본 컨텐츠는 과학기술정보통신부 정보통신기획평가원의
SW중심대학지원사업단의 연구결과로 수행되었습니다.
(2019-0-01838)

과학기술정보통신부 IITP 정보통신기획평가원 배재대학교 PAI CHAI UNIVERSITY AI·SW중심대학사업단

파이썬 기초와 게임 만들기 - 5

Taejun Lee (marlrero@kakao.com, +82-10-5223-2912) Doctor's course
Paichai University, Dept. Computer Engineering, Lab. MIE

2023 동산고등학교-배재대학교 동아리 교육

CONTENTS

01

tkinter - 텍스트 입력 필드

1. 엔트리
2. 엔트리와 버튼

1-1. 엔트리

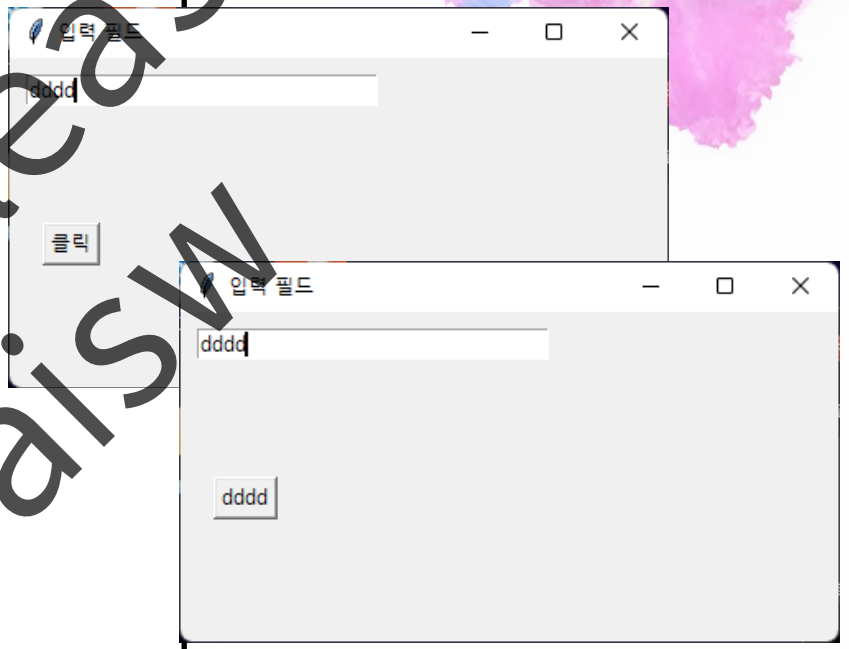
- Entry: 텍스트 입력 필드

```
import tkinter as tk

root = tk.Tk()
root.title("입력 필드")
root.geometry("400x200")

entry = tk.Entry(width=30)
entry.place(x=10, y=10)

root.mainloop()
```

1-2. 엔트리와 버튼

- 텍스트 입력 필드와 버튼 이벤트 활용

```
import tkinter as tk

def click_btn():
    txt = entry.get()
    btn["text"] = txt

root = tk.Tk()
root.title("입력 필드")
root.geometry("400x200")

entry = tk.Entry(width=30)
entry.place(x=10, y=10)

btn = tk.Button(text="클릭", command=click_btn)
btn.place(x=20, y=100)

root.mainloop()
```



이 밖에도 여러 개인 경우
Text(),
스크롤 형태는
ScrolledText() 등이 있음

02

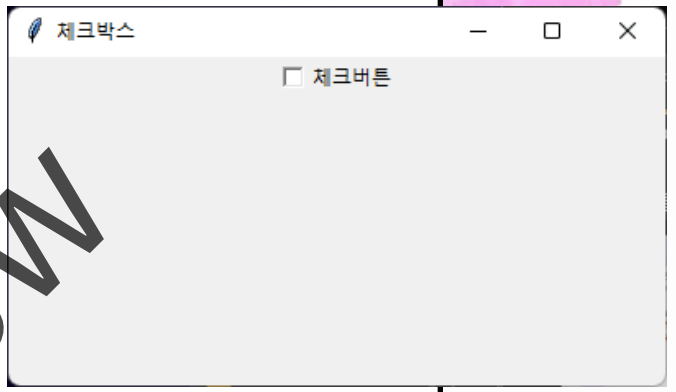
tkinter - 체크박스와 메시지박스

1. 체크박스
2. 메시지 박스
3. 다양한 메시지 박스

2-1. 체크박스

- 체크 박스(Checkbutton)과 체크 박스의 설정 여부 확인

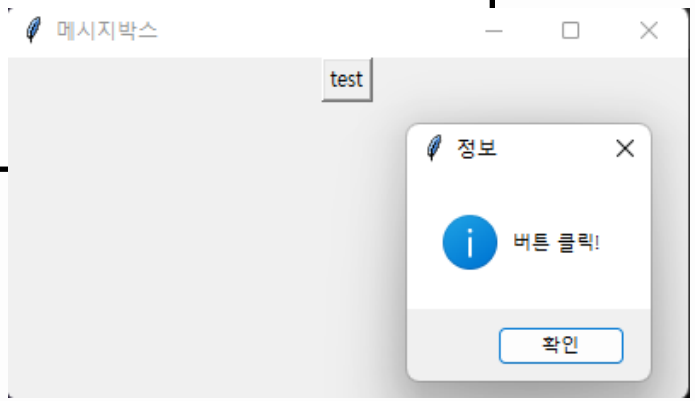
```
import tkinter as tk

def check():
    if check_val.get() == True:
        print("체크되어 있음")
    else:
        print("체크되어 있지 않음")

root = tk.Tk()
root.title("체크박스")
root.geometry("400x200")

check_val = tk.BooleanVar()
check_val.set(False)

check_btn = tk.Checkbutton(text="체크버튼", variable=check_val,
                           command=check)
check_btn.pack()

root.mainloop()
```



2-2. 메시지박스

- Message box 만들기

```
import tkinter as tk
import tkinter.messagebox

def click_btn():
    tkinter.messagebox.showinfo("정보", "버튼 클릭!")

root = tk.Tk()
root.title("메시지박스")
root.geometry("400x200")

btn = tk.Button(text="test", command=click_btn)
btn.pack()

root.mainloop()
```



2-3. 다양한 메시지박스

- showinfo() : 정보 표시 박스
- showwarning(): 경고 표시 박스
- showerror(): 에러 표시 박스
- askyesno(): 네, 아니오 버튼이 있는 박스
- askokcancle(): OK와 취소 버튼이 있는 박스

- 예제 1. 위에서 언급한 박스를 한 번 실행

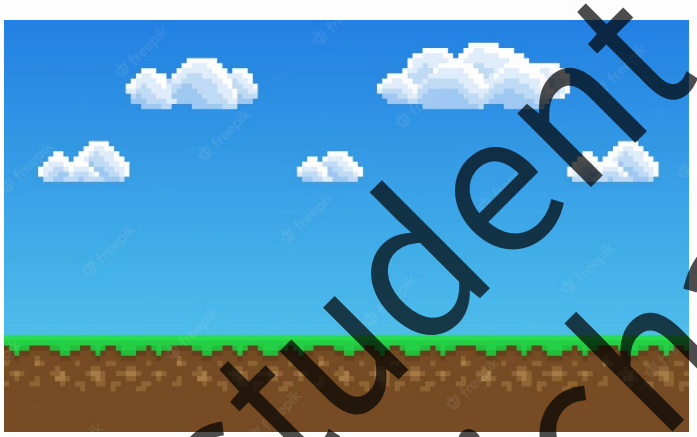
03

tkinter - 실시간 처리

1. 실시간 처리
2. after 함수

3-1. 실시간 처리

- 실시간(real time) 처리
  - 게임의 경우 시간과 함께 처리를 진행하는 경우가 많음
    - 액션 게임: 사용자가 아무것도 하지 않아도 적 캐릭터가 화면 위를 돌아다님
    - 배경의 구름이 흐려지거나 수면이 움직임
    - 제한 시간이 있는 게임의 경우 남은 시간이 줄어들기도 함



https://kr.freepik.com/premium-vector/pixel-art-game-background-grass-sky-and-clouds_9047947.htm

http://www.gamey.kr/news/articleView.html?idxno=3001175

3-2. after 함수

- 실시간 처리 수행 함수
- 숫자를 자동으로 새는 프로그램 만들기 (함수 실행 방식)


```
import tkinter

timer = 0  # 시간을 카운트 하는 변수

def count_up():
        global timer  # timer는 전역 변수
        timer += 1    # timer = timer + 1
        label["text"] = timer
        root.after(1000, count_up)  # 1초 뒤 다시 이 함수를 실행

root = tkinter.Tk()
label = tkinter.Label(font=("Time New Roman", 80))
label.pack()
root.after(1000, count_up)  # 첫 실행 지점(1초 후 지정 함수 실행)
root.mainloop()
```

1

2

3-2. after 함수

- 실시간 처리 수행 함수
- 숫자를 자동으로 새는 프로그램 만들기 (전역 변수 global)

```
import tkinter

timer = 0  # 시간을 카운트 하는 변수

def count_up():
        timer += 1    # timer = timer + 1
        label["text"] = timer
        root.after(1000, count_up) # 1초 뒤 다시 이 함수를 실행

root = tkinter.Tk()
label = tkinter.Label(font=("Time New Roman", 80))
label.pack()
root.after(1000, count_up)  # 첫 실행 지점(1초 후 지정 함수 실행)
root.mainloop()
```

timer 변수는 지역 변수가 됨
timer 변수는 함수가 실행될 때마다 초기화 됨
timer가 무엇으로 초기화되어 있는지가 명시되지 않음

3-2. after 함수

- 실시간 처리 수행 함수
- 숫자를 자동으로 새는 프로그램 만들기 (전역 변수 global)

```
import tkinter

timer = 0  # 시간을 카운트 하는 변수

def count_up():
        global timer  # timer는 전역 변수
        timer += 1    # timer = timer + 1
        label["text"] = timer
        root.after(1000, count_up)  # 1초 뒤 다시 이 함수를 실행

root = tkinter.Tk()
label = tkinter.Label(font=("Time New Roman", 80))
label.pack()
root.after(1000, count_up)  # 첫 실행 지점(1초 후 지정 함수 실행)
root.mainloop()
```

전역(global) 변수

timer라는 변수는 전역(global) 변수야!

04

키/마우스 입력

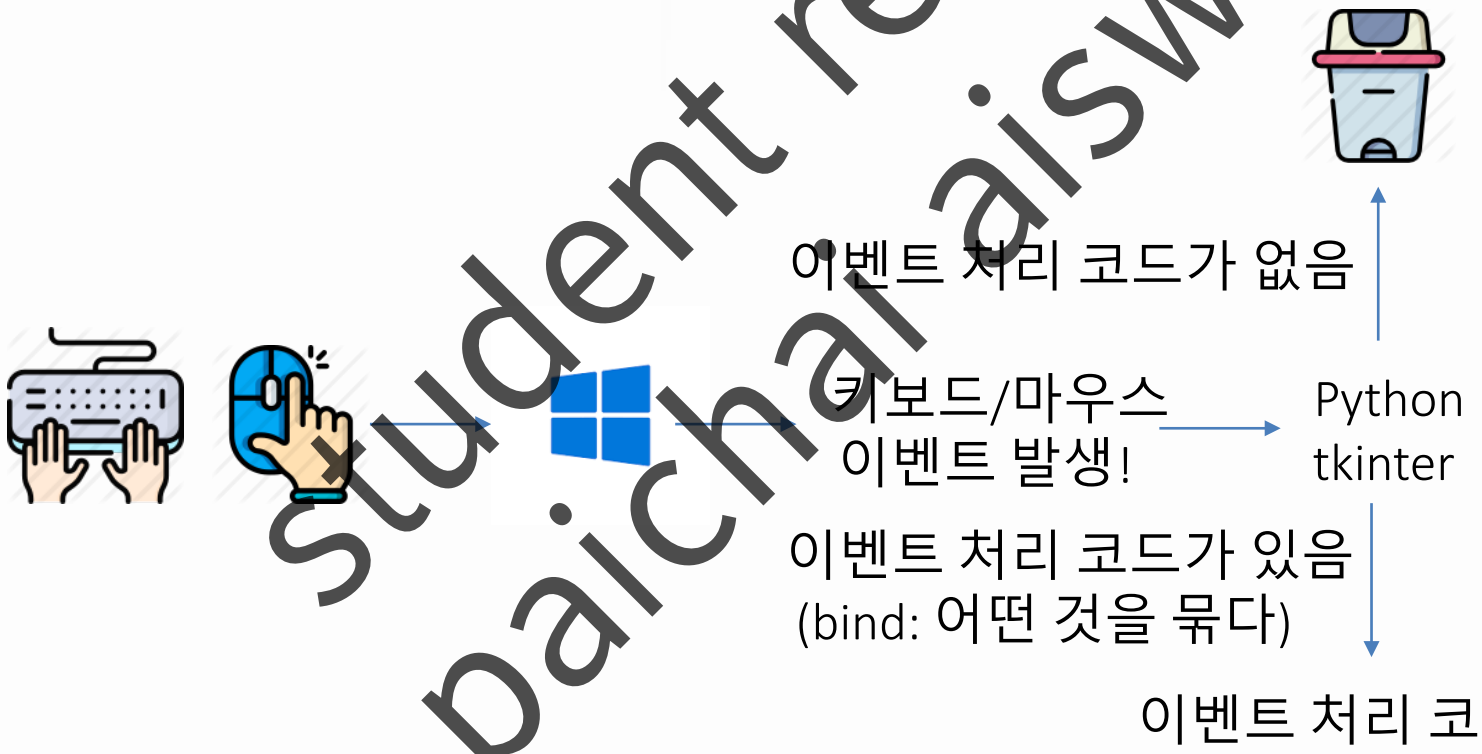
1. 이벤트
2. bind 함수
3. bind 함수와 after 함수
4. 키 코드와 키 심볼
5. 오리 움직이기 게임
6. 마우스 입력과 bind 함수

4-1. 이벤트

- Event
  - 사용자(user)가 소프트웨어(software)한테
    키보드나 마우스로 명령해서 이를 처리하는 것을 말함
  - tkinter의 윈도우에 버튼을 눌렀을 때
    → 버튼에 대해 마우스 이벤트 발생!
  - tkinter의 윈도우가 선택됐을 때 엔터 키를 눌렀을 때
    → 키보드 이벤트 발생
  - GUI(Graphic User Interface)는 이벤트 처리가 핵심
    → 마우스와 키보드를 누르는 일이 굉장히 많음
  - 심화: 그러면 그 이벤트는 누가 만들어서 우리에게 줄까요?
    예) 남/여자친구를 위한 이벤트는 누가 만드나요?

4-2. bind 함수

- tkinter에서 이벤트를 받을 때 사용
  - 이벤트는 운영체제(Operating System)가 사용자에게 키보드나 마우스 조작을 받아서 tkinter에게 넘겨줌
  - tkinter에게 넘겨준 이벤트를 처리하지 않으면 이 이벤트는 무시됨(버려짐)
  - 전문 용어로 이벤트 바인딩(event binding), 이벤트 핸들링(event handling)이라고 함



4-2. bind 함수

- bind 함수 예제

```
import tkinter

key = 0  # 키보드가 어떤 키가 입력됐는지 그 코드 값 저장
def key_down(e):
    global key
    key = e.keycode
    print("Key input: " + str(key))

root = tkinter.Tk()
root.title("키보드 입력")
root.bind("<KeyPress>", key_down)
    # 키가 눌리면 key_down 함수를 실행하라
root.mainloop()
```

4-2. bind 함수

- bind 함수 사용 방법
  - bind() 함수 사용 방법: bind("<이벤트>", 이벤트 발생 시 함수 이름)
    - "<이벤트>"는 아래 주요 이벤트와 같음

<이벤트>	설명
<KeyPress> or <Key>	키를 눌렀을 때
<KeyRelease>	키를 눌렀다가 땠을 때
<Motion>	마우스 포인터가 움직일 때
<ButtonPress> or <Button>	마우스 버튼을 클릭할 때

    - 이벤트 발생 시 함수는 프로그래머가 정의
      → 이 함수를 줄여서 전문 용어로 이벤트 핸들러(event handler)라고도 함

4-3. bind 함수와 after 함수

- bind() 함수와 실시간 처리(after() 함수) 같이 써 보기

```
import tkinter

key = 0  # 키보드 코드 값(어떤 것을 눌렀는지)
def key_down(e):
    global key
    key = e.keycode

def main_action():
    label["text"] = key
    root.after(100, main_action) # 0.1초 후에 이 함수 다시 실행

root = tkinter.Tk()
root.title("실시간으로 키 입력받기")
label = tkinter.Label(font=("Time New Roman", 50))
label.pack()

root.bind("<Key>", key_down)
main_action()  # 처음 실시간 처리 시작 지점
root.mainloop()
```

4-4. 키 코드와 키 심볼

- 키 코드(Key code): 어떤 키인지를 숫자로 표현함
  - Windows

키보드 키	키 코드
방향키 ←↑→↓	37, 38, 39, 40
스페이스 바	32
엔터	13
알파벳 A ~ Z	65 ~ 90
숫자 0 ~ 9	48 ~ 57

  - Windows, macOS와 키 코드가 다름 (방향키, 엔터, 알파벳)

4-4. 키 코드와 키 심볼

- 키 코드는 숫자라 직관적이지 않으며 운영체제마다 다름
  → keysym (key symbol)

```
import tkinter

key = ""  # 키 코드가 아닌 어떤 키가 눌렸는지에 관한 문자열
def key_down(e):
    global key
    key = e.keysym

def main_action():
    label["text"] = key
    root.after(100, main_action) # 0.1초 후에 이 함수 다시 실행

... 이전 코드(PPT 21페이지)와 같음 ...
```

4-5. 오리 움직이기 게임

- 1단계: 윈도우와 캔버스, 이미지 붙이기
  - create_image(이미지의 중심 x좌표, y좌표, image="이미지경로", tag="DUCK")
  - 이미지는 깃허브에 있음:
    https://github.com/Marlrero/Lecture_game_example/tree/master/tkinter_primary_adv

```
import tkinter

duck_x = 400  # 오리의 x좌표
duck_y = 300  # 오리의 y좌표

root = tkinter.Tk()
root.title("오리 움직이기 게임")
canvas = tkinter.Canvas(width=800, height=600, bg="skyblue")
canvas.pack()

img = tkinter.PhotoImage(file="duck.png")
canvas.create_image(duck_x, duck_y, image=img, tag="DUCK")

root.mainloop()
```

4-5. 오리 움직이기 게임

- 2단계: 오리 이미지 크기 설정(resize)
  - pip install pillow (터미널에서 실행해야 함)

```
from PIL import Image, ImageTk   # pip install pillow

... 생략 ...
canvas.pack()

duck_img = Image.open("duck.png")
resized_img = duck_img.resize((50, 50))
img = ImageTk.PhotoImage(resized_img)
canvas.create_image(duck_x, duck_y, image=img, tag="DUCK")

root.mainloop()
```

4-5. 오리 움직이기 게임

- 3단계: 키 이벤트 추가

```
import tkinter
from PIL import Image, ImageTk  # 이미지 리사이즈 (pip install pillow)

key = ""
def key_down(e):  # 키를 눌렀을 때
    global key
    key = e.keysym

def key_up(e):    # 키를 눌렀다가 땠을 때
    global key
    key = ""

duck_x = 400  # 오리의 x좌표
duck_y = 300  # 오리의 y좌표
... 생략 ...
```

4-5. 오리 움직이기 게임

- 3단계: 키 이벤트 추가

```
... 생략 ...
duck_x = 400  # 오리의 x좌표
duck_y = 300  # 오리의 y좌표

def main_action():
    global duck_x, duck_y
    if key == "Up":  # 위 방향키를 누르면
        duck_y -= 20  # y좌표 20픽셀 감소(위로 이동)
    if key == "Down": # 아래 방향키를 누르면
        duck_y += 20  # y좌표 20픽셀 증가(아래로 이동)
    if key == "Left":  # 왼쪽 방향키를 누르면
        duck_x -= 20  # x좌표 20픽셀 감소(왼쪽으로 이동)
    if key == "Right": # 오른쪽 방향키를 누르면
        duck_x += 20  # x좌표 20픽셀 증가(오른쪽으로 이동)
    canvas.coords("DUCK", duck_x, duck_y)  # 이미지 새 위치로
    root.after(100, main_action) # 0.1초후 다시 실행
... 생략 ...
```

4-5. 오리 움직이기 게임

- 3단계: 키 이벤트 추가

```
... 생략 ...

root.bind("<KeyPress>", key_down)
root.bind("<KeyRelease>", key_up)
main_action()
root.mainloop()
```

4-6. 마우스 입력과 bind 함수

- 마우스도 마찬가지로 bind 함수 사용
- 마우스 이벤트를 처리하기 위한 함수 만들기

```
import tkinter

mouse_x = 0  # 마우스 포인터 x좌표
mouse_y = 0  # 마우스 포인터 y좌표
mouse_c = 0  # 마우스 포인터 클릭 여부(flag)

def mouse_move(e):  # 마우스 포인터 이동 시
    global mouse_x, mouse_y
    mouse_x = e.x
    mouse_y = e.y

def mouse_press(e): # 마우스 버튼 클릭 시
    global mouse_c
    mouse_c = 1

def mouse_release(e): # 마우스 버튼 클릭 후 해제 시
    global mouse_c
    mouse_c = 0
```

4-6. 마우스 입력과 bind 함수

- 마우스의 x좌표와 y좌표, 눌렀는지 여부(플래그) 표시

```
def main():
    _font = ("Time New Roman", 20)
    txt = f"mouse({mouse_x}, {mouse_y}) {mouse_c}"
    cvs.delete("TEST")  # 캔버스의 텍스트 태그 TEST를 지우기
    cvs.create_text(456, 384, text=txt, fill="black", font=_font, \
                    tag="TEST")
    root.after(100, main)  # 0.1초 후 이 함수 재실행
```

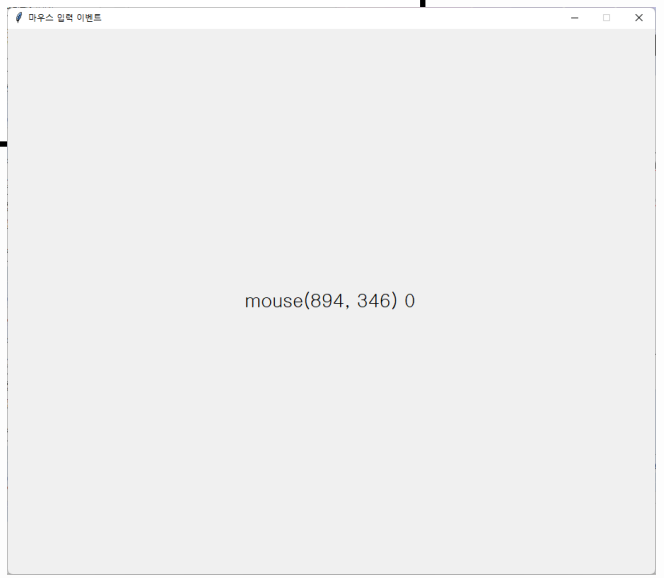
- root 윈도우 만들기

```
root = tkinter.Tk()
root.title("마우스 입력 이벤트")
root.resizable(False, False)  # 윈도우 크기 변경 불가
```

4-6. 마우스 입력과 bind 함수

- 마우스 이벤트 등록을 위한 bind 함수 사용하기

```
root.bind("<Motion>", mouse_move)
   # 마우스 포인터 이동 시 이벤트 등록
root.bind("<ButtonPress>", mouse_press)
   # 마우스 버튼 클릭 시 이벤트 등록
root.bind("<ButtonRelease>", mouse_release)
   # 마우스 버튼 클릭 후 해제 시 이벤트 등록

cvs = tkinter.Canvas(root, width=912, height=768)
cvs.pack()
main() # 실시간 처리 after 함수 시작점
root.mainloop()
```



05

미로 게임

1. 배열과 화면 구성
2. 이미지 추가
3. 키보드 이벤트 추가
4. 실시간 움직임 처리
5. 클리어 판정
6. 다시 시작 추가
7. 개선점

06

고양이 게임

1. 계획과 리소스 준비
2. 게임 화면 구성
3. 커서와 마우스 이벤트
4. 고양이 위치 관리
5. 낙하 알고리즘 구현
6. 판정 알고리즘 구현
7. 타이틀과 게임 오버 화면 구현

참고 자료

- 윤성우, 열혈 파이썬(기본편/중급편), 오렌지미디어, 2017.
- 히로세 츠요시, 파이썬으로 배우는 게임 개발 (입문편/실전편), 제이펍, 2020.
- 폴 데이텔, 하비 데이텔, 안진섭, 프로그래머를 위한 Python, 성안당, 2021.
- 루시아누 하말류, 전문가를 위한 파이썬, O'REILLY, 2016.
- 브렛 슬라킨, Effective Python, 2nd edition, 길벗, 2020.