

# tkinter의 사용법

## 1. tkinter란?

tkinter는 python의 표준 GUI 패키지이며, 이것은 명령이 실행되는 윈도우(창)를 만들 수 있게 해준다.

(GUI는 Graphic User Interface의 약자로, 입출력 등의 텍스트적 기능을 그래픽으로 나타낸 것이다.)

## 2. tkinter의 사용

기본 tkinter 코드의 형식

```
① import tkinter

② 윈도우이름 = tkinter.Tk()
   윈도우이름.title("제목")
   윈도우이름.geometry("너비x높이+x좌표+y좌표")
   윈도우이름.resizable(상하, 좌우)

③ # 이 사이에 위젯들을 추가한다.

④ 윈도우이름.mainloop()
```

일반적으로 tkinter를 사용 할 때는

①에서 tkinter를 import하고, ②에서 윈도우 기본 설정을 하며,  
③에서 위젯들을 추가한 다음에 ④를 선언하면서 코드를 종결시킨다.

### 2-① tkinter 가져오기

tkinter는 python에 내장된 GUI 모듈이기 때문에, 별다른 설치 필요 없이


`import tkinter`를 통해 tkinter 모듈을 불러올 수 있고,

`from tkinter import` 불러올 함수를 통해 tkinter의 함수만을 불러오는 것도 가능하다.

## 2-② 윈도우의 기본 설정

`윈도우이름 = tkinter.Tk()`를 통해 기본 배경이 될 창을 생성할 수 있으며.

다음의 목록을 통해 윈도우 창의 속성을 바꿀 수 있다.

사용 방법	설명
<code>윈도우이름.title("제목")</code>	윈도우 창의 제목  제목 을 설정할 수 있다.
<code>윈도우이름.geometry("너비x높이+x좌표+y좌표")</code>	너비, 높이 값을 통해 윈도우의 크기를, x좌표, y좌표 값을 통해 초기화면의 위치를 설정할 수 있다.
<code>윈도우이름.resizable(상하, 좌우)</code>	0, 1 또는 true, false 값을 통하여 윈도우의 상하좌우 크기 조절을 가능 혹은 불가능하게 설정할 수 있다.

## 2-③ tkinter의 위젯

tkinter 안의 위젯들은 각각의 기능들을 가지며, 종류와 기능은 다음과 같다.

위젯	설명
Button	단순한 버튼
Label	텍스트 혹은 이미지 표시
CheckButton	체크박스
Entry	단순한 한 라인 텍스트 박스
ListBox	리스트 박스
RadioButton	옵션 버튼
Message	Label과 비슷하게 텍스트 표시. Label과 달리 자동 래핑 기능이 있다.
Scale	슬라이스 바
Scrollbar	스크롤 바
Text	멀티 라인 텍스트 박스로서 일부 Rich Text 기능 제공
Menu	메뉴 Pane
Menubutton	메뉴 버튼
Toplevel	새 윈도우를 생성할 때 사용. Tk()는 윈도우를 자동으로 생성하지만, 추가로 새 윈도우 혹은 다이얼로그를 만들 때 Toplevel를 사용한다.
Frame	컨테이너 위젯. 다른 위젯들을 그룹화할 때 사용
Canvas	그래프와 점들로 그림을 그릴 수 있으며, 커스텀 위젯을 만드는데 사용될 수도 있다

## 2-③-(1). 위젯 사용하기

1). `frame label = tkinter.Frame(painting_window, width = 400, height = 100)`

`위젯 이름 = tkinter.사용할위젯(배치할 창의 이름, 파라미터1, 파라미터2...)`

의 형식으로 사용할 수 있으며, 위젯 이름은 어떤 문자가 와도 상관없다.

2). 사용할 위젯의 첫 글자는 대문자로 적는다.

3). 괄호 바로 다음에는 `painting_window`처럼 위젯을 배치할 창을 설정한 후, 각 위젯의 파라미터 값을 조절하여 각 위젯의 속성을 조절할 수 있다.

## 2-③-(2). 위젯 배치하기

위젯을 생성하였기에, 위젯의 위치를 정하는 위젯 배치가 필요하다.

1). 위젯은 `.pack()`, `.place()`, `.grid()` 메서드를 사용하며

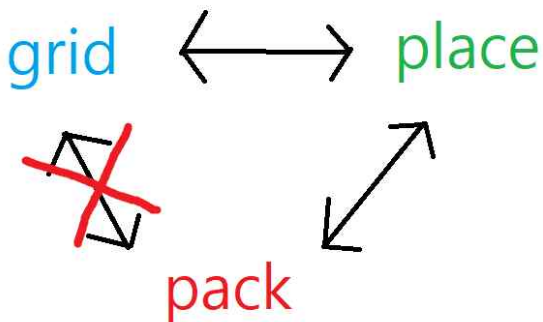
`button1.pack()` 혹은

배치 메서드를 하나만 사용할 경우 `button1 = ().pack()` 처럼

위젯을 생성하면서 배치를 할 수도 있다.

방식	설명
<code>grid</code>	셀 단위, 행과 열 단위로 배치한다.
<code>place</code>	위치를 절대 좌표값으로 배치한다(X값, Y값).
<code>pack</code>	위치를 상대 위치로 배치한다.

`grid`와 `pack`은 같이 사용 불가능하고, `place()`은 `pack()`, `grid()`와는 같이 사용할 수 있다.



2). 모든 배치의 순서는 선언한 순서에 따라 배치된다,

3). 위젯과 마찬가지로 parameter 값으로 속성을 변화할 수 있다.

## pack Parameter

이름	의미	기본값	속성
side	특정 위치로 공간 할당	top	top, bottom, left, right
anchor	할당된 공간 내에서 위치 지정	center	center, n, e, s, w, ne, nw, se, sw
fill	할당된 공간에 대한 크기 맞춤	none	none, x, y, both
expand	미사용 공간 확보	False	Boolean
ipadx	위젯에 대한 x 방향 내부 패딩	0	상수
ipady	위젯에 대한 y 방향 내부 패딩	0	상수
padx	위젯에 대한 x 방향 외부 패딩	0	상수
pady	위젯에 대한 y 방향 외부 패딩	0	상수

## grid Parameter

이름	의미	기본값	속성
row	행 위치	0	상수
column	열 위치	0	상수
rowspan	행 위치 조정	1	상수
columnspan	열 위치 조정	1	상수
sticky	할당된 공간 내에서의 위치 조정	-	n, e, s, w, nw, ne, sw, se
ipadx	위젯에 대한 x 방향 내부 패딩	0	상수
ipady	위젯에 대한 y 방향 내부 패딩	0	상수
padx	위젯에 대한 x 방향 외부 패딩	0	상수
pady	위젯에 대한 y 방향 외부 패딩	0	상수

## place Parameter

이름	의미	기본값	속성
x	x좌표 배치	0	상수
y	y좌표 배치	0	상수
relx	x좌표 배치 비율	0	0 ~ 1
rely	y좌표 배치 비율	0	0 ~ 1
width	위젯의 너비	0	상수
height	위젯의 높이	0	상수
relwidth	위젯의 너비 비율	0	0 ~ 1
relheight	위젯의 높이 비율	0	0 ~ 1
anchor	위젯의 기준 위치	nw	n, e, w, s, ne, nw, se, sw

## 2-③-(3). command 작성하기

1). 버튼의 파라미터값을 보면 **command**라는 값이 있다.

```
button1 = tkinter.Button(frame_button, command = red_color).pack()
```

위의 코드는 button1이라는 이름의 버튼이 눌렸을 경우, **command**라는 속성을 통해 **red\_color**라는 함수를 실행시킨다.

```
def red_color():  
    global color  
    color = "red"
```

**red\_color**함수는 **color**이라는 전역변수의 값을 "red"로 바꾸는 함수이다.

따라서, button1을 누르면 **red\_color**함수가 실행되어 **color**라는 전역변수의 값이 "red"로 바뀌게 되는 것이다.

이처럼 **command**라는 값은 버튼이 눌러졌을 경우 실행될 메소드(함수)를 설정할 수 있다.

(\* **command**에 쓸 함수는 **command**를 쓰는 위젯보다 먼저 생성해야 한다.)

## 2-④. 코드 종결하기

**윈도우이름.mainloop()**을 통하여 우리는 우리가 작성한 **윈도우이름**의 윈도우 창을 윈도우가 종료 될 때까지 실행 시킬수 있다.

```
① import tkinter  
  
② 윈도우이름 = tkinter.Tk()  
   윈도우이름.title("제목")  
   윈도우이름.geometry("너비x높이+x좌표+y좌표")  
   윈도우이름.resizable(상하, 좌우)  
  
③ # 이 사이에 위젯들을 추가한다.  
  
④ 윈도우이름.mainloop()
```

위부분에도 언급했듯이, 이 코드 종결문은 ④에 위치하도록 가장 마지막에 작성한다.

이것으로 기본 사용법은 끝이다.

### 3. 더보기

① 각 위젯들의 파라미터값과 더욱 자세한 사용법은

<https://076923.github.io/posts/#python-tkinter>에 각 위젯마다 적혀있다.

레퍼런스:

<https://docs.python.org/3/library/tk.html>

[https://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create\\_oval.html](https://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create_oval.html)(접속불가)

② 만약 2개의 파이썬 파일에서 서로서로 불러와 윈도우를 띄우고 싶을 때  
**A**라는 파이썬 파일과, **B**라는 파이썬 파일과 **B**안에 **C**라는 함수가 있다 가정하자.

**A**파일에서 import **B**를 사용하면 **A**를 실행 시켰을 때 **B**의 코드를 쓸수 있다.

(**A**-> **B** 가능)

그러나 다시 **B**에서 **A**를 쓰기위해 import **A**를 하면 오류가 나버린다

(**B**->**A** 불가능)

이처럼 import의 경우는 서로서로 파일을 가져오기에는 무리가 있는데,  
이를 사용 하고 싶으면

```
① import tkinter

② 윈도우이름 = tkinter.Tk()
   윈도우이름.title("제목")
   윈도우이름.geometry("너비x높이+x좌표+y좌표")
   윈도우이름.resizable(상하, 좌우)

③ # 이 사이에 위젯들을 추가한다.

④ 윈도우이름.mainloop()
```

에서 ②~④부분을 **하나의 함수**로 묶고,

1번째 줄에서 "import 다른파일명"의 형식보다는 함수를 통하여

```
def 함수이름
    from B import C
    C()
(def 함수이름
    from 다른파일명 import 다른파일의 함수이름
    다른파일의 함수이름() )
```

의 형식을 사용하여 다른파일의 윈도우를 띄울수 있는 함수를 만들어 사용하면 된다.

```
def return_menu():
    from painter_menu import painter_menuing
    painting_window.destroy()
    painter_menuing()
```

위는 return\_menu라는 함수를 만든 것이다.

이 함수는 painter\_menu라는 파일의 painter\_menuing이라는 함수를 불러오고, painter\_menuing을 실행한다.

③ 그리고 `painting_window.destroy()`는

`윈도우이름.destroy()`처럼 사용 가능하며,

이 코드는 윈도우이름(painting\_window)의 윈도우를 종료시키는 코드이다.

#### ④ 패딩?

grid, pack의 파라미터 값을 보면 padx, ipadx 등등 pad가 들어간 파라미터가 보인다.  
여기서 pad는 그림과 같이 선과 콘텐츠(위젯) 사이의 간격을 조절 가능하다.



(이 그림은 CSS에서 가져온 그림인데,  
padding의 개념은 같다.)

기본 윈도우		pad 값을 설정 하지 않았기에 기본값인 0이 들어가 기본윈도우 형태가 저런 형태로 나온다.
padx		x축(좌우)으로 간격이 생겼다.
pady		y축(상하)으로 간격이 생겼다.
inpadx		안쪽에서 바깥으로 x축(좌우) 간격이 생겼다. (이해를 위해 Green Grass 라벨에만 inpadx를 적용했다.)

이렇듯 padx, inpadx 등은 외부 혹은 내부에서의 간격을 조절 가능하게 해준다.