Chapter

10

그래픽 사용자 인터페이스 Tkinter와 pygame

학습 목표

- GUI를 이해하고 GUI 표준 모듈인 Tkinter를 사용해 필요한 위젯을 구성하고 윈도를 생성할 수 있다.
- 이벤트 처리를 이해하고 Tkinter에서 이벤트 처리를 구현할 수 있다.
- 써드파티 GUI 모듈인 pygame을 설치해 기본적인 윈도를 구현할 수 있다.

Tkinter 모듈 개요

- **Tkinter**
  - Tcl/Tk에 대한 파이썬 버전으로 Tcl/Tk를 파이썬에 사용할 수 있도록 한 경량(Lightweight)의 그래픽 사용자 인터페이스(GUI: Graphical User Interface) 모듈
    - **Tcl은 'Tool Command Language' 일종의 프로그래밍 언어**
  - GUI
    - **텍스트 모드가 아니라 위젯 또는 콤포넌트라 불리는 윈도나 각종 버튼, 마우스, 다이어로그 등을 사용해 사용자와 프로그램 간의 정보를 교환하는 방식**
    - **바로 우리가 사용하는 윈도나 애플의 운영체제가 바로 GUI**
- **Tkinter(Tk interface)**
  - 파이썬에 기본적으로 내장돼 있는 파이썬 표준 패키지
  - 통합 개발 환경(IDE)인 파이썬 IDLE가 바로 Tkinter를 사용해 작성
  - Tkinter에 관한 자료는 파이썬 홈페이지
    - **python.flowdas.com/library/tkinter.html에서 기본적으로 제공**

첫 tkinter 윈도 만들기

10-1 코딩 | 10-01basictkwin.py | 첫 tkinter 윈도 만들기 | 난이도 기본

```
1   from tkinter import *
2
3   win = Tk()
4   win.geometry('300x200')
5   win.title('반가워요, Tkinter!')
6   win.mainloop()
```

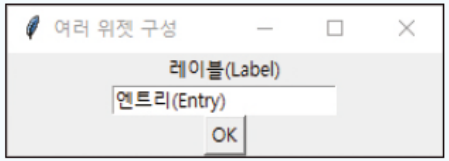
결과

레이블과 엔트리, 버튼 위젯으로 윈도 생성

10-2 코딩 | 10-02widgetpack.py | 레이블과 엔트리, 버튼 위젯으로 윈도 생성 | 난이도 기본

```
1   from tkinter import *
2
3   win = Tk()
4   win.title('여러 위젯 구성')
5
6   lbl = Label(win, text="레이블(Label)") # 레이블
7   lbl.pack() # 레이블을 윈도에 맞게 적정하게 배치
8
9   txt = Entry(win) # 엔트리
10  txt.insert(0, '엔트리(Entry)')
11  txt.pack()
12
13  btn = Button(win, text="OK") # 버튼
14  btn.pack()
15
16  win.mainloop()
```

결과

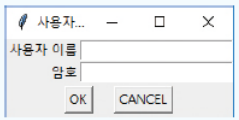


위젯의 테이블 형태 배치 방법인 그리드

10-3 코딩 | 10-03widgetgrid.py | 테이블 형태 배치 그리드 | 난이도 기본

```
1   from tkinter import *
2   win = Tk()
3   win.title('사용자 인증')
4
5   lb1 = Label(win, text='사용자 이름')
6   lb2 = Label(win, text='암호')
7   lb1.grid(row = 0, column = 0, sticky = E) # 현 구역에서 위치를 오른쪽으로 조정
8   lb2.grid(row = 1, column = 0, sticky = E) # 현 구역에서 위치를 오른쪽으로 조정
9
10  et1 = Entry(win)
11  et2 = Entry(win)
12  et1.grid(row = 0, column = 1)
13  et2.grid(row = 1, column = 1)
14
15  # 가장 아래에 버튼 2개를 저장할 레이블을 생성해 배치
16  lb = Label(win)
17  lb.grid(row=2, column=0, columnspan=2) # 두 열을 합해서 배치
18  bt1 = Button(lb, text='OK')
19  bt2 = Button(lb, text='CANCEL')
20  bt1.grid(row=0, column=0, padx = 10) # 외부 여백을 10을 둠
21  bt2.grid(row=0, column=1, padx = 10) # 외부 여백을 10을 둠
22
23  win.mainloop()
```

결과

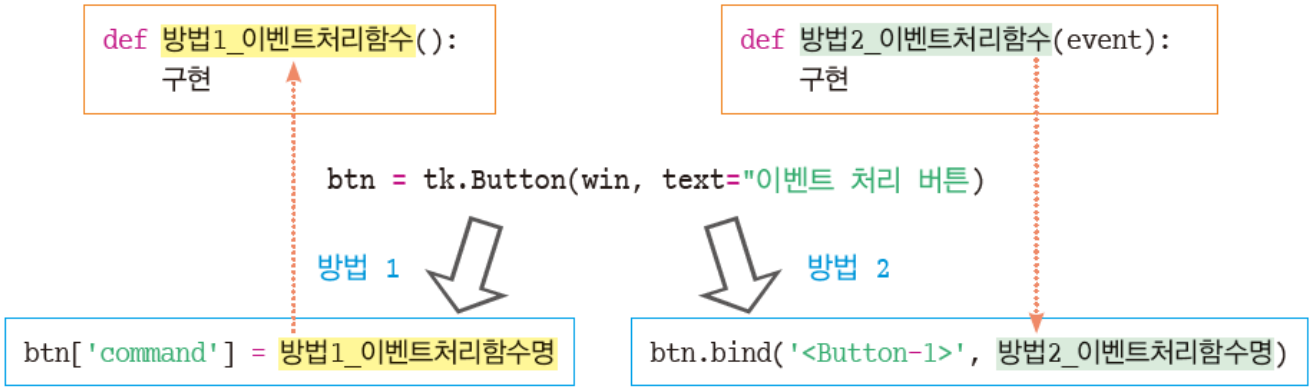


▼ **표1** 함수 grid()에서 키워드 인자 row, column, columnspan, rowspan 사용과 위치

row = 0, column = 0	row = 0, column = 1	row = 0, column = 2
row = 1, column = 0	row = 1, column = 1	
row = 2, column = 0, columnspan=2		row = 1, column = 2, rowspan=2

이벤트 처리 개요

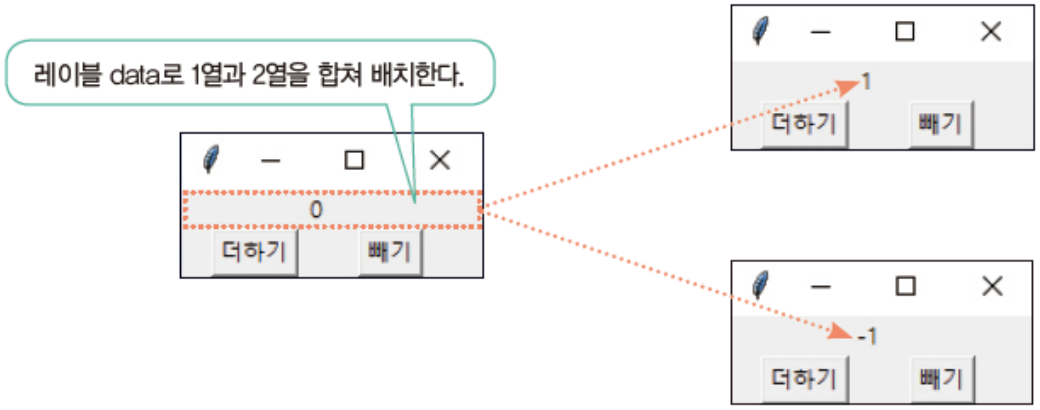
- **'이벤트 처리': 윈도에서 버튼이나 메뉴 클릭에서 특정한 작업을 수행**
  - 이벤트 핸들러(event handler)
    - **발생한 이벤트(event)를 처리하는 작업, 사용자 정의 함수나 시스템 함수에서 수행**
- **Tkinter 의 이벤트 처리 방법은 크게 두 가지**
  - ① 첫 번째 방법
    - **이벤트 진원지인 버튼이나 메뉴의 키워드 인자 'command = 함수명'으로 이벤트 핸들러를 지정하는 방법**
  - ② 두 번째 방법
    - **위젯 함수 bind(이벤트, 함수명)로 이벤트와 이벤트 핸들러인 함수를 연결하는 방법**
    - **정해진 마우스와 키보드의 다양한 이벤트에 대해 처리가 가능**



▲ 그림3 Tkinter 버튼 btn의 마우스 왼쪽 버튼 클릭 이벤트 처리

버튼으로 레이블 값을 변경하는 이벤트 처리

- **함수 add( )**
  - 레이블 data에 표시되는 StringVar 변수
  - get()과 set()으로 참조와 수정
- **함수 sub( )**
  - '빼기' 버튼인 bsub는 함수 sub( )와 연결



▲ 그림4 버튼 이벤트 처리

```
count = StringVar(value = '0') # count.set('0') 또는 value 인자에 첫 값인 0을 지정
data = Label(win, width = 20, textvariable = count)
```

```
def add(): # 1 증가시켜 레이블의 문자열에 지정
    n = int(count.get())
    n += 1
    count.set(n)
```

```
def sub(): # 1 감소시켜 레이블의 문자열에 지정
    n = int(count.get())
    n -= 1
    count.set(n)
```

```
badd = Button(win, text=str[0], command = add) # 누르면 함수 add() 호출
bsub = Button(win, text=str[1], command = sub) # 누르면 함수 sub() 호출
```

▲ 그림5 버튼의 이벤트 처리 구현

버튼으로 레이블의 정숫값 증가와 감소

10-4 코딩 | 10-04btneventhandle.py | 버튼으로 레이블의 정숫값 증가와 감소 | 난이도 응용

```
from tkinter import *

def add(): # 1 증가시켜 레이블의 문자열에 지정
    n = int(count.get())
    n += 1
    count.set(n)

def sub(): # 1 감소시켜 레이블의 문자열에
    n = int(count.get())
    n -= 1
    count.set(n)

str = ['더하기', '빼기']
win = Tk()

# 레이블에 표시되며, 값이 쉽게 레이블에 반영될 값인 count
# tkinter가 제공하는 StringVar라는 객체에 저장
count = StringVar(value = '0') # count.set('0') 또는 value 인자에 첫 값인 0을 지정
# 이후 count.get()으로 참조하고 count.set(값)으로 지정

# 키워드 인자 textvariable: 레이블에 표시할 값을 가져올 변수
# 키워드 인자 textvariable에 위에서 만든 StringVar 변수인 count를 설정
data = Label(win, width = 20, textvariable = count)
data.grid(row=0, column=0, columnspan=2)

badd = Button(win, text=str[0], command = add) # 누르면 함수 add() 호출
badd.grid(row=1, column=0)
bsub = Button(win, text=str[1], command = sub) # 누르면 함수 sub() 호출
bsub.grid(row=1, column=1)

win.mainloop()
```

결과

사용자 로그인 과정에서의 이벤트 처리

10-5 코딩 | 10-05logineventhandle.py | 로그인 과정의 이벤트 처리 | 난이도 응용

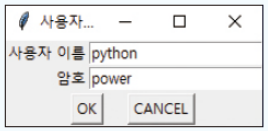
```
1   from tkinter import *
2
3   users = dict(python='power', java='beauty')
4
5   def checkid():
6       uid = et1.get().strip() # 입력된 사용자 이름 문자열받기
7       pwd = et2.get().strip() # 입력된 암호 문자열받기
8       print(uid, pwd)
9       if uid in users.keys(): # 사용자 이름이 있는지 검사
10          if users[uid] == pwd: # 사용자 이름과 암호 일치
11              print('로그인 성공')
12          else:
13              print('암호를 확인하세요.')
14      else:
15          print('사용자 이름을 확인하세요.')
16
17  win = Tk()
18  win.title('사용자 인증')
19
20  lb1 = Label(win, text='사용자 이름')
21  lb2 = Label(win, text='암호')
22  lb1.grid(row = 0, column = 0, sticky = E) # 현 구역에서 위치를 오른
23  lb2.grid(row = 1, column = 0, sticky = E) # 현 구역에서 위치를 오른
```

python의 암호는 power, java의 암호는 beauty다. 사용자 이름과 암호를 추가하려면 딕셔너리의 원소를 추가한다.

```
24
25      et1 = Entry(win)
26      et2 = Entry(win)
27      et1.grid(row = 0, column = 1)
28      et2.grid(row = 1, column = 1)
29
30      # 가장 하단에 버튼 2개를 저장할 레이블을 생성해 배치
31      lb = Label(win)
32      lb.grid(row=2, column=0, columnspan=2) # 두 열을 합해 배치
33      bt1 = Button(lb, text='OK', command = checkid) # 버튼을 누르면 checked 함수 호출
34      bt2 = Button(lb, text='CANCEL')
35      bt1.grid(row=0, column=0, padx = 10) # 외부 여백에 10을 둠
36      bt2.grid(row=0, column=1, padx = 10) # 외부 여백에 10을 둠
37
38      win.mainloop()
```

결과

```
python power
로그인 성공
```



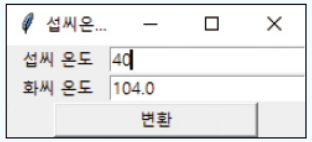
버튼 이벤트 처리로 섭씨 온도를 화씨 온도로 변환

10-6 코딩 | 10-06convertfah.py | 버튼 이벤트 처리로 섭씨 온도를 화씨 온도로 변환 | 난이도 응용

```
1   from tkinter import *
2
3   def fahrenhite(celsius):
4       return celsius * 9 / 5 + 32
5
6   def clicked():
7       degree = eval(cel.get()) # 엔트리 cel에서 값 가져오기
8       print(degree) # 디버깅을 위해 출력
9       fah.delete(0, END) # 이전의 자료를 비우고
10      fah.insert(0, str(fahrenhite(degree))) # 새로 변환된 자료를 쓰고
11
12  win = Tk()
13  win.title('섭씨 온도를 화씨 온도로 변환')
14
15  lbcel = Label(win, text="섭씨 온도", width=10)
16  lbfar = Label(win, text="화씨 온도")
17  lbcel.grid(row=0, column=0)
18  lbfar.grid(row=1, column=0)
19
20      cel = Entry(win, width=20)
21      fah = Entry(win)
22      cel.grid(row=0, column=1)
23      fah.grid(row=1, column=1)
24
25      cvt = Button(win, text="변환", width=20, command = clicked)
26      cvt.grid(row=2, column=0, columnspan=2)
27
28      win.mainloop()
```

결과

```
40
```



위젯의 메소드 바인드로 마우스 이벤트에 직선 그리기

- **캔버스(canvas)는 선, 다각형, 원 등을 그리기 위한 위젯으로 활용**
  - 키워드 인자로 relief: 테두리 모양으로 flat, groove, raised, ridge, solid, sunken
  - 외곽 테두리 두께(borderwidth)는 borderwidth 또는 bd에 지정
  - 함수 pack( )에서 확장 여부를 expand에 지정
  - 원도의 크기가 변함에 따라 채워질 방향을 가로 세로 양쪽인 fill='both'로 지정

```
canvas = Canvas(win, relief='solid', bd=2)
canvas.pack(expand=True, fill='both')
```

```
def click(event):
    global sX, sY
    print("클릭 위치", event.x, event.y)
    sX, sY = event.x, event.y
```

```
def release(event):
    global eX, eY
    print("릴리즈 위치", event.x, event.y)
    eX, eY = event.x, event.y
    # 직선 라인 그리기
    canvas.create_line(sX, sY, eX, eY, …)
```

```
# 왼쪽 마우스 버튼 클릭 바인딩
canvas.bind("<Button-1>", click)
# 왼쪽 마우스 버튼 릴리즈 바인딩
canvas.bind("<ButtonRelease-1>", release)
```

▲ 그림8 함수 bind()로 이벤트와 처리 함수 연결

캔버스에 직선 그리기

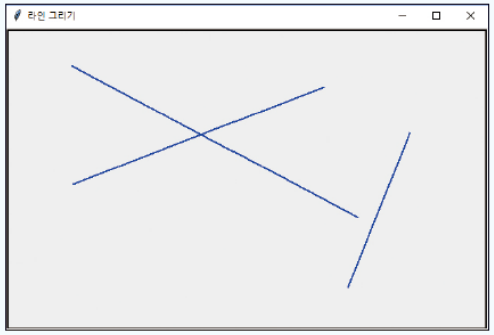
10-7 코딩 | 10-07canvasdrawline.py | 캔버스에 직선 그리기 | 난이도 응용

```
1   from tkinter import *
2
3   win = Tk()
4   win.title('라인 그리기')
5   win.geometry('640x400+100+100')
6
7   def click(event):
8       global sX, sY
9       print("클릭 위치", event.x, event.y)
10      sX, sY = event.x, event.y
11
12  def release(event):
13      global eX, eY
14      print("릴리즈 위치", event.x, event.y)
15      eX, eY = event.x, event.y
16      # 직선 라인 그리기
17      canvas.create_line(sX, sY, eX, eY, fill="blue", width=2)
18
19      canvas = Canvas(win, relief='solid', bd=2)
20      canvas.pack(expand=True, fill='both')
21
22      # 왼쪽 마우스 버튼 클릭 바인딩
23      canvas.bind("<Button-1>", click)
24      # 왼쪽 마우스 버튼 릴리즈 바인딩
25      canvas.bind("<ButtonRelease-1>", release)
26
27      win.mainloop()
```

geometry("너비x높이+x 좌표+y 좌표")를 의미하므로, 윈도 창의 너비와 높이, 초기 화면 좌측 상단 위치의 x 좌표와 y 좌표를 설정할 수 있다. 그러므로 윈도 크기가 640, 400이며, 윈도가 표시되는 위치를 (100, 100)으로 지정한다.

결과

```
클릭 위치 87 50
릴리즈 위치 467 251
클릭 위치 422 78
릴리즈 위치 89 207
클릭 위치 536 138
릴리즈 위치 453 344
```



이미지 파일을 캔버스에 생성

10-8 코딩 | 10-08imagecanvas.py | 이미지 파일을 캔버스에 그리기 | 난이도 기본

```
1   from tkinter import *
2
3   win = Tk()
4   win.title("그림 로드")
5   win.geometry("660x930")
6
7   # 캔버스 생성
8   canvas = Canvas(win, bg='Yellow')
9   # 캔버스를 윈도에 배치, 가로 세로로 확장되게
10  canvas.pack(expand=YES, fill=BOTH)
11
12  # 사진 생성
13  img = PhotoImage(file="imitation.png")
14
15  # 사진을 캔버스 위에 생성
16  canvas.create_image(10, 10, anchor=NW, image=img)
17
18  win.mainloop()
```

결과

이미지 파일을 레이블에 생성

- **레이블 생성 시 인자 image=img로 레이블에 지정한 이미지**
  - 위젯 PhotoImage로 이미지 파일 img를 생성

10-9 코딩 | 10-09imagelabel.py | 이미지 파일을 레이블에 그리기 | 난이도 기본

```
1   from tkinter import *
2   win = Tk()
3   win.title("레이블 그림 로드")
4
5   # 사진 생성
6   img = PhotoImage(file="imitation.png")
7   # 사진을 담은 레이블 생성
8   lbimg = Label(win, image=img)
9   # 캔버스를 윈도에 배치, 가로 세로로 확장되게
10  lbimg.pack()
11
12  win.mainloop()
```

결과

pygame 개요와 설치

- **pygame은 윈도 게임 파이썬 라이브러리로 원래 버전은 pyZine**
  - pygame 홈페이지(www.pygame.org)에서 매뉴얼 문서(www.pygame.org/docs)를 활용
- **pygame 설치**

▲ 그림10 파워 쉘에서 모듈 pygame 설치

Pygame 프로그램 구조

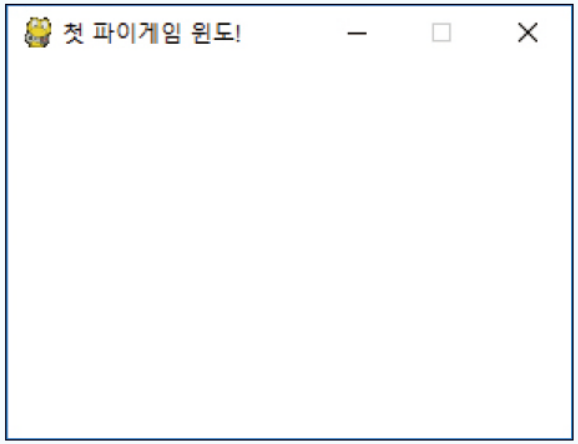
```
① pygame 불러오기
    import pygame
② pygame 초기화
    pygame.init()
③ pygame에서 사용할 주요 전역 변수 선언 및 윈도 설정
    screen = pygame.display.set_mode(size)
    pygame.display.set_caption('첫 파이게임 윈도!')
④ pygmae 메인 루프(while문)
    while not done:
        4-1. 이벤트 리스트에서 이벤트를 꺼내 이벤트 처리 준비
               for event in pygame.event.get():
        4-2. 화면의 색상을 바꾸는 등 필요한 프로그램 구현
               screen.fill(WHITE)  # 스크린 색상을 흰색으로 지정
               …                   # 화면에 여러 그림을 그림
        4-3. 화면에 그림을 그린 후 반드시 화면을 update() 처리
               pygame.display.update() # 화면의 필요한 부분만을 수정

⑤ pygmae 프로그램 종료
    pygame.quit()
```

첫 pygame 윈도

10-10 코딩 | 10-10firstpygame.py | 첫 pygame 윈도 | 난이도 기본

```
1   import pygame
2
3   # 색상 정의
4   WHITE = (255, 255, 255)
5   # 윈도 크기 정의
6   size = (300, 200)
7
8   # 윈도 초기화
9   pygame.init()
10
11  # 화면 크기 지정해 스크린을 생성
12  screen = pygame.display.set_mode(size)
13  # 제목인 캡션 지정
14  pygame.display.set_caption('첫 파이게임 윈도!')
15
16  # 윈도 종료 플래그로 사용되는 변수 초기값 지정
17  done = False
18  # 메인 루프
19  while not done:
20      for event in pygame.event.get():  # 여러 이벤트를 받아 처리
21          if event.type == pygame.QUIT:  # 윈도 종료 버튼을 누르면
22              done = True  # 프로그램을 종료하기 위해 True 지정
23
24      screen.fill(WHITE) # 스크린 색상을 흰색으로 지정
25      pygame.display.update() # 화면의 필요한 부분만을 수정
26
27  # 메인 루프를 빠져나오면 프로그램 종료
28  pygame.quit()
```

결과



화면에 글자 Hello, PyGame!을 그리는 프로그램

10-11 코딩 | 10-11hellopygame.py | pygame 윈도에 Hello, Pygame! 표시 | 난이도 기본

```
1   # %% 10-11hellopygame.py     pygame 윈도에 Hello, pygame! 표시
2   import pygame
3
4   # 색상 정의
5   WHITE = (255, 255, 255)
6   BLUE = (0, 0, 255)
7
8   pygame.init()
9
10  # 윈도 크기 지정
11  size = [300, 200]
12  screen = pygame.display.set_mode(size)
13  # 제목인 캡션 지정
14  pygame.display.set_caption("Hello, pygame!")
15
16  # 폰트 생성과 출력할 문자열 지정
17  font = pygame.font.SysFont('Arial', 20)
18  outstr = 'Hello, PyGame!'
19
20  # 메인 루프
21  done = False
22  while not done:
23      for event in pygame.event.get():
24          if event.type == pygame.QUIT:
25              done = True
26
27      screen.fill(WHITE)
28
29      # 지정된 문자열 글씨를 그린 화면을 반환해 text에 저장
30      text = font.render(outstr, True, BLUE)
31      # 글씨가 그려진 화면인 text를 윈도 스크린 위치 [x, y]에 그리기
32      screen.blit(text, [100, 80])
33
34      pygame.display.update()
35
36  pygame.quit()
```

결과

밤에 눈이 오는 모습 그리기(1)

- **패키지 pygame**
  - 게임을 위한 라이브러리로, 애니메이션 그림을 그리기에 적합
  - 좌표 (x, y)에 흰색의 원을 그리고
    - **이후 다시 좌표 (x, y+1)에 그리는 작업을 반복하면 눈이 내리는 듯한 애니메이션**
  - SNOW_CNT에 저장된 70개의 눈이 그려질 좌표를 snow_list 리스트로 저장

```
# 눈의 좌표 리스트
snow_list = []

# snow_list에 SNOW_CNT 수만큼의 좌표를 저장
for i in range(SNOW_CNT):
    x = random.randrange(0, SIZE[0]) # x 좌표 저장
    y = random.randrange(0, SIZE[1]) # y 좌표 저장
    snow_list.append([x, y]) # 리스트에 추가
```

- **메인 루프에서 display.update( )**
  - 위에서 작업한 내용이 화면에 그려짐
  - clock을 저장
    - **메인 루프에서 초당 화면을 몇 번 그리는지를 설정**
    - **초당 화면 표시 수인 FPS(Frame Per Second, 또는 Frame Rate)를 설정**
  - clock.tick(20)으로 초당 20 프레임 화면이 그려지도록 하면 눈 내리는 모습이 재현

밤에 눈이 오는 모습 그리기(2)

- **메인 루프 내부**
  - 좌표 리스트인 snow_lis
    - **좌표를 얻어 반지름 1인 원을 흰색으로 그림**
- **눈 snow_list[i]의 (x, y) 좌표**
  - y인 snow_list[i][1]을 1 증가시켜 눈이 내리도록 수정
  - x는 -1, 0, 1 중 하나를 임의로 선택해 저장
- **y가 윈도의 높이인 SIZE[1]보다 커져 바닥으로 내려가면**
  - 잠시 후 화면 상단에서 다시 보이도록 y인 snow_list[i][1]을 적정한 음수가 되도록 다시 지정
  - x인 snow_list[i][0]도 화면 가로 길이보다 적은 임의의 값으로 다시 저장

```
# 눈 내리는 모습 그리기
for i in range(len(snow_list)):

    # 눈 모양의 원 그리기
    radius = 1
    # radius = random.randint(1, 3)
    pygame.draw.circle(screen, WHITE, snow_list[i], radius)

    # 눈 snow_list[i]의 y 좌표를 1 증가시켜 눈이 내리도록 수정
    snow_list[i][1] += 1
    # snow_list[i][1] += random.randint(1, 3)
    # 눈 snow_list[i]의 x 좌표를 (-1, 0, 1) 중 하나에서 선택, 이전 값에 더해
      대입
    snow_list[i][0] += random.randint(-1, 1)

    # snow_list[i]가 윈도 바닥으로 내려가면, 즉 보이지 않게 되면
    if snow_list[i][1] > SIZE[1]: # snow_list[i][1]은 snow_list[i] 눈의 y 좌표
        # y 좌표를 위(음수)로 다시 지정
        snow_list[i][1] = random.randrange(-5, 0)
        # x 좌표도 다시 지정, snow_list[i][0]은 snow_list[i] 눈의 x 좌표
        snow_list[i][0] = random.randrange(0, SIZE[0])
```

밤에 눈이 오는 모습 그리기(3)

10-12 코딩 | 10-12snownight.py | 밤에 눈이 오는 모습 그리기 | 난이도 응용

```python
import pygame, random

# 게임 엔진 초기화
pygame.init()

BLACK = [0, 0, 0]
WHITE = [255, 255, 255]
SIZE = [300, 300] # 윈도 크기
SNOW_CNT = 70 # 눈의 개수

screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption("눈 오는 밤")

# 눈의 좌표 리스트
snow_list = []

# snow_list에 SNOW_CNT 수만큼의 좌표 저장
for i in range(SNOW_CNT):
    x = random.randrange(0, SIZE[0]) # x 좌표 저장
    y = random.randrange(0, SIZE[1]) # y 좌표 저장
    snow_list.append([x, y]) # 리스트에 추가
# 화면 수정에 사용될 시계 저장
clock = pygame.time.Clock()

# 메인 루프
done = False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # 배경색을 검은색으로
    screen.fill(BLACK)

    # 눈 내리는 모습 그리기
    for i in range(len(snow_list)):

        # 눈 모양의 원 그리기
        radius = 1
        # radius = random.randint(1, 3)
        pygame.draw.circle(screen, WHITE, snow_list[i], radius)

        # 눈 snow_list[i]의 y 좌표를 1 증가시켜 눈이 내리도록 수정
        snow_list[i][1] += 1
        # snow_list[i][1] += random.randint(1, 3)
        # 눈 snow_list[i]의 x 좌표를 (-1, 0, 1) 중 하나에서 선택, 이전 값에 더해 대입
        snow_list[i][0] += random.randint(-1, 1)

        # snow_list[i]가 윈도 바닥으로 내려가면, 즉 보이지 않게 되면
        if snow_list[i][1] > SIZE[1]: # snow_list[i][1]은 snow_list[i] 눈의 y 좌표
            # y 좌표를 위(음수)로 다시 지정
            snow_list[i][1] = random.randrange(-5, 0)
            # x 좌표도 다시 지정, snow_list[i][0]은 snow_list[i] 눈의 x 좌표
            snow_list[i][0] = random.randrange(0, SIZE[0])

    pygame.display.update()
    # 초당 수정될 프레임 수 지정, 초당 20 프레임 화면이 수정됨
    clock.tick(20)

pygame.quit()
```

결과