



DAY 13

거북이 그래픽 응용하기

모두의 파이썬 20일 만에 배우는 프로그래밍 기초

길벗

거북이 그래픽 응용하기

- 01 자주 사용하는 거북이 그래픽 명령어
- 02 키보드로 거북이를 조종해서 그림 그리기
- 03 마우스로 거북이를 조종해서 그림 그리기

1. 자주 사용하는 거북이 그래픽 명령어

» 자주 사용하는 거북이 그래픽 명령어 2

함수	설명	사용 예
pos() / position()	거북이의 현재 위치(좌표)를 구합니다 (x, y 둘 다).	t.pos()
xcor(), ycor()	거북이의 x 좌표나 y 좌표를 구합니다 (x, y 중 하나만).	a = t.ycor() # 거북이의 y 좌표를 구해 a에 저장합니다.
goto(x, y), setpos(x, y)	거북이를 특정 위치(좌표)로 보냅니다 (x, y 둘 다).	t.goto(100,50)
setx(x), sety(y)	거북이의 x 좌표나 y 좌표를 지정한 위치로 이동합니다(x, y 중 하나만).	t.sety(50) # 거북이의 y 좌표를 50만큼 이동합니다. x 좌표는 그대로 둡니다
distance(x, y)	현재 거북이가 있는 위치에서 특정 위치까지의 거리를 구합니다.	d = t.distance(100,100) # 현재 위치에서 (100, 100)까지의 거리를 구해서 d에 저장합니다.
heading()	거북이가 현재 바라보는 각도를 구합니다.	ang = t.heading()
towards(x, y)	현재 거북이가 있는 위치에서 특정 위치까지 바라보는 각도를 구합니다.	ang = t.towards(10,10) # 현재 위치에서 (10, 10)까지 가는 데 필요한 각도를 구해 ang에 저장합니다.
setheading(각도)/ seth(각도)	거북이가 바라보는 방향을 바꿉니다.	t.setheading(90) # 거북이가 화면 위쪽을 바라봅니다. ※ 거북이가 오른쪽을 바라볼 때의 각도가 0이며, 시계 반대 방향으로 돌면서 각도가 커집니다.
home()	거북이의 위치와 방향을 처음 상태로 돌립니다.	t.home() # 거북이가 화면 가운데인 (0, 0)에서 오른쪽(0도)을 바라봅니다.

1. 자주 사용하는 거북이 그래픽 명령어

» 자주 사용하는 거북이 그래픽 명령어 2

함수	설명	사용 예
onkeypress(함수, "키 이름")	키보드를 눌렀을 때 실행할 함수를 정합니다.	<pre>def f(): t.forward(10) t.onkeypress(f, "Up") # 위쪽 방향키 ↑ 를 누르면 f 함수를 호출합니다(f 함수는 거북이를 10만큼 앞으로 이동시킵니다).</pre>
onclick(함수)	마우스 버튼을 눌렀을 때 실행할 함수를 정합니다.	<pre>t.onclick(t.goto) # 마우스 버튼을 누르면 앞에서 정의한 goto 함수를 호출합니다(goto 함수는 거북이를 마우스 버튼을 누른 위치로 이동시킵니다).</pre>
ontimer(함수, 시간)	일정한 시간이 지난 뒤 실행할 함수를 정합니다.	<pre>def f(): t.forward(10) t.ontimer(f, 1000) # 1000밀리초(1초) 후에 f 함수를 호출합니다(f 함수는 거북이를 10만큼 앞으로 이동시킵니다.)</pre>
listen()	사용자 입력이 잘 처리되도록 거북이 그래픽 창에 포커스를 줍니다	<pre>t.listen()</pre>
title("창 이름")	거북이 그래픽 창의 이름을 지정합니다.	<pre>t.title("welcome") # 거북이 그래픽 창의 이름이 Untitled에서 welcome으로 바뀝니다.</pre>
write("문자열")	현재 거북이 위치에 문자를 출력합니다.	<pre>t.write("Hello") # 현재 거북이 위치에 Hello를 출력합니다. t.write("Hello", False, "center", (0, 20)) # 현재 거북이 위치에 가운데 정렬로 크기가 20인 Hello를 출력합니다(이 문장 전체를 구문처럼 통째로 기억하는 정도로만 알고 넘어가도 괜찮습니다).</pre>

1. 자주 사용하는 거북이 그래픽 명령어

» 태극 모양을 그리는 프로그램

» `import turtle as t`

```
t.bgcolor("black")  
t.speed(0)
```

배경색을 검은색으로 지정
거북이 속도를 가장 빠르게 지정

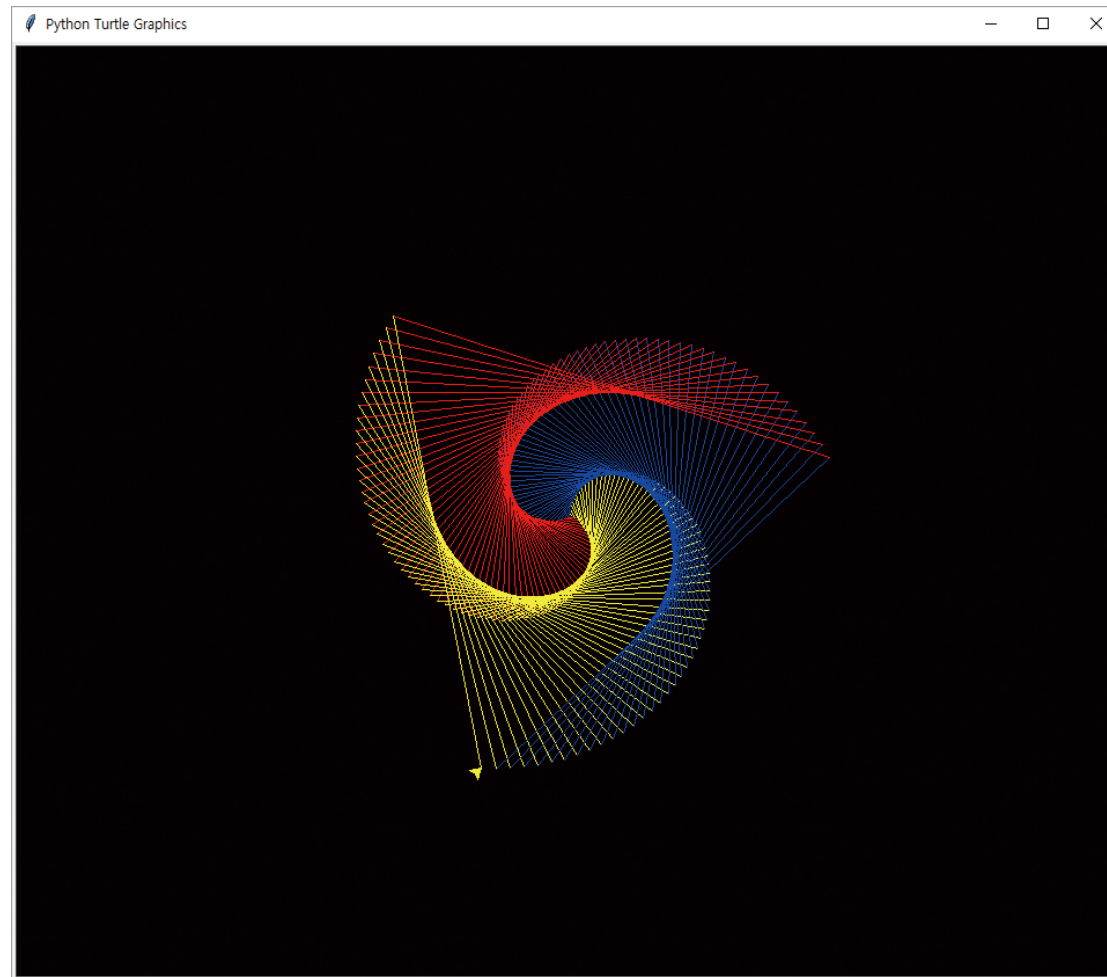
```
for x in range(200):  
    if x % 3 == 0:  
        t.color("red")  
    if x % 3 == 1:  
        t.color("yellow")  
    if x % 3 == 2:  
        t.color("blue")  
    t.forward(x * 2)  
    t.left(119)
```

for 반복 블록을 200번 실행
번갈아 가면서 선 색을 바꿈

x*2만큼 앞으로 이동(반복하면서 선이 점점 길어짐)
거북이를 119도 왼쪽으로 회전.

1. 자주 사용하는 거북이 그래픽 명령어

» 태극 모양을 그리는 프로그램



1. 자주 사용하는 거북이 그래픽 명령어

» 나머지 연산자(%)를 사용하여 색을 반복하는 원리

x	$x \% 3$ (3으로 나눈 나머지)	실행되는 문장	선 색
0	0	<code>t.color("red")</code>	빨간색
1	1	<code>t.color("yellow")</code>	노란색
2	2	<code>t.color("blue")</code>	파란색
3	0	<code>t.color("red")</code>	빨간색
4	1	<code>t.color("yellow")</code>	노란색
5	2	<code>t.color("blue")</code>	파란색

1. 자주 사용하는 거북이 그래픽 명령어

» 질문 119를 120으로 바꾸면 결과가 어떻게 나올까?

» `import turtle as t`

```
t.bgcolor("black")  
t.speed(0)
```

배경색을 검은색으로 지정
거북이 속도를 가장 빠르게 지정

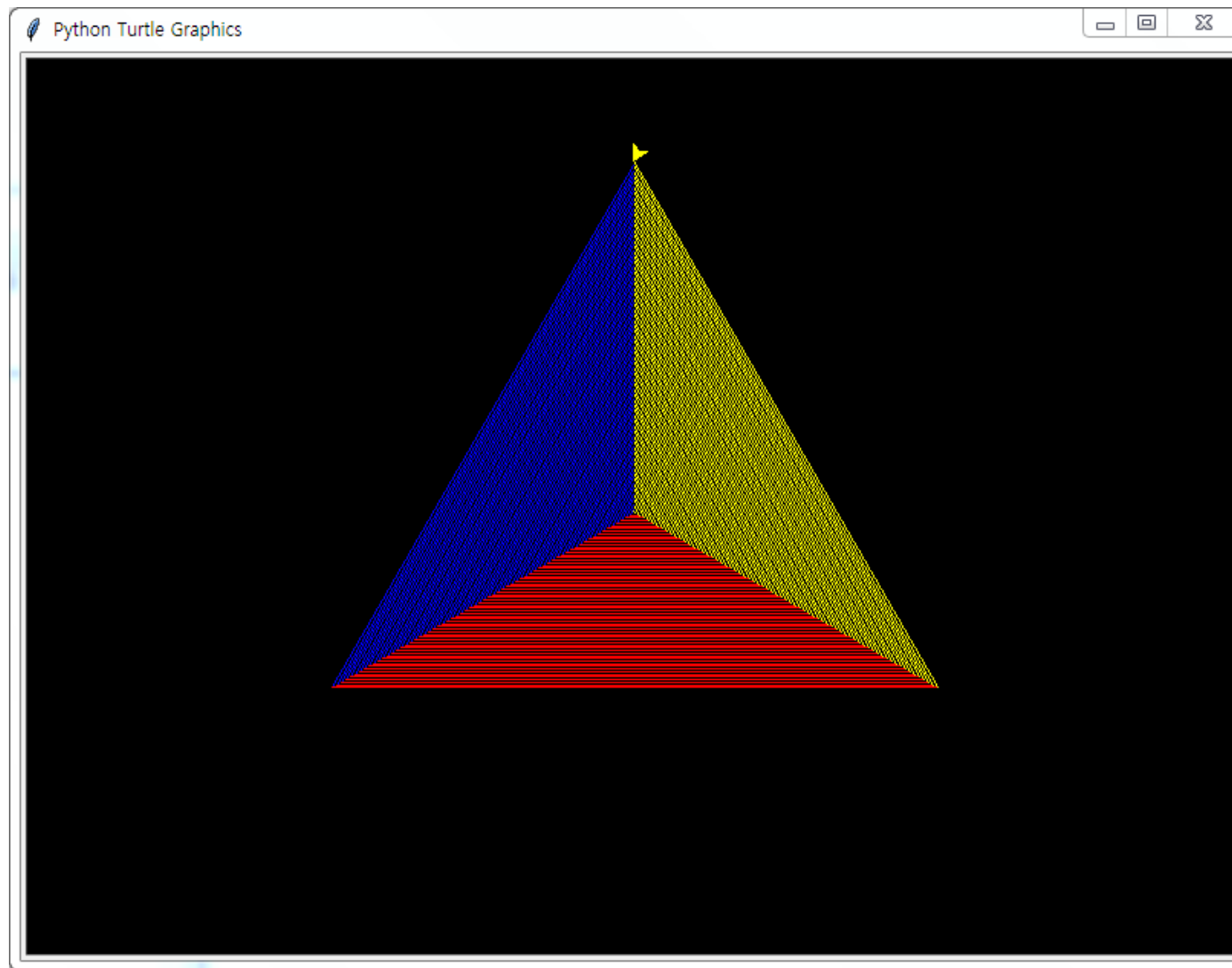
```
for x in range(200):  
    if x % 3 == 0:  
        t.color("red")  
    if x % 3 == 1:  
        t.color("yellow")  
    if x % 3 == 2:  
        t.color("blue")  
    t.forward(x * 2)  
    t.left(120)
```

for 반복 블록을 200번 실행
번갈아 가면서 선 색을 바꿈

x*2만큼 앞으로 이동(반복하면서 선이 점점 길어짐)
거북이를 119도 왼쪽으로 회전.

1. 자주 사용하는 거북이 그래픽 명령어

» 삼각형 모양을 그리는 프로그램



2. 키보드로 거북이를 조종해서 그림 그리기

» 자주 사용하는 거북이 그래픽 명령어 2 (다시 보기)

함수	설명	사용 예
setheading (각도)/ seth(각도)	거북이가 바라보는 방향을 바꿉니다.	<pre>t.setheading(90)</pre> # 거북이가 화면 위쪽을 바라봅니다. ※ 거북이가 오른쪽을 바라볼 때의 각도가 0이며, 시계 반대 방향으로 돌면서 각도가 커 집니다.
Forward (거리)/ fd(거리)	거북이가 앞으로 이동합니다.	<pre>t.forward(100)</pre> # 거북이가 100만큼 앞으로 이동합니다.

2. 키보드로 거북이를 조종해서 그림 그리기

```
» import turtle as t
```

```
def turn_right():          # 오른쪽으로 이동하는 함수  
    t.setheading(0)        # t.seth(0)으로 입력해도 됨  
    t.forward(10)          # t.fd(10)으로 입력해도 됨
```

```
def turn_up():             # 위로 이동하는 함수  
    t.setheading(90)  
    t.forward(10)
```

```
def turn_left():           # 왼쪽으로 이동하는 함수  
    t.setheading(180)  
    t.forward(10)
```

```
def turn_down():           # 아래로 이동하는 함수  
    t.setheading(270)  
    t.forward(10)
```

1. 자주 사용하는 거북이 그래픽 명령어

» 자주 사용하는 거북이 그래픽 명령어 2

함수	설명	사용 예
onkeypress(함수, "키 이름")	키보드를 눌렀을 때 실행할 함수를 정합니다.	<pre>def f(): t.forward(10) t.onkeypress(f, "Up") # 위쪽 방향키 ↑ 를 누르면 f 함수를 호출합니다 # f 함수는 거북이를 10만큼 앞으로 이동시킵니다.</pre>
listen()	사용자 입력이 잘 처리되도록 거북이 그래픽 창에 포커스를 줍니다	t.listen()

2. 키보드로 거북이를 조종해서 그림 그리기

```
def blank():
```

```
    t.clear()
```

화면을 지우는 함수

```
t.shape("turtle")
```

거북이 모양을 사용

```
t.speed(0)
```

거북이 속도를 가장 빠르게 지정

```
t.onkeypress(turn_right, "Right")
```

→ 를 누르면 turn_right 함수를 실행

```
t.onkeypress(turn_up, "Up")
```

```
t.onkeypress(turn_left, "Left")
```

```
t.onkeypress(turn_down, "Down")
```

```
t.onkeypress(blank, "Escape")
```

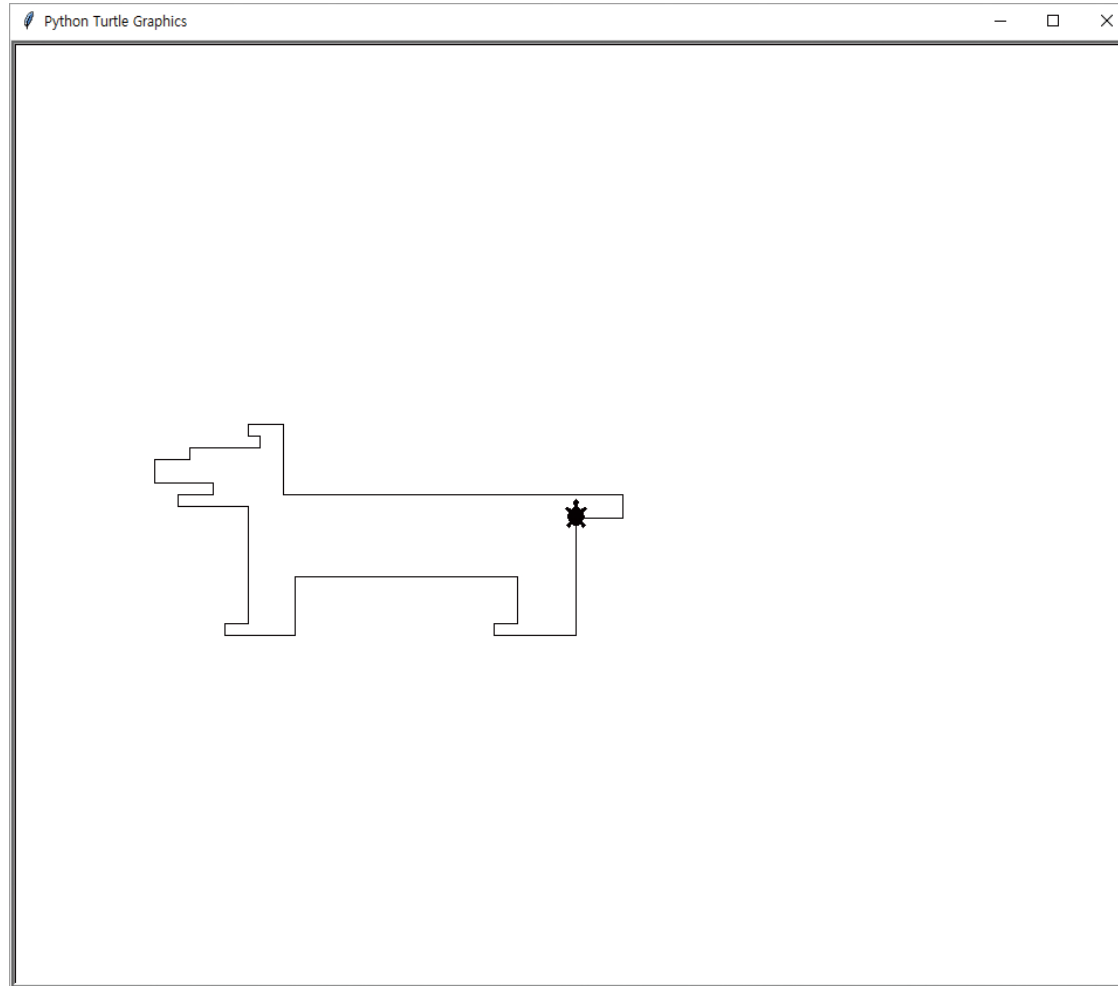
ESC 를 누르면 blank 함수를 실행

```
t.listen()
```

거북이 그래픽 창이 키보드 입력을 받음

02

2. 키보드로 거북이를 조종해서 그림 그리기



2. 키보드로 거북이를 조종해서 그림 그리기

» 실행하자마자 프로그램이 종료되었어요!

- 파이썬 IDLE 프로그램이 아닌 다른 파이썬 개발 프로그램(예를 들어 파이참)을 사용하고 있다면 실행하자마자 결과 없이 바로 프로그램이 종료될 수 있습니다.
- IDLE 프로그램을 사용하더라도 실행 설정이 다르다면 같은 현상이 나타날 수 있습니다.
- 그럴 때는 코드 제일 끝(13B-walk.py에서는 t.listen() 아래)에 다음 코드를 한 줄 추가한 다음 프로그램을 실행해 보세요.

```
t.mainloop()
```

- t.mainloop 함수는 사용자가 거북이 그래픽 창을 종료할 때까지 프로그램을 실행하면서 마우스나 키보드 입력을 계속 처리하도록 하는 함수입니다.

3. 마우스로 거북이를 조종해서 그림 그리기

```
» import turtle as t
```

```
t.speed(0)
```

```
# 거북이의 속도를 가장 빠르게 지정
```

```
t.pensize(2)
```

```
# 펜 굵기를 2로 지정
```

```
t.hideturtle()
```

```
# 거북이를 화면에서 숨김
```

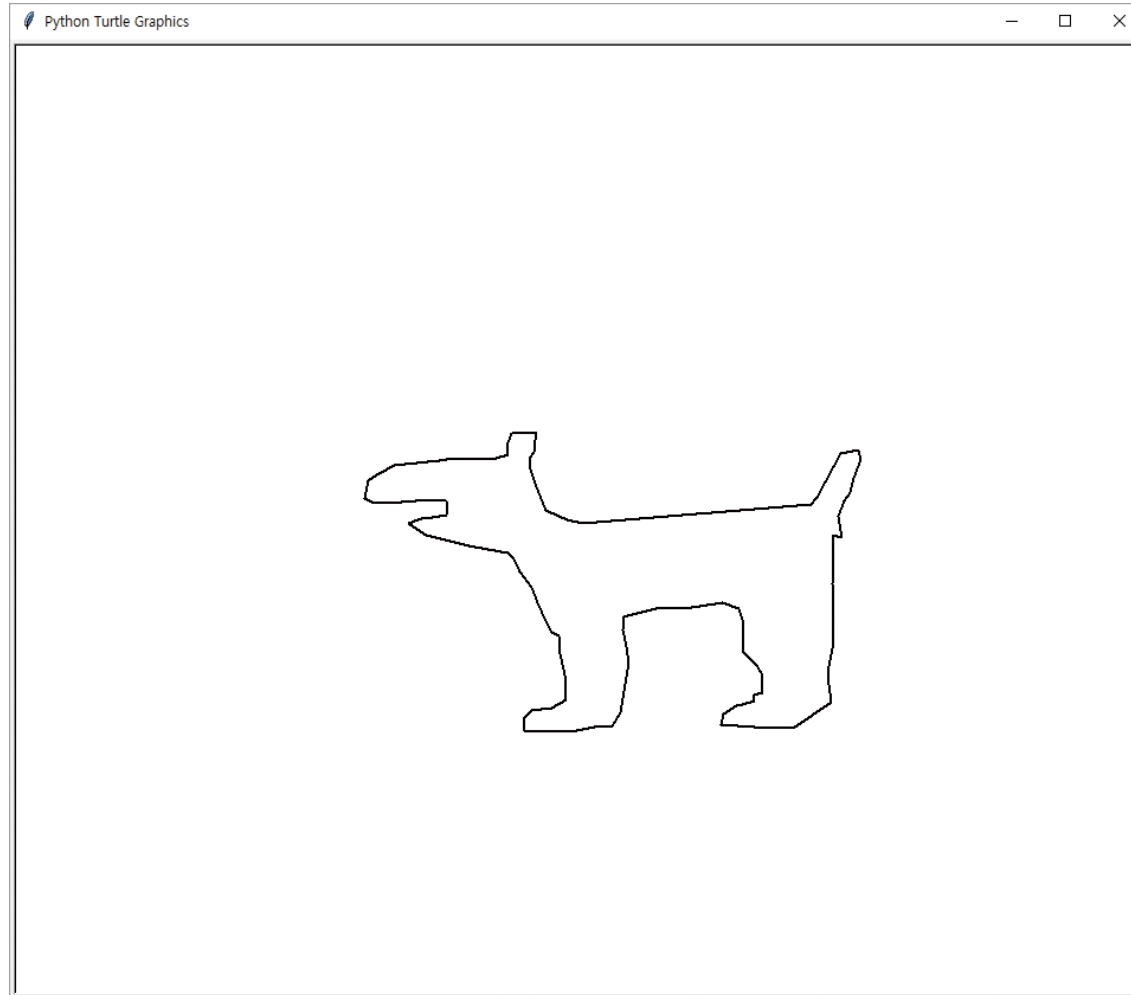
```
t.onscreenclick(t.goto)
```

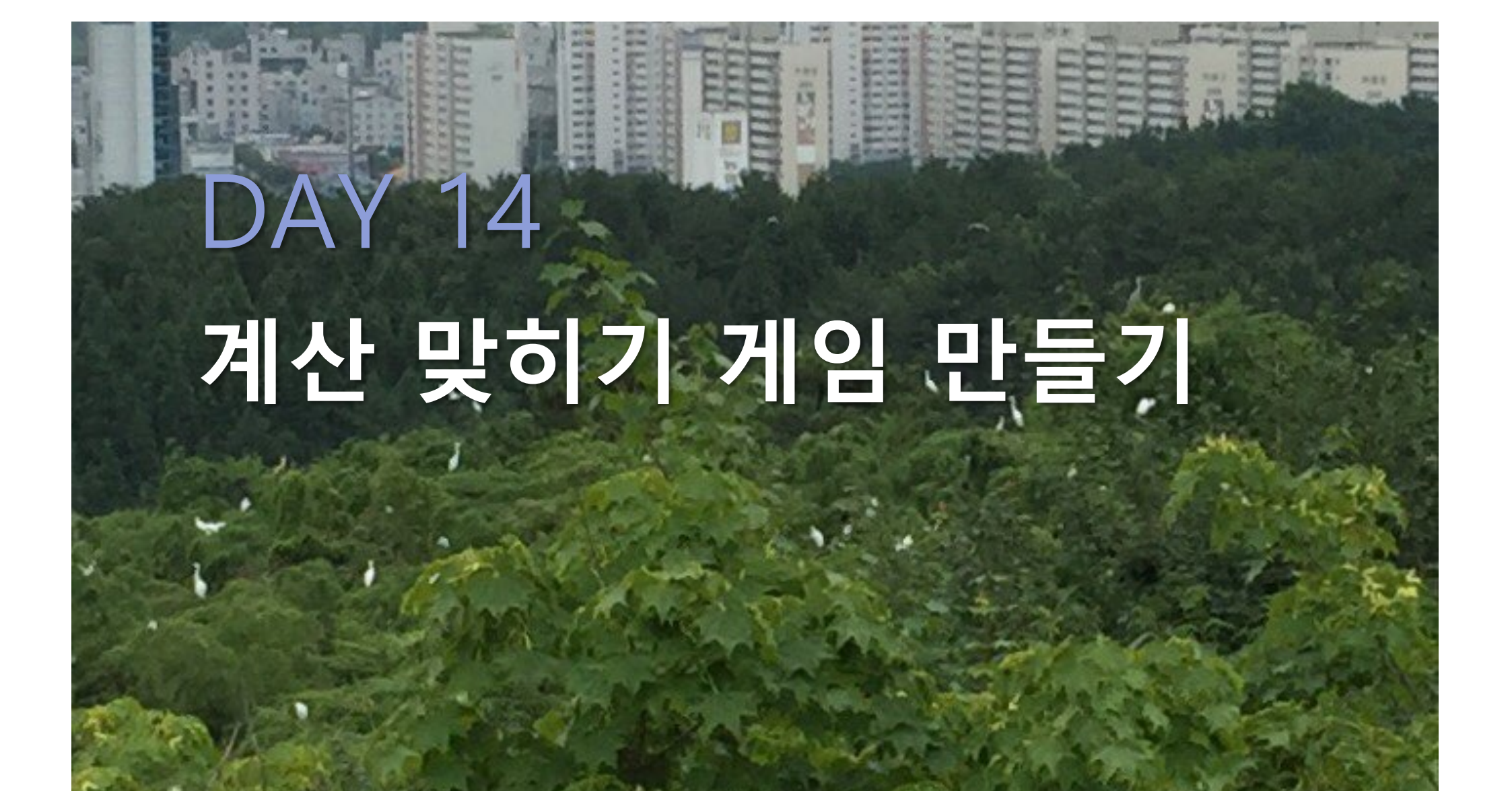
```
# 마우스 버튼을 누르면 t.goto 함수를 호출
```

```
# 그 위치로 거북이가 움직이면서 선을 그림
```

03

3. 마우스로 거북이를 조종해서 그림 그리기





DAY 14

계산 맞추기 게임 만들기

모두의 파이썬 20일 만에 배우는 프로그래밍 기초



계산 맞추기 게임 만들기

01 계산 맞추기 게임이란?

02 eval 함수

03 프로젝트 구조

1. 계산 맞추기 게임이란?

» 컴퓨터는 random 모듈을 이용해서 간단한 덧셈, 뺄셈, 곱셈 문제를 임의로 만들어 보여줌.

- 사용자가 이 문제를 보고계산을 해서 답을 입력하면, 컴퓨터는 이 답이 정답인지 오답인지 계산해서 점수를 매김.
- 이 과정을 다섯 번 반복해서 전체 정답 수를 알려 주는 게임.

» 8, 9일 자료 참조

2. eval 함수

» 문자열 수식을 계산한 결과로 돌려주는 함수

- 사용자는 eval 함수의 괄호 안에 문자열로 된 수식을 넣는다.
- eval 함수는 이 문자열을 수식으로 처리해서 계산한다.
- eval 함수는 계산 값을 함수의 결과값으로 돌려준다.

» eval 함수의 예:

```
>>>eval("3+5")
```

```
8
```

» 왜 eval 함수를 사용하는가?

- 계산기 프로그램에서는 문자열 입출력과 수식 계산이 계속 변환되어야 하는 관계로, eval 함수를 사용하면 알고리즘을 단순화할 수 있기 때문이다.

3. 프로젝트 구조

» 프로젝트 구조

- 사용자에게 제시할 계산 문제를 만드는 `make_question` 함수
 - 이 함수는 `random.randint` 함수로 계산에 필요한 숫자를 두 개 만든 후 덧셈(1), 뺄셈(2), 곱셈(3) 중 하나를 골라 계산 문제를 완성함
 - 이 함수는 인자는 없지만, 함수를 실행해서 만들어진 문제를 곱셈으로 돌려줌.
- 메인 프로그램
 - 실제로 게임을 진행하는 부분으로 정답/오답 횟수를 기록하는 변수 `sc1`, `sc2`를 0으로 초기함
 - 이후, `make_question` 함수를 호출하여 문제를 만들고 이를 사용자에게 보여줌.
 - 이후, 다음 사용자에게 입력을 받아 정답/오답을 판단하는 과정을 다섯 번 반복함.

» 여기서 잠깐! 위 프로그램의 순서도를 작성하라 (모둠 Lab 진행)

3. 프로젝트 구조

» 계산 문제를 맞히는 게임

```
import random
```

```
def make_question():
```

```
    a = random.randint(1, 40)
```

```
    b = random.randint(1, 20)
```

```
    op = random.randint(1, 3)
```

```
    # 문자열 변수 q에 문제를 만듭니다.
```

```
    # 첫 번째 숫자를 q에 저장합니다.
```

```
    q = str(a)
```

```
    # 연산자를 추가합니다.
```

```
    if op == 1:
```

```
        q = q + "+"
```

```
    if op == 2:
```

```
        q = q + "-"
```

```
    if op == 3:
```

```
        q = q + "*"
```

```
    # 두 번째 숫자를 q에 저장합니다.
```

```
    q = q + str(b)
```

```
    # 만들어진 문제를 돌려줍니다.
```

```
    return q
```

```
# 1~40 사이의 임의의 수를 a에 저장
```

```
# 1~20 사이의 임의의 수를 b에 저장
```

```
# 1~3 사이의 임의의 수를 op에 저장
```

```
# a 값(정수)을 문자열로 바꾸어 저장
```

```
# op 값이 1이면 덧셈 문제로 만들
```

```
# op 값이 2이면 뺄셈 문제로 만들
```

```
# op 값이 3이면 곱셈 문제로 만들
```

```
# b 값(정수)을 문자열로 바꾸어 q에 추가
```

3. 프로젝트 구조

```
# 정답/오답 횟수를 저장할 변수 sc1과 sc2를 0으로 초기화
sc1 = 0
sc2 = 0

for x in range(5):
    q = make_question()
    print(q)
    ans = input("=")
    r = int(ans)

    # 다섯 문제를 풀어봄
    # 문제를 만듦
    # 문제를 출력
    # 사용자에게 정답을 입력받음
    # 입력받은 정답을 정수로 바꿈

    # 컴퓨터가 계산한 결과인 eval(q)의 값과 사용자가 입력한 결과(r)를 비교
    if eval(q) == r:
        print("정답!")
        sc1 = sc1 + 1
    else:
        print("오답!")
        sc2 = sc2 + 1

print("정답 :", sc1, "오답 :", sc2)
if sc2 == 0: #
    print("당신은 천재입니다!") #오답이 0개일 때(전부 정답을 맞혔을 때)
```

3. 프로젝트 구조

» 실행결과

25*3

=75

정답!

37-18

=19

정답!

6-4

=2

정답!

3*11

=33

정답!

15-13

=12

오답!

정답 : 4 오답 : 1