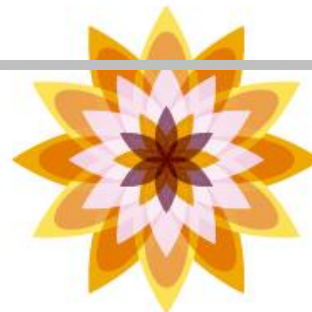
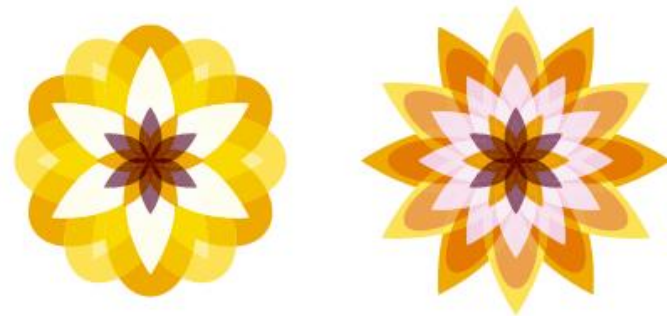
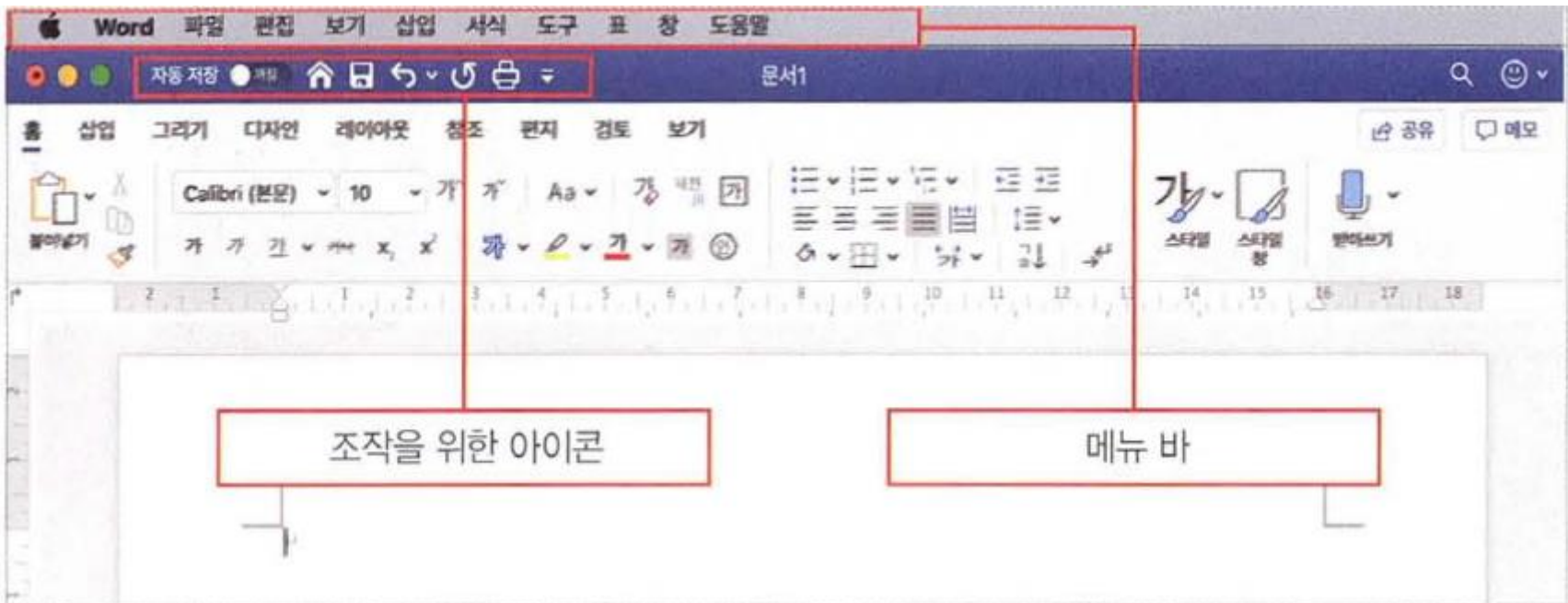


Chapter 05
GUI 기초(1)



1. GUI란?

- PC의 문서 작성 소프트웨어나 인터넷에 접속하는 브라우저에는 메뉴 바에 'File'이나 'Help' 등의 문자가 배치되어 있습니다.
- 또는 문서 작성 소프트웨어에는 파일을 저장하는 아이콘, 브라우저에는 페이지 새로고침 아이콘 등이 표시되어 있습니다.
- 사용자는 해당 아이콘을 클릭해 필요한 조작을 수행할 수 있습니다.



1. GUI란?

- 이처럼 소프트웨어의 조작 방법을 직관적으로 알 수 있는 인터페이스가 GUI입니다.
- 아이콘 이미지가 필요할 뿐만 아니라 예를 들면, '개수'라고 표시된 텍스트 옆에 사각형 입력 필드가 있다면 해당 필드에 숫자를 입력해야 함을 쉽게 알 수 있습니다.
- 또한, 버튼이 표시되어 있다면 버튼을 눌러야 한다는 것도 알 수 있습니다.
- GUI란 문자나 숫자 입력 필드, 혹은 버튼 등을 포함한 것을 의미합니다.

1. GUI란?

■ 윈도우 표시

- 파이썬에서 GUI를 다루기 위해서는 tkinter 모듈을 사용합니다.
- 우선 화면에 윈도우를 표시합니다.
- 다음 프로그램을 입력하고 파일 이름을 붙여 저장한 뒤 실행해 봅니다.
- 예제 ex0501_1.py

```
1 import tkinter
```

```
2 root = tkinter.Tk( )
```

```
3 root.mainloop( )
```

tkinter 모듈 импорт

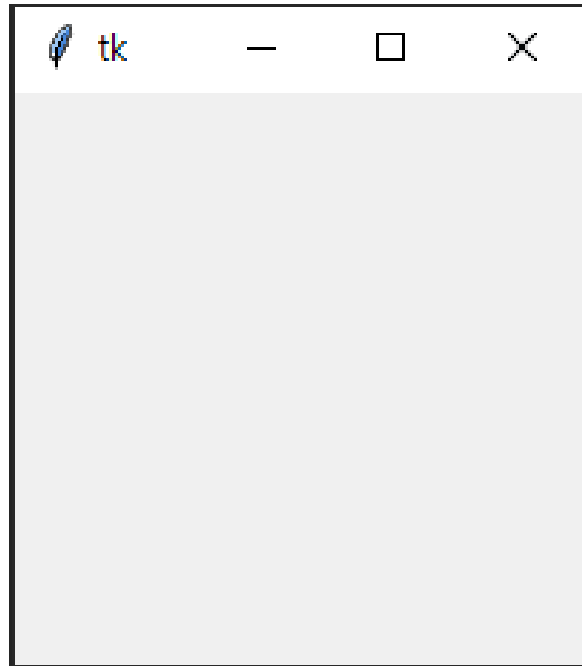
윈도우 요소(객체) 생성

윈도우 표시

1. GUI란?

■ 윈도우 표시

- 이 프로그램을 실행하면 그림과 같은 윈도우가 표시됩니다.
- 2번 행 `'root = tkinter.Tk()'`로 윈도우 요소(객체)를 만듭니다.
- 이 요소를 객체라고 합니다.
- 이 프로그램에서는 `root`라는 변수가 윈도우 객체입니다.
- 객체를 만들었다고 해서 곧바로 화면에 표시되는 것은 아닙니다.
- 3번 행과 같이 `mainloop()` 명령을 사용해 화면에 표시합니다.



1. GUI란?

■ 제목과 크기 지정하기

- 다음으로 윈도우 제목과 크기를 설정합니다.
- 제목은 `title()` 명령, 크기는 `geometry()` 명령으로 지정합니다.
- 다음 프로그램을 입력하고 파일 이름을 붙여 저장한 뒤 실행해 봅니다.
- 예제 `ex0501_2.py`

1	<code>import tkinter</code>	tkinter 모듈 импорт
2	<code>root = tkinter.Tk()</code>	윈도우 객체 생성
3	<code>root.title("첫 번째 윈도우")</code>	윈도우 제목 지정
4	<code>root.geometry("800x600")</code>	윈도우 크기 지정
5	<code>root.mainloop()</code>	윈도우 표시

- `title()`의 인수로 윈도우 제목을 지정합니다.
- 윈도우의 폭과 높이는 `geometry()`의 인수로 '폭 × 높이'를 입력해 지정합니다.
- x는 영소문자로 입력합니다.
- 윈도우 크기는 `geometry()` 외에 `minsize(폭, 높이)`로 최소 크기 `maxsize(폭, 높이)`로 최대 크기를 지정할 수 있습니다.

1. GUI란?

■ 제목과 크기 지정하기

- 이 프로그램을 실행하면 지정한 크기의 윈도우에 제목이 표시됩니다.



2. 라벨 배치하기

■ 라벨 배치

- 라벨은 `Label()` 명령으로 만들고 `place()` 명령으로 배치합니다.
- 다음 프로그램을 입력하고 파일 이름을 붙여 저장한 뒤 실행해 봅니다.
- 예제 `ex0502_1.py` (라벨 생성과 배치 관련 내용은 굵게 표기)

1	<code>import tkinter</code>	tkinter 모듈 импорт
2	<code>root = tkinter.Tk()</code>	윈도우 객체 생성
3	<code>root.title("첫 번째 라벨")</code>	윈도우 제목 지정
4	<code>root.geometry("800x600")</code>	윈도우 크기 지정
5	<code>label = tkinter.Label(root, text="라벨 문자열", font=("System", 24))</code>	라벨 컴포넌트 생성
6	<code>label.place(x=200, y=100)</code>	윈도우에 라벨 배치
7	<code>root.mainloop()</code>	윈도우 표시

2. 라벨 배치하기

■ 라벨 배치

- 예제 ex0502_1.py (라벨 생성과 배치 관련 내용은 굵게 표기)
- 5 ~ 6번 행에서 라벨을 만들고 배치합니다.
- 그 식은 다음과 같습니다.

```
■ 라벨 변수명 = tkinter.Label(윈도우 객체, text="라벨 문자열", font=("폰트 명", 폰트 크기))  
■ {라벨 변수명}.place(x=X 좌표, y=Y 좌표)
```

- 폰트 명에는 파이썬에서 사용 가능한 폰트를 지정하는데 PC에 따라 사용할 수 있는 폰트 종류가 다릅니다.
- 이 프로그램을 실행하면 다음과 같이 윈도우 안에 라벨이 표시됩니다.

첫 번째 라벨

— □ ×

라벨 문자열

2. 라벨 배치하기

■ 사용 가능한 폰트 확인

- 사용 가능한 폰트의 종류를 확인하기 위해서는 'tkinter.font.families()' 값을 print() 명령으로 출력합니다.
- 다음 프로그램을 입력하고 파일 이름을 붙여 저장한 뒤 실행해 봅니다.
- 예제 ex0502_2.py

1	import tkinter	tkinter 모듈 импорт
2	import tkinter.font	tkinter.font импорт
3	root = tkinter.Tk()	
4	print(tkinter.font.families())	tk.font.families() 값 출력

- 이 프로그램을 실행하면 셸 윈도우에 사용 가능한 폰트 목록이 출력됩니다.
- 만일 여러분이 사용하는 윈도우 PC에는 궁서체가 있다면, 'font = ("궁서체 " 24)'를 입력하면 해당 폰트를 사용할 수 있습니다.

2. 라벨 배치하기

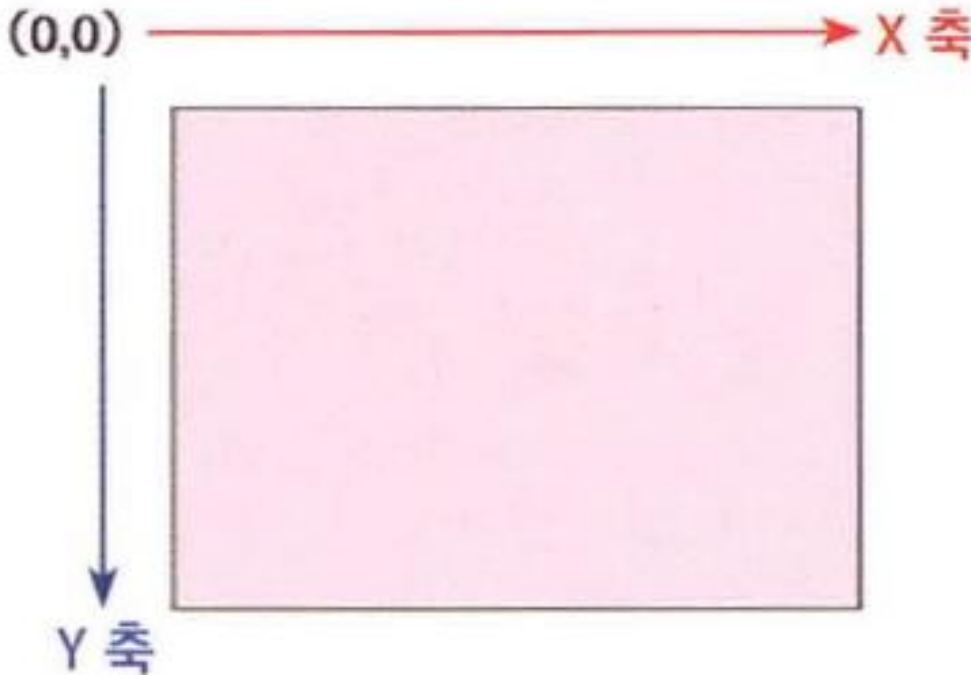
■ 사용 가능한 폰트 확인

- 사용 가능한 폰트는 PC에 따라 다릅니다.
- 따라서 인터넷에서 일반에 공개할 프로그램에는 특수한 폰트를 지정하지 않는 것이 안전합니다.
- 다만 프로그래밍을 학습 중인 분들은 다양한 폰트를 표시해 보길 바랍니다.
- 존재하지 않는 폰트를 지정하면 파이썬에서는 기본으로 지정된 폰트를 사용해서 표시합니다.
- 일반적으로 Times New Roman은 많은 환경에서 사용할 수 있는 폰트입니다.
- 이후 프로그램에서 폰트는 모두 Times New Roman으로 지정합니다.

2. 라벨 배치하기

■ 라벨 표시 위치

- 컴퓨터 화면이나 윈도우 안의 좌표는 왼쪽 위 모서리를 원점 (0, 0)으로 합니다.
- 가로 방향이 X 축, 세로 방향이 Y 축입니다.
- Y 축은 반대로 아래로 갈수록 값이 커집니다.
- `place()` 명령은 이 X 좌표와 Y 좌표로 값을 지정합니다.



3. 버튼 배치하기

- 다음은 버튼을 배치합니다.
- 그리고 버튼 클릭을 판정하는 프로그램을 확인합니다.
- 버튼 배치
 - 버튼은 `Button()` 명령으로 만든 뒤 `place()` 명령으로 배치합니다.
 - 다음 프로그램을 입력하고 파일 이름을 붙여 저장한 뒤 실행해 봅니다.
 - 예제 `ex0503_1.py` (버튼 생성과 배치 관련 내용은 굵게 표기)

```
1 import tkinter
2 root = tkinter.Tk()
3 root.title("첫 번째 버튼")
4 root.geometry("800x600")
5 button = tkinter.Button(root, text="버튼 문자열",  
font=("Times New Roman", 24))
6 button.place(x=200, y=100)
7 root.mainloop()
```

tkinter 모듈 импорт
윈도우 객체 생성
윈도우 제목 지정
윈도우 크기 지정
버튼 컴포넌트 생성

윈도우에 버튼 배치
윈도우 표시

- 이 프로그램을 실행하면 윈도우 안에 버튼이 표시됩니다.

3. 버튼 배치하기

■ 버튼 배치

- 5~6번 행에서 버튼을 만들고 배치하는 식은 다음과 같습니다.

```
■ 버튼 변수명 = tkinter.Button(윈도우 객체, text = "라벨 문자열", font = ("폰트 명", 폰트 크기))  
■ {버튼 변수명}.place(x = X 좌표, y = Y 좌표)
```

- 표시할 문자열과 폰트 지정은 라벨 생성 식과 동일하며, 버튼 배치 또한 place() 명령을 사용합니다.

■ 버튼 클릭 시 반응

- 버튼을 클릭하면 반응하도록 합니다.
- 파이썬에서는 버튼을 클릭했을 때의 처리를 함수로 정의하고, 버튼을 생성하는 식 안에 'command=함수'를 입력하면 버튼을 클릭했을 때 해당 함수를 실행하고 이를 확인합니다.

3. 버튼 배치하기

■ 버튼 클릭 시 반응

■ 예제 ex0503_2.py

```
1 import tkinter
2
3 def click_btn( ):
4     button["text"] = "클릭했습니다."
5
6 root = tkinter.Tk( )
7 root.title("첫 번째 버튼")
8 root.geometry("800x600")
9 button = tkinter.Button(root, text="클릭하십시오.",
10 font=("Times New Roman", 24), command=click_btn)
11
12 button.place(x=200, y=100)
13 root.mainloop( )
```

tkinter 모듈 импорт

click_btn() 함수 선언
버튼 문자열 변경

윈도우 객체 생성
윈도우 제목 지정
윈도우 크기 지정

버튼 컴포넌트 생성,
'command='로 클릭
시 동작할 함수 지정

윈도우에 버튼 배치
윈도우 표시

3. 버튼 배치하기

■ 버튼 클릭 시 반응

- 함수와 버튼을 생성하는 식은 다음과 같습니다.

```
def click_btn():  
    button["text"] = "클릭했습니다."
```

버튼을 클릭하면 함수를
실행합니다.

```
button = tkinter.Button(root, text="클릭하십시오.",  
font=("Times New Roman", 24), command=click_btn)
```


4. 캔버스 사용하기

- 이미지나 도형을 그리는 GUI를 캔버스(Canvas)라고 부릅니다.
- 캔버스는 게임 개발에 반드시 필요한 컴포넌트 중 하나입니다.
- 캔버스 사용법을 확인합니다.

- 캔버스 배치
 - 캔버스는 `Canvas()` 명령으로 생성하고 `pack()` 명령과 `place()` 명령으로 배치합니다.

4. 캔버스 사용하기

■ 캔버스 배치

- 캔버스는 `Canvas()` 명령으로 생성하고 `pack()` 명령과 `place()` 명령으로 배치합니다.
- 다음 프로그램을 입력하고 파일 이름을 붙여 저장한 뒤 실행해 봅니다.
- 예제 `ex0504_1.py`

1	<code>import tkinter</code>	tkinter 모듈 импорт
2	<code>root = tkinter.Tk()</code>	윈도우 객체 생성
3	<code>root.title("첫 번째 캔버스")</code>	윈도우 제목 지정
4	<code>canvas = tkinter.Canvas(root, width=400, height=600, bg="skyblue")</code>	캔버스 컴포넌트 생성
5	<code>canvas.pack()</code>	윈도우에 캔버스 배치
6	<code>root.mainloop()</code>	윈도우 표시

- 이 프로그램을 실행하면 하늘색 캔버스가 배치된 윈도우가 표시됩니다.

4. 캔버스 사용하기

■ 캔버스 배치

- `pack()` 명령으로 배치하면 캔버스 크기에 맞춰 윈도우 크기가 결정됩니다.
- 윈도우에 캔버스만을 배치하는 경우에는 이 프로그램과 같이 `root.geometry()`를 생략할 수 있습니다.
- 캔버스를 만드는 식은 다음과 같습니다.

```
캔버스 변수명 = tkinter.Canvas(윈도우 객체,  
width=폭, height=높이, bg=배경색)
```

- 배경색은 `red`, `green`, `blue`, `yellow`, `black`, `white` 등의 영어 단어나 16진수 값으로 지정할 수 있습니다.

4. 캔버스 사용하기

■ 캔버스에 이미지 표시하기

- 캔버스에 이미지를 표시 할 때는 `PhotoImage()` 명령으로 이미지를 로딩하고 `create_image()` 명령으로 이미지를 그립니다.
- 다음 프로그램을 입력하고 파일 이름을 붙여 저장한 뒤 실행해 봅니다.
- 예제 `ex0504_2.py` (이미지 로딩 및 그리기 관련 내용은 굵게 표시)

1	<code>import tkinter</code>	tkinter 모듈 импорт
2	<code>root = tkinter.Tk()</code>	윈도우 객체 생성
3	<code>root.title("첫 번째 캔버스")</code>	윈도우 제목 지정
4	<code>canvas = tkinter.Canvas(root, width=400, height=600)</code>	캔버스 컴포넌트 생성
5	<code>canvas.pack()</code>	윈도우에 캔버스 배치
6	<code>photo = tkinter.PhotoImage(file="hyunju.png")</code>	photo에 이미지 파일 로딩
7	<code>canvas.create_image(200, 300, image=photo)</code>	캔버스에 이미지 그리기
8	<code>root.mainloop()</code>	윈도우 표시

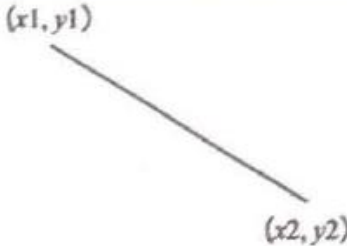
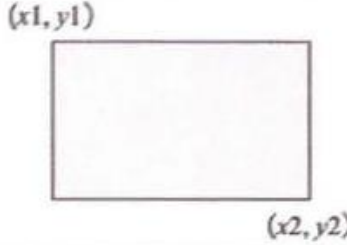
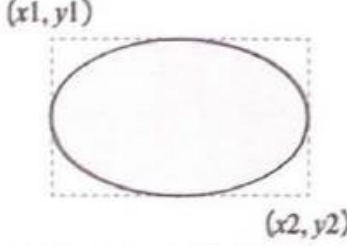
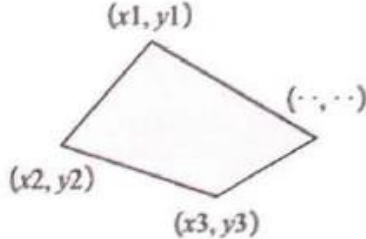
- 이 프로그램을 실행하면 캔버스에 이미지가 표시됩니다.

■ 캔버스에 이미지 표시하기

- 그러면 이미지 로딩과 그리기에 관해 살펴보겠습니다.
- 6번 행 `PhotoImage()` 명령에서 'file=파일 명'으로 이미지 파일을 지정하고 변수 `photo`에 이미지를 로드합니다.
- 7번 행 `create_image()` 명령의 인수는 이미지를 그릴 X 좌표와 Y 좌표, `image='이미지'를 로딩한 변수'`입니다.
- `create_image()` 명령에 지정한 X 좌표, Y 좌표는 이미지의 중점임에 주의합니다.
- 예를 들어, `canvas.create_image(0, 0, image=photo)`라고 입력하면 캔버스 원점인 왼쪽 상단 모서리를 이미지의 중점으로 지정하기 때문에 이미지의 일부만 보이게 됩니다.

5. 캔버스에 도형 표시하기

■ 이번 절에서는 캔버스에 도형을 그리는 명령에 대해 살펴보겠습니다.

직선	<code>create_line(x1, y1, x2, y2, fill=색, width=선 굵기)</code> ※ 3번째, 4번째 점 등 여러 점 지정 가능 ※ 3점 이상 지정 시 <code>smooth=True</code> 로 지정하면 곡선을 그립니다.	
사각형	<code>create_rectangle(x1, y1, x2, y3, fill=내부 색, outline=테두리 색, width=테두리 굵기)</code>	
타원형	<code>create_oval(x1, y1, x2, y2, fill=내부 색, outline=테두리 색, width=테두리 굵기)</code>	
다각형	<code>create_polygon(x1, y1, x2, y2, x3, y3, ..., fill=내부 색, outline=테두리 색, width=테두리 굵기)</code> ※ 여러 점 지정 가능	

5. 캔버스에 도형 표시하기

- 정사각형, 직사각형은 프로그램에서는 모두 사각형으로 간주합니다.
- 이 밖에 원호를 그리는 명령어는 다음과 같습니다.

```
create_arc(x1, y1, x2, y2, fill=내부 색, outline=테두리 색, start=시작 각도,  
extent=그릴 횟수, style=tkinter.***)
```

- ***에는 PIESLICE, CHORD, ARC를 넣을 수 있습니다.
- 인수에 따라 어떤 도형이 그려지는지 프로그램에서 확인해 봅니다.

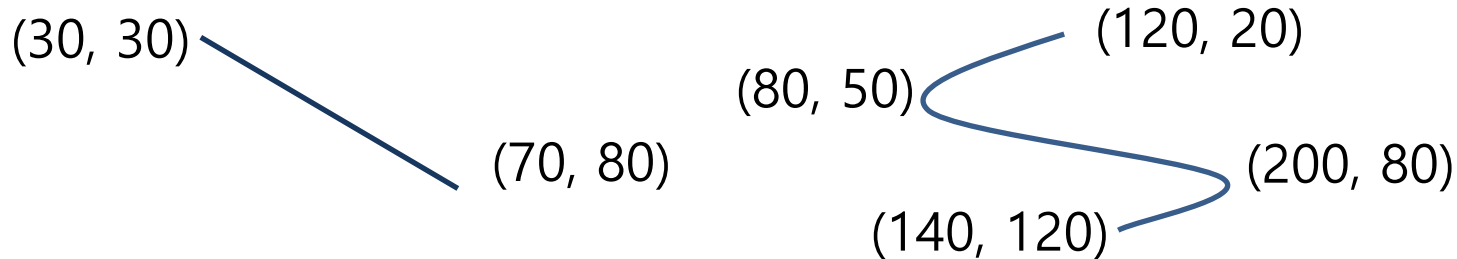
5. 캔버스에 도형 표시하기

- 다음 프로그램에서 그리기 명령들을 확인해 봅시다.
- 캔버스에 “자기 이름”을 색은 녹색, 폰트는 ‘Times New Roman’, 크기는 24로 설정하여 출력해보겠습니다.
- 출력되는 문자열의 위치는 x축으로 250, y축으로 25 떨어진 지점에 위치시킵니다.

```
1 import tkinter
2 root = tkinter.Tk( )
3 root.title("캔버스에 도형 그리기")
4 root.geometry("500x400")
5 cvs = tkinter.Canvas(root, width=500, height=400, bg='white')
6 cvs.pack( )
7 cvs.create_text(250, 25, text='문자열', fill='green', font=('Times New Roman',
8 cvs.mainloop( )
```


5. 캔버스에 도형 표시하기

■ 이번에는 캔버스에 다음과 같은 선들을 추가해 보겠습니다.



```
1 import tkinter
2 root = tkinter.Tk( )
3 root.title("캔버스에 도형 그리기")
4 root.geometry("500x400")
5 cvs = tkinter.Canvas(root, width=500, height=400, bg='white')
6 cvs.pack( )
7 cvs.create_text(250, 25, text='문자열', fill='green', font=('Times New Roman',
8 24))
9 cvs.create_line(30, 30, 70, 80, fill='navy', width=5)
10 cvs.create_line(120, 20, 80, 50, 200, 80, 140, 120, fill='blue', smooth=True)
11 cvs.mainloop( )
```

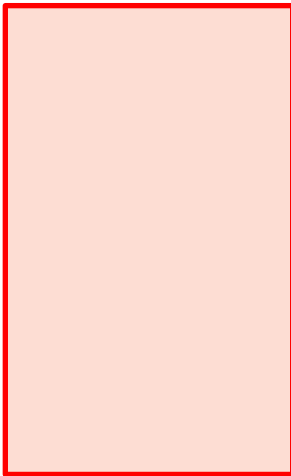
5. 캔버스에 도형 표시하기

■ 이번에는 사각형을 추가하겠습니다.

(40, 140)



(60, 240)



(160, 200)

(120, 360)

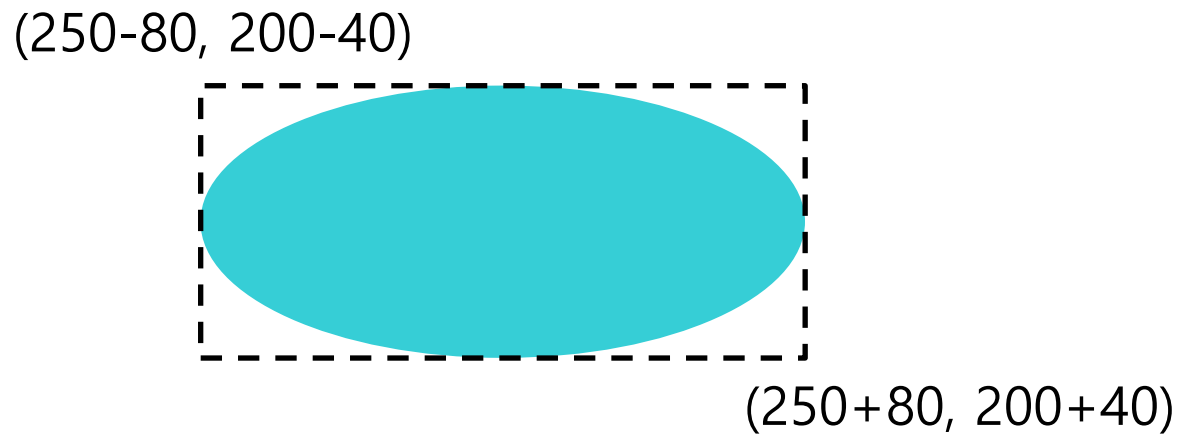
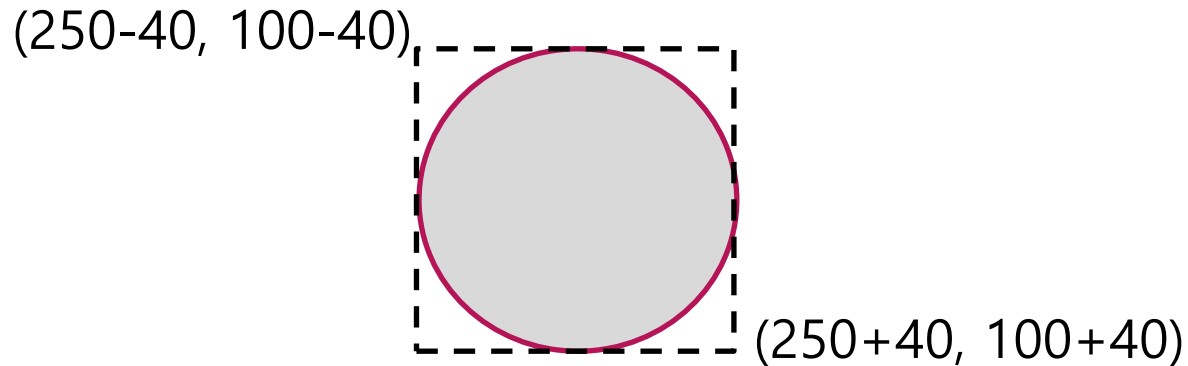
5. 캔버스에 도형 표시하기

■ 이번에는 사각형을 추가하겠습니다.

```
1 import tkinter
2 root = tkinter.Tk( )
3 root.title("캔버스에 도형 그리기")
4 root.geometry("500x400")
5 cvs = tkinter.Canvas(root, width=500, height=400, bg='white')
6 cvs.pack( )
7 cvs.create_text(250, 25, text='문자열', fill='green', font=('Times New Roman',
8 24))
9 cvs.create_line(30, 30, 70, 80, fill='navy', width=5)
10 cvs.create_line(120, 20, 80, 50, 200, 80, 140, 120, fill='blue', smooth=True)
11 cvs.create_rectangle(40, 140, 160, 200, fill='lime')
12 cvs.create_rectangle(60, 240, 120, 360, fill='pink', outline='red', width=5)
13 cvs.mainloop( )
```

5. 캔버스에 도형 표시하기

- 이번에는 원과 타원을 추가하겠습니다.



5. 캔버스에 도형 표시하기

■ 이번에는 원과 타원을 추가하겠습니다.

1~11행 생략

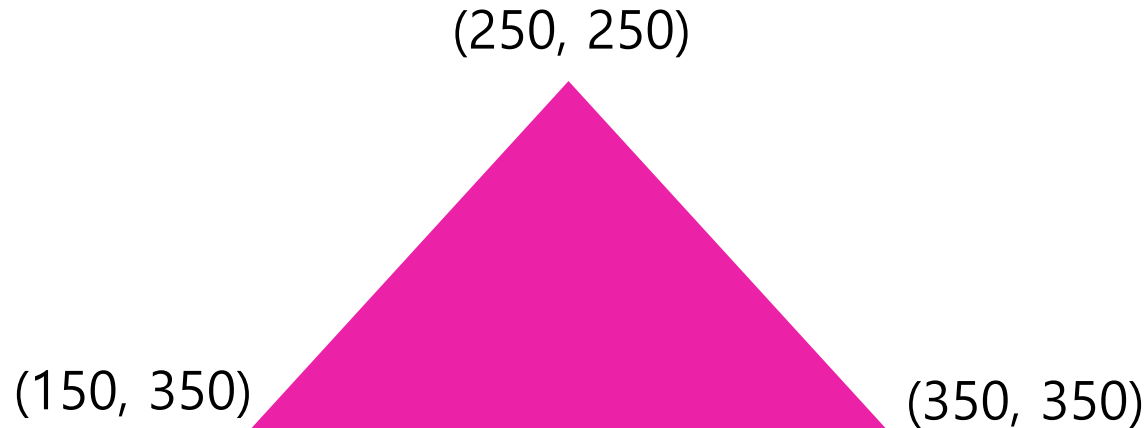
```
12  cvs.create_oval(250-40, 100-40, 250+40, 100+40, fill="silver",  
    outline="purple")
```

```
13  cvs.create_oval(250-80, 200-40, 250+80, 200+40, fill="cyan", width=0)
```

```
14  cvs.mainloop( )
```

5. 캔버스에 도형 표시하기

- 이번에는 삼각형을 추가하겠습니다.

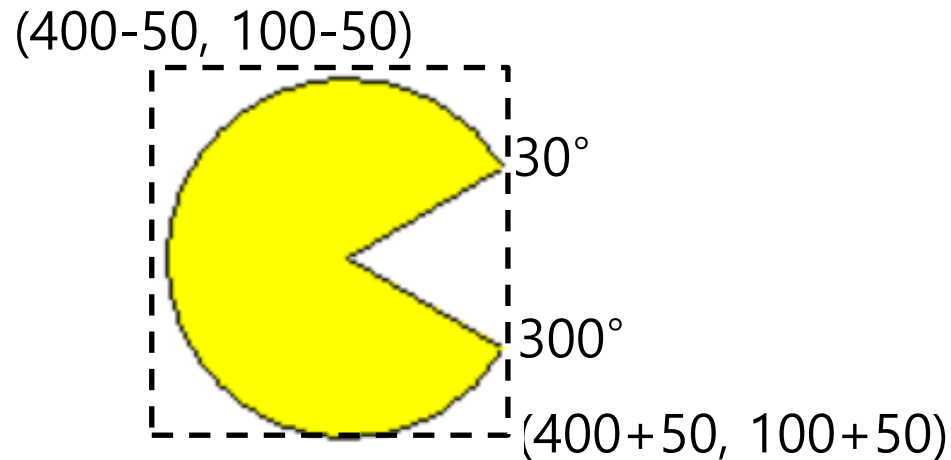


1~11행 생략

```
12  cvs.create_oval(250-40, 100-40, 250+40, 100+40, fill="silver",  
    outline="purple")  
13  cvs.create_oval(250-80, 200-40, 250+80, 200+40, fill="cyan", width=0)  
14  cvs.create_polygon(250, 250, 150, 350, 350, 350, fill="magenta", width=0)  
15  cvs.mainloop( )
```

5. 캔버스에 도형 표시하기

- 이번에는 원호를 추가하겠습니다.



1~14행 생략

```
15  cvs.create_arc(400-50, 100-50, 400+50, 100+50, fill="yellow", start=30,  
    extent=300)
```

```
16  cvs.mainloop( )
```

5. 캔버스에 도형 표시하기

- 다음 프로그램에서 그리기 명령들을 확인해 봅니다.

1~12행 생략

```
13 cvs.create_oval(250-80, 200-40, 250+80, 200+40, fill="cyan", width=0)
14 cvs.create_polygon(250, 250, 150, 350, 350, 350, fill="magenta", width=0)
15 cvs.create_arc(400-50, 100-50, 400+50, 100+50, fill="yellow", start=30,
    extent=300)
16 cvs.create_arc(400-50, 250-50, 400+50, 250+50, fill="gold", start=0,
    extent=120, style=tkinter.CHORD)
17 cvs.create_arc(400-50, 350-50, 400+50, 350+50, fill="orange", start=0,
    extent=120, style=tkinter.ARC)
18 cvs.mainloop( )
```


5. 캔버스에 도형 표시하기

- 문자는 7번 행과 같이 `create_text(x, y, text="문자열", fill=색, font=("폰트 명", 크기))`로 표시합니다.
- 원의 중심점이 (x, y) , 반지름이 r 인 경우 표시 위치 인수를 ' $x - r, y - r, x + r, y + r$ '로 지정하면 이해하기 쉽습니다(12번 행).
- `create_arc`에서 `style`을 생략하면 `style=tkinter.PIESLICE`가 됩니다(15번 행).



Thank You
