DAY 06

거북이 그래픽으로 그림 그리기

likegnu@Facebook

모두의 파이썬 20일 만에 배우는 프로그래밍 기초

길벗

거북이 그래픽으로 그림 그리기

01 거북이 그래픽 사용하기

02 거북이 그래픽의 동작 방식

01

1. 거북이 그래픽 사용하기

거북이 그래픽을 사용하기 전에 '거북이 그래픽을 이용하겠다'고 알려주어야 함

Import turtle 문장이 이런 알림 역할을 함

as t를 붙이지 않았을 때	as t를 붙였을 때
import turtle turtle.forward(100) turtle.right(100) turtle.forward(100)	import turtle as t t.forward(100) t.right(100) t.forward(100)

01

1. 거북이 그래픽 사용하기

다양한 거북이 모양을 지정할 수 있음
: “arrow”, “turtle”, “circle”, “square”, “triangle”, “classic”.

02

2. 거북이 그래픽의 동작 방식

» 자주 사용하는 거북이 그래픽 명령어 1

함수	설명	사용 예
forward(거리)/ fd(거리)	거북이가 앞으로 이동합니다.	t.forward(100) # 거북이가 100만큼 앞으로 이동합니다.
backward(거리) / back(거리)	거북이가 뒤로 이동합니다.	t.back(50) # 거북이가 50만큼 뒤로 이동합니다.
left(각도) / lt(각도)	거북이가 왼쪽으로 회전합니다.	t.left(45) # 거북이가 45도 왼쪽으로 회전합니다.
right(각도) / rt(각도)	거북이가 오른쪽으로 회전합니다.	t.right(45) # 거북이가 45도 오른쪽으로 회전합니다.
circle(반지름)	현재 위치에서 원을 그립니다.	t.circle(50) # 반지름이 50인 원을 그립니다.
down( ) / pendown( )	펜(잉크 묻힌 꼬리)을 내립니다.	t.down() # 이제 움직이면 그림이 그려집니다.
up( ) / penup( )	펜(잉크 묻힌 꼬리)을 올립니다.	t.up() # 거북이가 움직여도 선이 그려지지 않습니다.
shape("모양")	거북이 모양을 바꿉니다.	t.shape("turtle") # 진짜 거북이 모양으로 지정합니다. t.shape("arrow") # 화살표 모양의 거북이로 지정합니다. ※ 거북이 모양으로 "circle", "square", "triangle"을 사용할 수 있습니다.
speed(속도)	거북이 속도를 바꿉니다.	t.speed(1) # 가장 느린 속도 t.speed(10) # 빠른 속도 t.speed(0) # 최고 속도

02

2. 거북이 그래픽의 동작 방식

자주 사용하는 거북이 그래픽 명령어 2

pensize(굵기) / width	펜 굵기를 바꿉니다.	t.pensize(3) # 굵은 선으로 선을 그립니다.
color("색 이름")	펜 색을 바꿉니다.	t.color("red") # 빨간색으로 선을 그립니다.
bgcolor("색 이름")	화면의 배경색을 바꿉니다.	t.bgcolor("black") # 배경색을 흰색에서 검은색으로 바꿉니다.
fillcolor("색 이름")	도형 내부를 칠하는 색을 바꿉니다.	t.fillcolor("green") # 녹색으로 도형 내부를 칠합니다. ※ 색상을 따로 지정하지 않으면 현재 색으로 칠합니다.
begin_fill( )	도형 내부를 색칠할 준비를 합니다.	t.begin_fill() # 거북이 움직임을 색칠할 준비를 합니다.
end_fill( )	도형 내부를 색칠합니다.	t.end_fill() # begin_fill( ) 이후부터 지금까지 그린 그림에 맞춰 내부를 색칠합니다.
showturtle( ) / st( )	거북이를 화면에 표시합니다.	t.st() # 거북이를 화면에 표시합니다(기본 상태).
hideturtle( ) / ht( )	거북이를 화면에서 가립니다.	t.ht() # 거북이를 숨깁니다.
clear( )	거북이를 그대로 둔 채 화면을 지웁니다.	t.clear()
reset( )	화면을 지우고 거북이도 원래 자리와 상태로 되돌립니다.	t.reset()

02

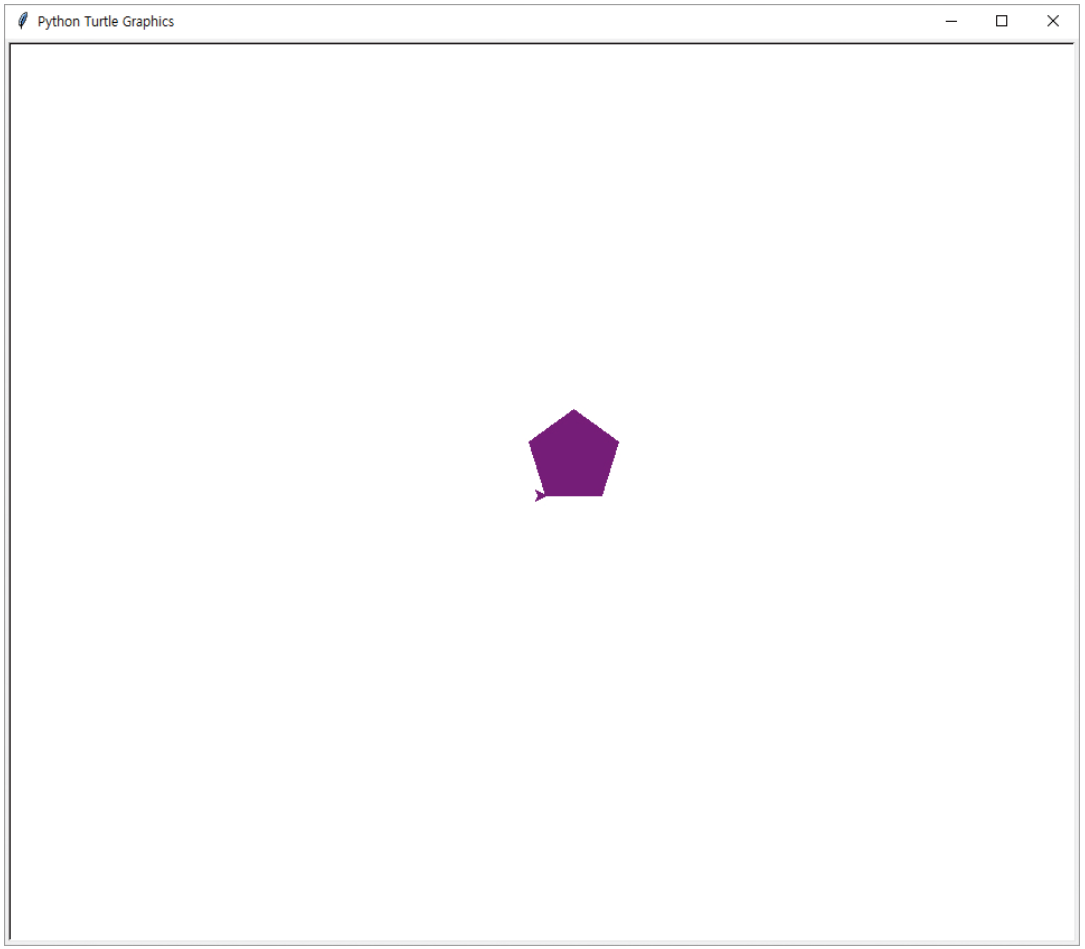
2. 정오각형을 그리는 프로그램

```
» import turtle as t

n = 5                   # 오각형을 그림(다른 값을 입력하면 다른 도형을 그림)
t.color("purple“)
t.begin_fill()          # 색칠할 영역을 시작
for x in range(n):      # n번 반복
    t.forward(50)       # 거북이가 50만큼 앞으로 이동
    t.left(360/n)       # 거북이가 360/n만큼 왼쪽으로 회전
t.end_fill()            # 색칠할 영역을 마무리
```

02

2. 거북이 그래픽의 동작 방식



02

2. 정오각형을 그리는 프로그램

» 정다각형의 외각

중학교 수학을 배웠다면 '모든 다각형 외각의 합은 360° '라는 사실을 배웠을 것입니다.

정n각형에는 모두 n개의 외각이 있는데, 이 값은 모두 같으므로 한 외각의 크기는 360/n이 됩니다.

t.left(360/n)으로 360/n°씩 회전하면서 같은 거리를 전진하면 정다각형이 그려지는 원리가 이해되었나요?

» 문제: 정육각형을 그려 봅시다!

02

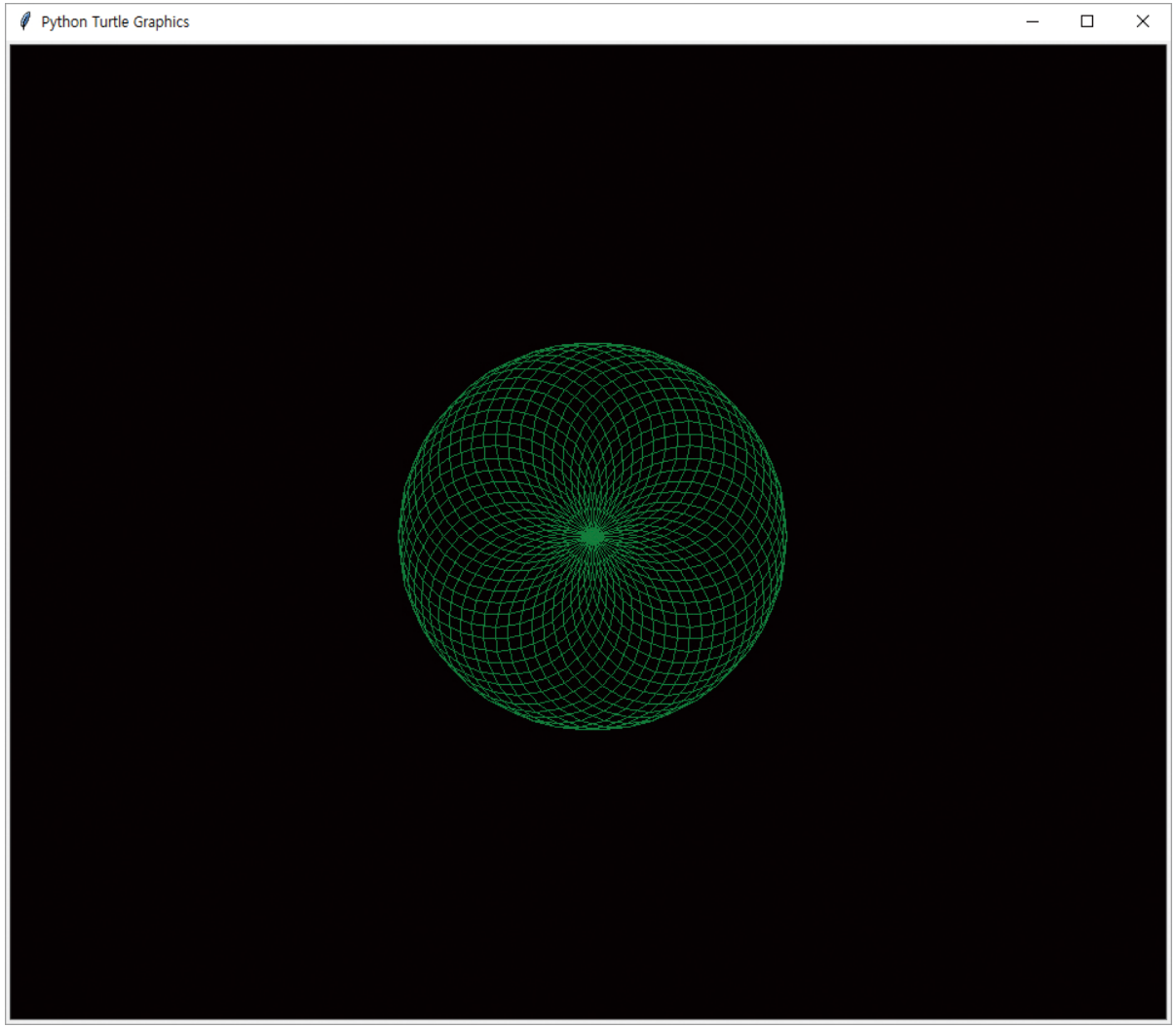
2. 원을 반복해서 그리는 프로그램

```
» import turtle as t

n = 50                  # 거북이가 왼쪽으로 회전할 각도를 지정(값을 바꿀 수 있음).
t.bgcolor("black")      # 배경색을 검은색으로 지정
t.color("green")        # 펜 색을 녹색으로 지정
t.speed(0)              # 거북이 속도를 가장 빠르게 지정
for x in range(n):      # x 값을 n번 실행
    t.circle(80)        # x 번 원을 그림
    t.left(360/n)       # 거북이가 360/n만큼 왼쪽으로 회전
```

02

2. 거북이 그래픽의 동작 방식



02

2. 선을 반복해서 그리는 프로그램

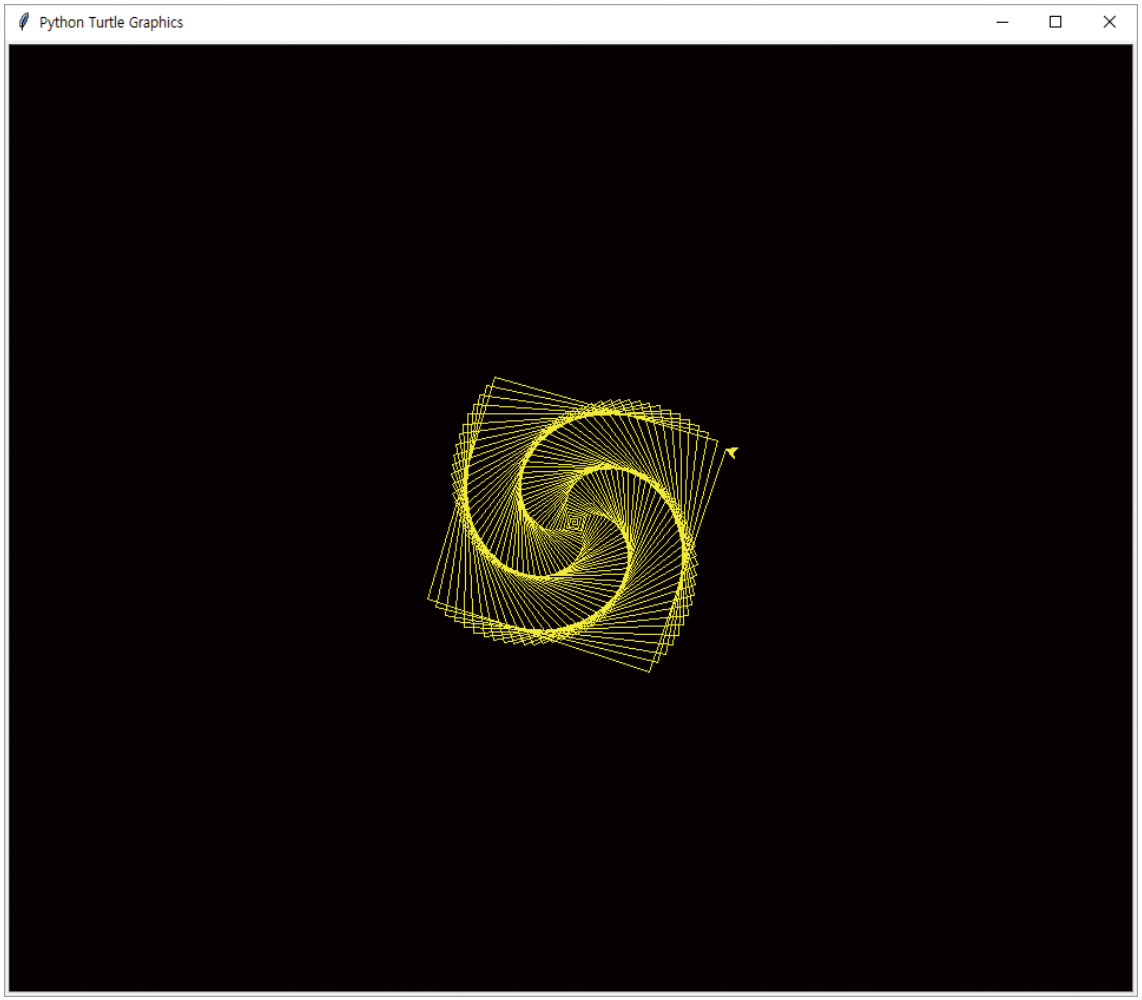
```
import turtle as t

angle = 89                  # 거북이가 왼쪽으로 회전할 각도를 지정(값을 바꿀 수 있음).

t.bgcolor("black")          # 배경색을 검은색으로 지정
t.color("yellow")           # 펜 색을 노란색으로 지정
t.speed(0)                  # 거북이 속도를 가장 빠르게 지정
for x in range(200):        # x 값을 0에서 199까지 바꾸면서 200번 실행
    t.forward(x)            # x만큼 앞으로 이동(실행을 반복하면서 선이 길어짐)
    t.left(angle)           # 거북이가 왼쪽으로 89도 회전
```

02

2. 거북이 그래픽의 동작 방식



DAY 07
정보 입력하기

likegnu@Facebook

모두의 파이썬 20일 만에 배우는 프로그래밍 기초

길벗

정보 입력하기

01 컴퓨터의 입출력 장치

02 파이썬의 입력 처리

03 파이썬의 자료형

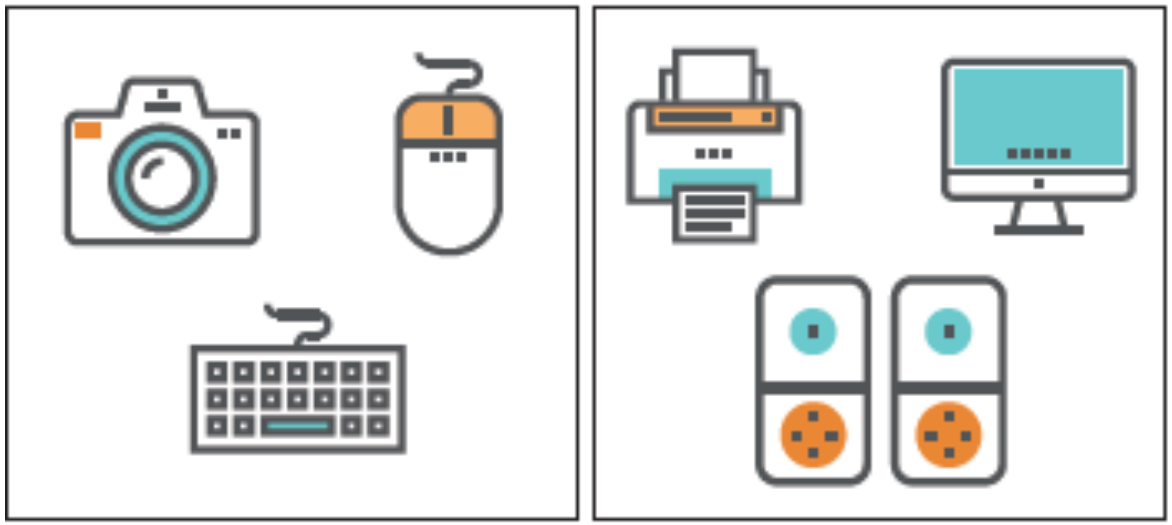
04 문자열이란?

01

1. 컴퓨터의 입출력 장치

» '입력 →처리 → 출력' 과정을 효과적으로 처리하기 위해 다양한 종류의 입력 장치와 출력 장치가 있음

- 입력 장치 : 키보드, 마우스, 터치스크린, 마이크, 카메라, 스캐너
- 출력 장치 : 모니터, 프린터, 스피커



02

2. 파이썬의 입력 처리

» 이름을 입력받아 Hello와 함께 보여 주는 프로그램

```
name = input("Your name? ")    # 이름을 입력받아 name 변수에 저장

print("Hello", name)           # Hello와 함께 name을 출력
```

» 실행결과

```
Your name? 김길벗
Hello 김길벗
```

2. 파이썬의 입력 처리

» 이름을 입력받아 Hello와 함께 보여 주는 프로그램

```
name = input("Your name? ")    # 이름을 입력받아 name 변수에 저장
```

입력 전 화면에 출력하는 부분, 입력과 무관

입력 명령어, 사용자가 입력을 진행해야 다음으로 넘어감

입력한 문자열을 저장하는 변수

2. 파이썬의 입력 처리

- 파이썬은 윈도(Windows)에서는 한글을 사용하는 데 큰 문제가 없습니다.

- 단, 애플(Apple)의 맥(Mac) 운영체제인 OS X에서는 한글이 제대로 입력되지 않는 현상이 나타납니다.

  - 이럴 때는 예제에 나오는 한글 문자열을 영어로 바꿔서 입력해야 합니다.

  - input( )을 이용해서 문자열을 입력 받을 때도 한글을 입력하면 제대로 입력되지 않습니다.

3. 파이썬의 자료형

» 파이썬에서 자주 사용하는 자료형

자료형	영어 이름	**파이썬 표기**	설명	예
정수	Integer	**int**	소수점이 없는 수	-2, -1, 0, 1, 2, 3
소수	Floating-point number	**float**	소수점(.)이 있는 수, 부동소수점수라고도 불린다.	-3.5, 0.0, 1.25, 5.0
문자열	String	**str**	알파벳 혹은 다른 문자로 이루어진 문장	“a”, “abc”, Hello?”, “3 people”, “비”, “여름”

03

3. 파이썬의 자료형

» 파이썬에서 사용하는 자료형 확인하기

- type(1)

- type(-3.5)

- type(“hello”)

4. 문자열이란?

» 문자열 : '문자의 나열'

- 한 글자로 된 문자열 : "a", "가", "3"
- 단어로 된 문자열 : "boy", "소년", "24"
- 문장으로 된 문자열 : "It rains.", "비가 옵니다."

» 문자열 변경은?

- 입력은 모두 문자열입니다.

04

4. 문자열이란?

» 퀴즈: 다음의 결과는 무엇일까요?

```
a = input("?")     # 변수 a에 첫 번째 입력을 받습니다. a = 문자열

b = input("?")     # 변수 b에 두 번째 입력을 받습니다. b = 문자열

print(a + b)       # a와 b를 ?
```

» 실행결과

```
? 3
? 7
```

04

4. 문자열이란?

» 퀴즈: 다음의 결과는 무엇일까요?

```
a = input("?")     # 변수 a에 첫 번째 입력을 받습니다. a = 문자열

b = input("?")     # 변수 b에 두 번째 입력을 받습니다. b = 문자열

print(a * b)       # a와 b를 ?
```

» 실행결과

```
? 3
? 7
```

04

4. 문자열이란?

» 숫자 두 개를 입력받아 곱하는 프로그램

```
x = input("?")          # 변수 x에 첫 번째 입력을 받습니다. x = 문자열
a = int(x)              # 문자열 x의 값을 정수(int)로 바꿔서 a에 넣음

x = input("?")          # 변수 x에 두 번째 입력을 받습니다. x = 문자열
b = int(x)              # 문자열 x의 값을 정수(int)로 바꿔서 b에 넣음

print(a * b)            # a와 b를 곱한 결과를 출력
```

» 실행결과

```
? 3
? 7
21
```

04

4. 문자열이란?

- 숫자 두 개를 입력받아 곱하는 프로그램을 다음과 같이 줄일 수 있습니다.

```
a = int(input("?"))    # 입력 문자열 값을 정수(int)로 바꿔서 a에 넣음

b = int(input("?"))    # 입력 문자열 값을 정수(int)로 바꿔서 b에 넣음

print(a * b)           # a와 b를 곱한 결과를 출력
```

- 실행결과

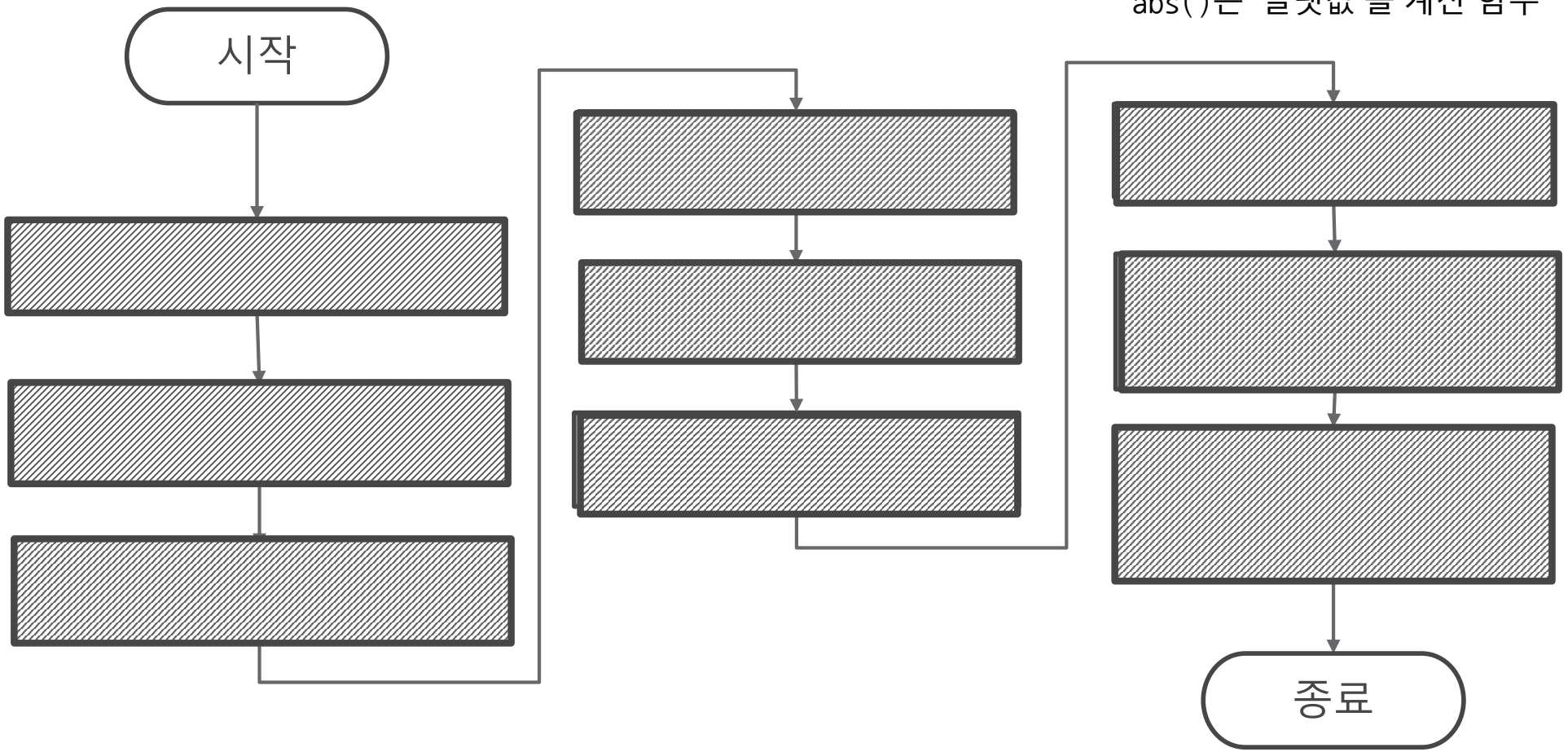
```
? 3
? 7
21
```

05

5. 속으로 20초를 세어 맞히는 프로그램

» 프로그램의 전체적인 흐름

abs()는 '절댓값'을 계산 함수



05

5. 속으로 20초를 세어 맞히는 프로그램

» import time

```
input("엔터를 누르고 20초를 셉니다.")
start = time.time()

input("20초 후에 다시 엔터를 누릅니다.")
end = time.time()

diff = end - start
print("실제 시간 :", diff, "초")
print("차이 :", abs(diff - 20), "초")  # end 시간에서 start 시간을 빼면 실제 걸린 시간을 계산할 수 있음
```

» 실행결과

```
엔터를 누르고 20초를 셈
20초 후에 다시 엔터를 누름
실제 시간 : 20.608863830566406 초
차이 : 0.6088638305664062 초
```