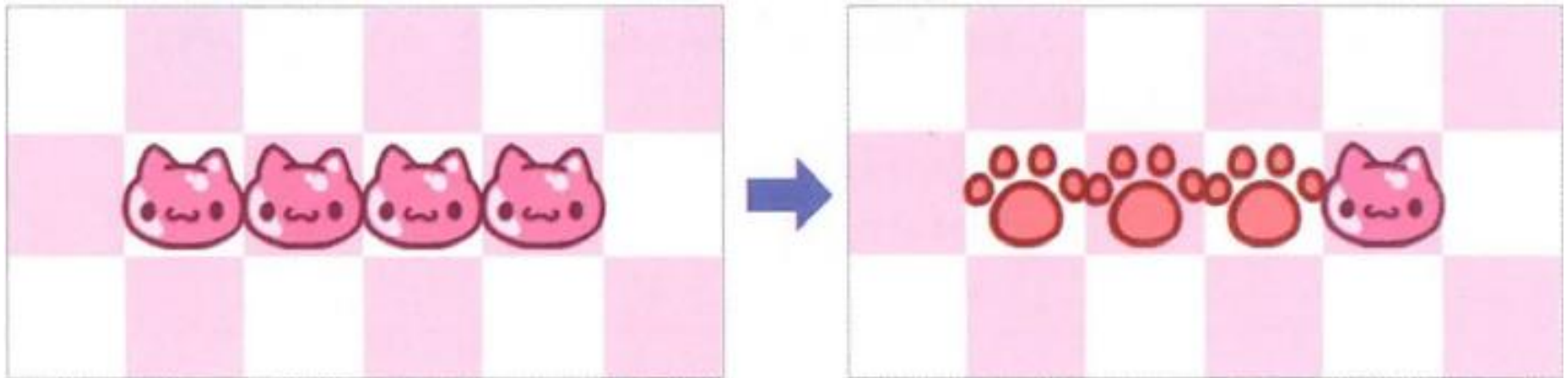
### ■ 이 판정 방법의 문제점

- 이 방법에서 가로로 3개가 나란히 놓인 것을 판정할 수 있음을 알았습니다.
- 동일하게 세로로 3개 나란히 놓인 것, 대각선으로 3개 나란히 놓인 것을 판정할 수 있으면 됩니다.
- 하지만 실은 예시로 든 방법에는 두 가지 문제점이 있습니다.

#### 문제점 ①

4개, 5개, 7개, 8개가 나란히 놓인 상황에서는 판정할 수 없는 칸이 존재합니다.

- 예를 들어, 4개가 나란히 연결되었을 때 테스트 문자를 클릭해 봅니다.
- 다음 그림과 같이 가장 오른쪽 고양이 하나가 남게 됩니다.



## 7. 블록 모임 판정 알고리즘

### ■ 이 판정 방법의 문제점

문제점 ②

가로로 나란히 놓였는지 판정한 후, 세로로 나란히 놓였는지 판정하면 올바르게 판정할 수 없습니다.

- 다음과 같이 프로그램을 입력한 뒤, 세로로 3개가 나란히 놓였는지도 판정해 보도록 합니다.

```
def yoko_neko():
```

```
    for y in range(10):
        for x in range(1, 7):
            if neko[y][x] > 0:
                if neko[y][x-1] == neko[y][x] and neko[y]
[x+1] == neko[y][x]:
                    neko[y][x-1] = 7
                    neko[y][x] = 7
                    neko[y][x+1] = 7
```

가로로 3개가 나란히 놓였는가?

## 7. 블록 모임 판정 알고리즘

### ■ 이 판정 방법의 문제점

문제점 ②

가로로 나란히 놓였는지 판정한 후, 세로로 나란히 놓였는지 판정하면 올바르게 판정할 수 없습니다.

- 다음과 같이 프로그램을 입력한 뒤, 세로로 3개가 나란히 놓였는지도 판정해 보도록 합니다.

```
for y in range(1, 9):  
    for x in range(8):  
        if neko[y][x] > 0:  
            if neko[y-1][x] == neko[y][x] and neko[y+1]  
[x] == neko[y][x]:  
                neko[y-1][x] = 7  
                neko[y][x] = 7  
                neko[y+1][x] = 7
```

세로로 3개가 나란히 놓였는가?

## 7. 블록 모임 판정 알고리즘

### ■ 이 판정 방법의 문제점

- 이 프로그램에서는 다음 그림의 왼쪽과 같은 경우에는 판정할 수 있으나, 오른쪽과 같이 십자로 놓인 경우에는 가로 방향을 먼저 판정하므로 세로 방향은 판정하지 못합니다.



## 7. 블록 모임 판정 알고리즘

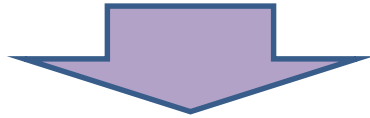
### ■ 이 판정 방법의 문제점

- 다음 절에서는 고양이가 어떤 형태로 나란히 놓여 있더라도 올바르게 판정하도록 알고리즘을 개선하겠습니다.
- Ex0807\_1.py의 프로그램은 낭비였던 것이 아니라, 판정 방법을 개선한 알고리즘입니다.

## 8. 올바른 알고리즘 조합하기

- 7절에서 찾아낸 문제점을 해결하고 가로, 세로, 대각선으로 3개 이상의 블록이 나란히 놓였는지 정확하게 판정하는 알고리즘을 완성하겠습니다.
- 판정용 리스트 사용하기
  - 7절에서 설명한 문제가 발생하지 않는 판정은 다음과 같이 수행합니다.

① 판정용 리스트를 준비하고, 해당 리스트에 칸별 데이터를 복사합니다.



② 판정용 리스트를 조사해서 같은 고양이가 3개 나란히 놓였는지 확인하고 나란히 놓였다면 게임용 리스트 값을 변경합니다.

## 8. 올바른 알고리즘 조합하기

### ■ 판정용 리스트 사용하기

- ①을 이미지로 나타내면 다음과 같습니다.
- 판정용 리스트 명은 `check`로 하겠습니다.



## 8. 올바른 알고리즘 조합하기

### ■ 판정용 리스트 사용하기

- 예제 ex0808\_1.py (※ 3개가 나란히 놓였는지 정확히 판정하는 처리 관련 처리는 굵게 표시)

```
1 import tkinter
2 import random
```

```
3
4 cursor_x = 0
5 cursor_y = 0
6 mouse_x = 0
7 mouse_y = 0
8 mouse_c = 0
```

```
9
10 def mouse_move(e) :
11     global mouse_x, mouse_y
12     mouse_x = e.x
13     mouse_y = e.y
```

tkinter 모듈 импорт  
random 모듈 импорт

커서 가로 방향 위치(왼쪽부터 몇 번째 칸인지)  
커서 세로 방향 위치(위쪽부터 몇 번째 칸인지)  
마우스 포인터 X 좌표  
마우스 포인터 Y 좌표  
마우스 포인터 클릭 판정 변수

마우스 포인터 이동 시 실행할 함수  
mouse\_x, mouse\_y를 전역변수로 선언  
mouse\_x에 마우스 포인터의 X좌표 대입  
mouse\_y에 마우스 포인터의 Y좌표 대입



## 8. 올바른 알고리즘 조합하기

### ■ 판정용 리스트 사용하기

- 예제 ex0808\_1.py (※ 3개가 나란히 놓였는지 정확히 판정하는 처리 관련 처리는 굵게 표시)

```
14
15 def mouse_press(e) :
16     global mouse_c
17     mouse_c = 1
18
19     neko = [ ]
20     check = [ ]
21     for i in range(10) :
22         neko.append([0, 0, 0, 0, 0, 0, 0, 0, 0])
23         check.append([0, 0, 0, 0, 0, 0, 0, 0, 0])
24
```

마우스 버튼 클릭 시 실행할 함수  
mouse\_c를 전역변수로 선언  
mouse\_c에 1을 대입

칸을 관리할 2차원 리스트  
판정용 2차원 리스트

append( ) 명령으로 리스트 초기화

## 8. 올바른 알고리즘 조합하기

### ■ 판정용 리스트 사용하기

- 예제 ex0808\_1.py (※ 3개가 나란히 놓였는지 정확히 판정하는 처리 관련 처리는 굵게)

```
25 def draw_neko( ) :  
26     for y in range(10) :  
27         for x in range(8) :  
28             if neko[y][x] > 0 :  
29                 cvs.create_image(x*72+60,  
30                 y*72+60, image=img_neko[neko[y][x]],  
31                 tag="NEKO")  
32  
33 def check_neko( ) :  
34     for y in range(10) :  
35         for x in range(8) :  
36             check[y][x] = neko[y][x]
```

고양이 표시 함수

반복: y는 0부터 9까지 1씩 증가

반복: x는 0부터 7까지 1씩 증가

리스트 엘리먼트 값이 0보다 크면

고양이 이미지 표시

고양이가 가로, 세로, 대각선 3개인  
지 확인하는 함수

반복: y는 0부터 9까지 1씩 증가

반복: x는 0부터 7까지 1씩 증가

판정용 리스트에 고양이 값 복사

## 8. 올바른 알고리즘 조합하기

```
36     for y in range(1, 9) :
37         for x in range(8) :
38             if check[y][x] > 0 :
39                 if check[y-1][x] == check[y][x]
and check[y+1][x] == check[y][x] :
40                     neko[y-1][x] = 7
41                     neko[y][x] = 7
42                     neko[y+1][x] = 7
43
44     for y in range(10) :
45         for x in range(1, 7) :
46             if check[y][x] > 0 :
47                 if check[y][x-1] == check[y][x]
and check[y][x+1] == check[y][x] :
48                     neko[y][x-1] = 7
49                     neko[y][x] = 7
50                     neko[y][x+1] = 7
```

확

반복: y는 0부터 8까지 1씩 증가  
반복: x는 0부터 7까지 1씩 증가  
칸에 고양이가 놓여 있고  
위 아래가 같은 고양이라면

해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경

반복: y는 0부터 9까지 1씩 증가  
반복: x는 0부터 6까지 1씩 증가  
칸에 고양이가 놓여 있고  
좌우가 같은 고양이라면

해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경

## 8. 올바른 알고리즘 조합하기

```
51
52     for y in range(1, 9) :
53         for x in range(1, 7) :
54             if check[y][x] > 0 :
55                 if check[y-1][x-1] ==
check[y][x] and check[y+1][x+1] ==
check[y][x] :
56                     neko[y-1][x-1] = 7
57                     neko[y][x] = 7
58                     neko[y+1][x+1] = 7
59                 if check[y+1][x-1] ==
check[y][x] and check[y-1][x+1] ==
check[y][x] :
60                     neko[y+1][x-1] = 7
61                     neko[y][x] = 7
62                     neko[y-1][x+1] = 7
63
```

확

반복: y는 0부터 8까지 1씩 증가  
반복: x는 0부터 6까지 1씩 증가  
칸에 고양이와 놓여 있고  
왼쪽 위 오른쪽 아래가 같은 고양이  
이라면

해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경  
왼쪽 아래, 오른쪽 위가 같은 고양이  
이라면

해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경  
해당 칸을 발자국으로 변경

## 8. 올바른 알고리즘 조합하기

```
64 def game_main( ) :
65     global cursor_x, cursor_y, mouse_c
66     if 660 <= mouse_x and mouse_x
        < 840 and 100 <= mouse_y and
        mouse_y < 160 and mouse_c == 1 :
67         mouse_c = 0
68         check_neko( )
69         if 24 <= mouse_x and mouse_x
            < 24+72*8 and 24 <= mouse_y and
            mouse_y < 24+72*10 :
70             cursor_x = int((mouse_x - 24) / 72)
71             cursor_y = int((mouse_y - 24) / 72)
72             if mouse_c == 1 :
73                 mouse_c = 0
74                 neko[cursor_y][cursor_x] =
                    random.randint(1, 2)
```

확

메인처리(실시간 처리) 수행 함수  
전역 변수 선언

풍선의 "테스트"라는 문자를 클릭  
하면

클릭 플래그 해제

가로 놓임 확인 함수 실행

마우스 포인터 좌표가 격자 위라면

마우스 X좌표에서 커서 가로위치  
계산

마우스 Y좌표에서 커서 세로위치  
계산

마우스를 클릭했다면

클릭 플래그 해제

커서 위치 칸에 무작위로 고양이  
배치

## 8. 올바른 알고리즘 조합하기

```
75     cvs.delete("CURSOR")
76     cvs.create_image(cursor_x*72+60,
77         cursor_y*72+60, image=cursor,
78         tag="CURSOR")
79     cvs.delete("NEKO")
80     draw_neko( )
81     root.after(100, game_main)
82
83 root = tkinter.Tk( )
84 root.title("세로, 가로, 대각선으로 3개가 나
85     란히 놓였는가?")
86 root.resizable(False, False)
87 root.bind("<Motion>", mouse_move)
88 root.bind("<ButtonPress>", mouse_press)
89 cvs = tkinter.Canvas(root, width=912,
90     height=768)
91 cvs.pack( )
```

확

커서 삭제

새로운 위치에 커서 표시

고양이 이미지 삭제

고양이 표시

0.1초 후 메인 함수 호출

윈도우 객체 생성

윈도우 제목 지정

윈도우 크기 변경 불가 설정

마우스 이동 시 실행할 함수 지정

마우스 클릭 시 실행할 함수 지정

캔버스 컴포넌트 생성

캔버스 컴포넌트 배치

## 8. 올바른 알고리즘 조합하기

```
88
89  bg =
tkinter.PhotoImage(file="neko_bg.png")
90  cursor =
tkinter.PhotoImage(file="neko_cursor.png")
91  img_neko = [
92      None,
93      tkinter.PhotoImage(file="neko_1.png")
94      tkinter.PhotoImage(file="neko_2.png")
95      tkinter.PhotoImage(file="neko_3.png")
96      tkinter.PhotoImage(file="neko_4.png")
97      tkinter.PhotoImage(file="neko_5.png")
98      tkinter.PhotoImage(file="neko_6.png")
99      tkinter.PhotoImage(file=
"neko_niku.png")
100 ]
```

확히 배경 이미지 로딩

커서 이미지 로딩

리스트로 여러 고양이 이미지 관리  
첫번째 요소는 아무것도 없는 값임

## 8. 올바른 알고리즘 조합하기

### ■ 판정용 리스트 사용하기

- 예제 ex0808\_1.py (※ 3개가 나란히 놓였는지 정확히 판정하는 처리 관련 처리는 굵게 표시)

101		
102	<code>cvs.create_image(456, 384, image=bg)</code>	캔버스에 배경 그리기
103	<code>cvs.create_rectangle(660, 100, 840, 160, fill="white")</code>	풍선 내 테두리 그리기
104	<code>cvs.create_text(750, 130, text="테스트", fill="red", font=("Times New Roman", 30))</code>	테스트 문자 표시
105	<code>game_main( )</code>	메인 처리 수행 함수 호출
106	<code>root.mainloop( )</code>	윈도우 표시



## 8. 올바른 알고리즘 조합하기

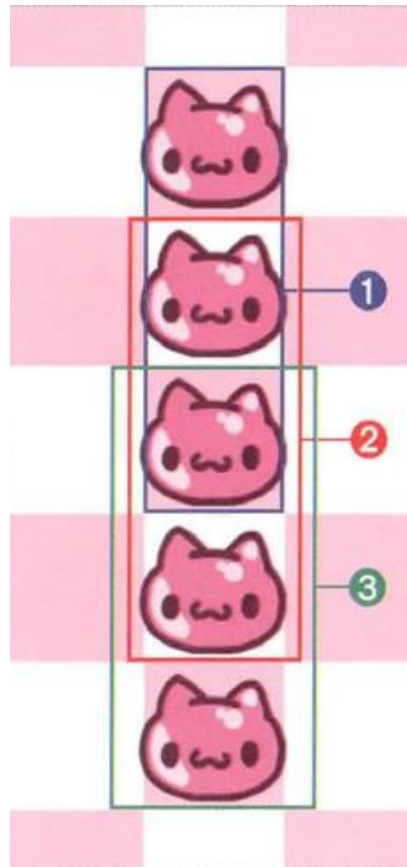
### ■ 판정용 리스트 사용하기

- 31~62번 행에 정의한 `check_neko()` 함수에서 고양이가 가로, 세로, 대각선으로 연결되어 있는지 판정합니다.
- 우선 32~34번 행에서 판정용 리스트 `ckeck`에 `neko`의 데이터를 복사합니다.
- 36~42번 행에서 세로로 나란히 3개가 연결되어 있는지 확인하고, 연결되어 있다면 발자국 모양으로 바꿉니다.
- 확인하는 리스트는 `check[][]`, 발자국으로 바꾸는(즉, 7을 대입하는) 리스트는 `neko[][]`라는 점이 핵심입니다.
- 이 방법으로 세로로 3개가 나란히 놓였는 지 빠짐없이 확인할 수 있습니다.

## 8. 올바른 알고리즘 조합하기

### ■ 판정용 리스트 사용하기

- 예를 들어 다음 그림과 같이 5칸에 나란히 놓인 경우에는 ①, ②, ③의 순서대로 판정이 수행됩니다.
- ①~③ 각각을 판정할 시, `neko [][]`의 값이 7이 되므로 모든 고양이가 발자국으로 바뀝니다.



## 8. 올바른 알고리즘 조합하기

### ■ 판정용 리스트 사용하기

- 44~50번 행에서는 가로 방향 52~62번 행에서는 대각선 방향으로 같은 처리를 수행합니다.
- 고양이를 다양한 형태로 늘어 놓은 후 테스트 문자를 클릭하고 이 알고리즘이 고양이가 연결되어 놓인 곳을 올바르게 판정하는지 확인해 봅시다.
- 리스트는 다양한 방법으로 복사할 수 있습니다.
- 예를 들면, copy 모듈을 사용하기 위해 import copy한 뒤 'check = copy.deepcopy(neko)'라고 입력하면 neko의 값을 복사해서 check를 만들 수 있습니다.
- 파이썬에서 리스트를 복사할 때 주의할 점은 '복사된 리스트 = 복사할 리스트' 명령으로는 리스트를 복사할 수 없다는 것입니다.
- 예를 들어, 이번 프로그램에서 'check = neko'로 입력해도 데이터를 복사한 새로운 리스트가 만들어지지 않고 결과적으로 올바르게 판정할 수도 없습니다.

## 7. 디지털 사진 액자 만들기

### ■ ex0707.py

```
13 root = tkinter.Tk( )
14 root.title("디지털 액자")
15 canvas = tkinter.Canvas(width=800,
16 height=600)
17 canvas.pack( )
18 photo = [
19     tkinter.PhotoImage(file="cat00.png"),
20     tkinter.PhotoImage(file="cat01.png"),
21     tkinter.PhotoImage(file="cat02.png"),
22     tkinter.PhotoImage(file="cat03.png")
23 ]
24 photograph( )
25 root.mainloop( )
```

윈도우 객체 생성

윈도우 타이틀 지정

캔버스 컴포넌트 생성

캔버스 배치

리스트로 이미지 파일 정의

실시간 처리 함수 호출

윈도우 표시

## 7. 디지털 사진 액자 만들기

### ■ ex0707.py

```
13 root = tkinter.Tk( )
14 root.title("디지털 액자")
15 canvas = tkinter.Canvas(width=800,
16 height=600)
17 canvas.pack( )
18 photo = [
19     tkinter.PhotoImage(file="cat00.png"),
20     tkinter.PhotoImage(file="cat01.png"),
21     tkinter.PhotoImage(file="cat02.png"),
22     tkinter.PhotoImage(file="cat03.png")
23 ]
24 photograph( )
25 root.mainloop( )
```

윈도우 객체 생성

윈도우 타이틀 지정

캔버스 컴포넌트 생성

캔버스 배치

리스트로 이미지 파일 정의

실시간 처리 함수 호출

윈도우 표시

## 7. 디지털 사진 액자 만들기

### ■ ex0707.py

- 9번 행의 `len( )` 명령으로 `( )` 안에 입력한 리스트의 엘리먼트 수를 알 수 있습니다.
- 17~22번 행에서 `photo`라는 리스트에 네 종류의 이미지 파일을 정의했으므로 `len(photo)` 값은 4가 됩니다.
- 9~10번 행의 조건 분기에서 가장 마지막 이미지를 표시한 후 다시 처음 이미지부터 표시되도록 하기 위해 `len( )` 명령을 사용합니다.
- 이렇게 하면 리스트에 이미지 파일명을 추가하는 것만으로 프로그램의 다른 부분을 수정하지 않고 모든 이미지를 표시한 뒤, 가장 처음 이미지부터 다시 표시할 수 있습니다.

## 7. 디지털 사진 액자 만들기

### ■ ex0707.py

- 이 프로그램을 실행하면 다음과 같이 고양이 그림들이 순서대로 표시됩니다.



- 몇 초마다 이미지를 변경할 것인지는 11번 행 `after( )` 명령의 인수로 지정합니다.
- 취미 사진이나 가족 사진, 좋아하는 슬라이드 등으로 오리지널 디지털 액자를 만들어 보기 바랍니다.



Thank You

---