학습목표

- 파이썬 프로그램의 거북이 그래픽 응용하기와 오브젝트을 응용하는 계산 맞히기 게임 만들기, 타자게임 만들기, 거북이 대포게임 만들기의 프로젝트 프로그램 제작 할 수 있어야 한다.

Unit 1. 거북이 그래픽 응용하기

01 자주 사용하는 거북이 그래픽 명령어

02 키보드로 거북이를 조종해서 그림 그리기

03 마우스로 거북이를 조종해서 그림 그리기

01

슬라이드1. 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.

01

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.	def f(): t.forward(10) t.onkeypress(f, "Up") # 위쪽 방향키 ↑ 를 누르면 f 함수를 호출합니다(f 함수는 거북이를 10만큼 앞으로 이동시킵니다).
onscreenclick(함수)	마우스 버튼을 눌렀을 때 실행할 함수를 정합니다.	t.onscreenclick(t.goto) # 마우스 버튼을 누르면 앞에서 정의한 goto 함수를 호출합니다(goto 함수는 거북이를 마우스 버튼을 누른 위치로 이동시킵니다).
ontimer(함수, 시간)	일정한 시간이 지난 뒤 실행할 함수를 정합니다.	def f(): t.forward(10) t.ontimer(f, 1000) # 1000밀리초(1초) 후에 f 함수를 호출합니다(f 함수는 거북이를 10만큼 앞으로 이동시킵니다.)
listen( )	사용자 입력이 잘 처리되도록 거북이 그래픽 창에 포커스를 줍니다	t.listen()
title(“창 이름”)	거북이 그래픽 창의 이름을 지정합니다.	t.title("welcome") # 거북이 그래픽 창의 이름이 Untitle에서 welcome으로 바뀝니다.

write(“문자열”)	현재 거북이 위치에 문자를 출력합니다.	t.write("Hello")
		# 현재 거북이 위치에 Hello를 출력합니다.
		t.write("Hello", False, "center", ("", 20))
		# 현재 거북이 위치에 가운데 정렬로 크기가 20인 Hello를 출력합니다(이 문장 전체를 구문처럼 통째로 기억하는 정도로만 알고 넘어가도 괜찮습니다).

01

1. 자주 사용하는 거북이 그래픽 명령어

» 태극 모양을 그리는 프로그램

» import turtle as t

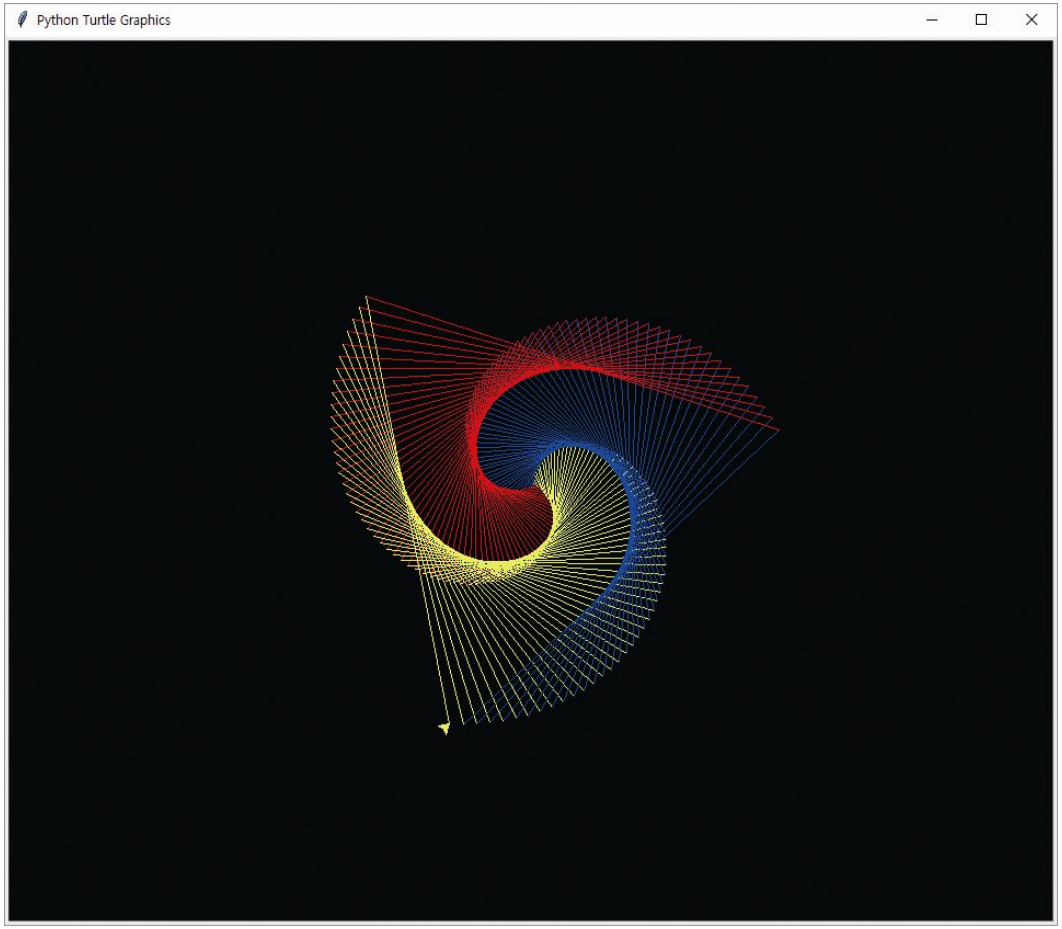
```
t.bgcolor("black")          # 배경색을 검은색으로 지정
t.speed(0)                  # 거북이 속도를 가장 빠르게 지정

for x in range(200):        # for 반복 블록을 200번 실행
    if x % 3 == 0:          # 번갈아 가면서 선 색을 바꿈
        t.color("red")
    if x % 3 == 1:
        t.color("yellow")
    if x % 3 == 2:
        t.color("blue")
    t.forward(x * 2)        # x*2만큼 앞으로 이동(반복하면서 선이 점점 길어짐)
    t.left(119)             # 거북이를 119도 왼쪽으로 회전.
```

01

1. 자주 사용하는 거북이 그래픽 명령어

» 태극 모양을 그리는 프로그램



01

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	t.color("red")	빨간색
1	1	t.color("yellow")	노란색
2	2	t.color("blue")	파란색
3	0	t.color("red")	빨간색
4	1	t.color("yellow")	노란색
5	2	t.color("blue")	파란색

02

슬라이드 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)
```

02

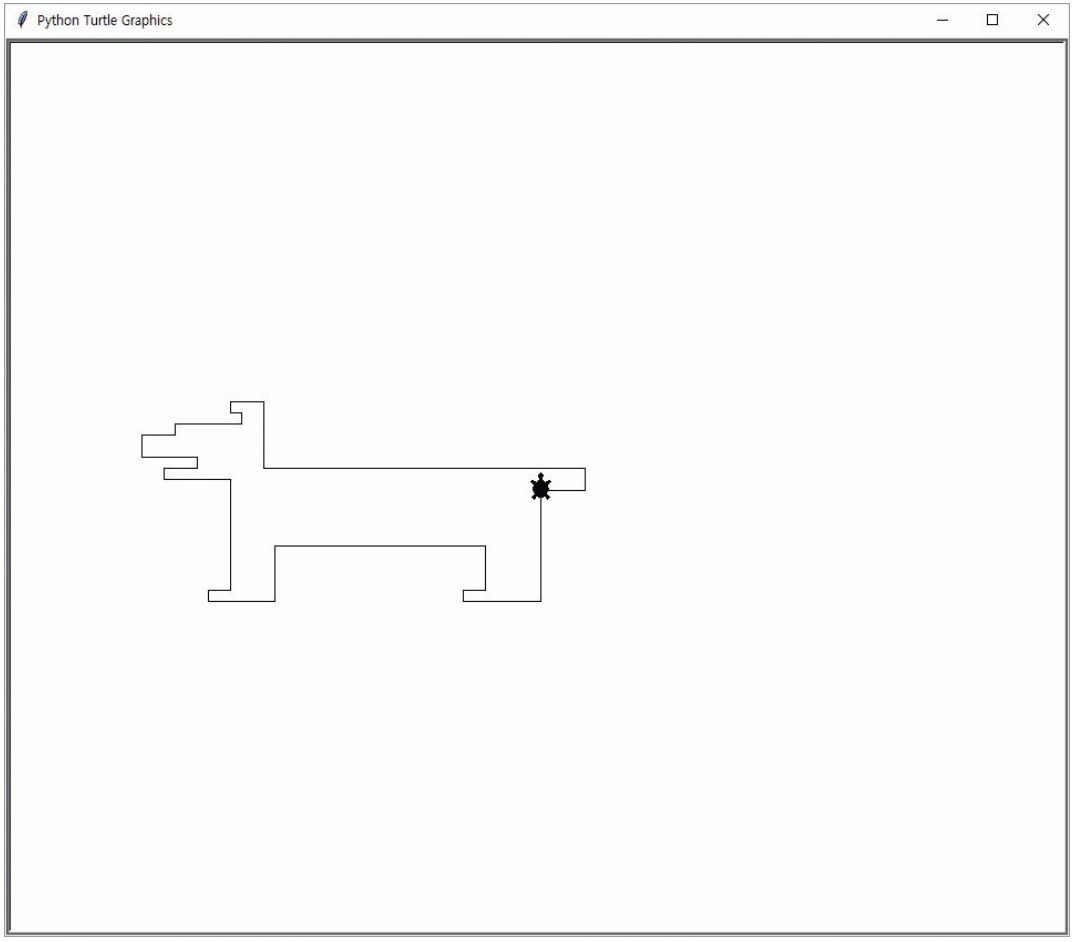
2. 키보드로 거북이를 조종해서 그림 그리기

```
def blank():                              # 화면을 지우는 함수
t.clear()

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



02

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()
```

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

03

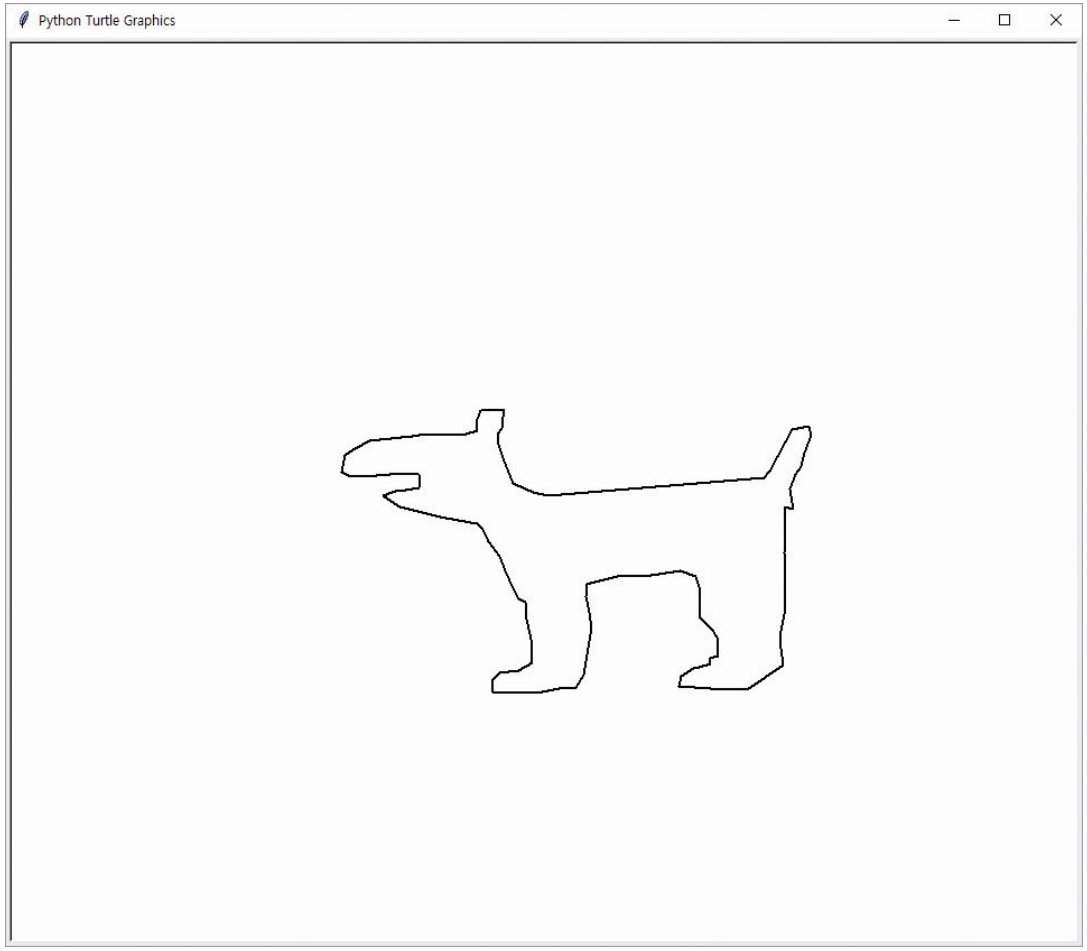
슬라이드 3. 마우스로 거북이를 조종해서 그림 그리기

```
» 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



Unit 2. 계산 맞히기 게임 만들기

01 계산 맞히기 게임이란?

02 eval 함수

03 프로젝트 구조

01

슬라이드1. 계산 맞히기 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.

02

슬라이드 2. eval 함수

- eval 함수의 괄호 안에 문자열로 된 수식을 넣으면, eval 함수는 이 문자열을 수식으로 처리해서 계산한 후, 계산 값을 함수의 결괏값으로 돌려줌

- 사용자가 풀 문제를 만들어서 '문자열'로 보여 줘야 하고, 또 사용자가 입력한 값이 맞는지 틀렸는지 체크하려면 이 값을 '계산'해야 하는 프로젝트에서 문제의 결괏값을 계산하기 위해 eval 함수를 사용

03

슬라이드 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 결괏값으로 돌려줌.
- 메인 프로그램
  실제로 게임을 진행하는 부분으로 정답/오답 횟수를 기록하는 변수 sc1, sc2를 0으로 초기화한 후, make_question 함수를 호출하여 문제를 만들고 이를 사용자에게 보여줌. 그런 다음 사용자에게 입력을 받아 정답/오답을 판단하는 과정을 다섯 번 반복함.

03

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 만듦

# op 값이 2이면 뺄셈 문제로 만듦

# op 값이 3이면 곱셈 문제로 만듦


# b 값(정수)을 문자열로 바꾸어 q에 추가
```

03

3. 프로젝트 구조

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

for x in range(5):                  # 다섯 문제를 풀어봄
    q = make_question()             # 문제를 만듦
    print(q)                        # 문제를 출력
    ans = input("=")                # 사용자에게 정답을 입력받음
    r = int(ans)                    # 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: #                      #오답이 0개일 때(전부 정답을 맞혔을 때)
   print("당신은 천재입니다!")
```

03

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
```

Unit 3.타자 게임 만들기

01 타자 게임이란?

02 리스트

03 random.dhoice 함수

04 프로젝트 구조

01

슬라이드1. 타자 게임이란?

» 게임이 시작되면 동물 이름으로 된 영어 단어가 화면에 표시됨. 사용자는 그 단어를 최대한 빠르고 정확하게 입력해야 함. 바르게 입력했으면 다음 문제로 넘어가고, 오타가 있으면 같은 단어가 한 번 더 나옴. 틀린 문제를 다시 입력하는 동안에도 시간은 계속 흐르기 때문에 속도뿐만 아니라 정확도도 중요한 게임.

02

슬라이드 2. 리스트

- 여러 정보를 하나로 묶어서 저장하고 관리할 수 있게 하는 기능

```
>>> a = [5, 7, 9]
>>> a
[5, 7, 9]
>>> a[0]
5
>>> a[2]
9
>>>
```

- 정수 5, 7, 9를 묶어 a라는 리스트를 만든 것
  a를 입력하면 [5, 7, 9]가 표시됨

  주의점 : 파이썬의 리스트에서는 순서를 1이 아닌 0부터 셈

```
a[0] = 5        # 리스트 a의 첫 번째 값
a[1] = 7        # 리스트 a의 두 번째 값
a[2] = 9        # 리스트 a의 세 번째 값
```

03

슬라이드 3. random.choice 함수

- 리스트에 들어 있는 자료들 중에서 임의로 하나를 고르는 함수

```
>>> x = ["a", "b", "c", "d"]
>>> import random
>>> random.choice(x)
'c'
>>> random.choice(x)
'b'
>>>
```

04

슬라이드 4. 프로젝트 구조

프로젝트 구조

- 사전 준비
  게임에 필요한 모듈을 임포트(import)함. Input 함수를 이용하여 사용자가 타자 게임을 시작할 준비를 하고 Enter 를 누를 때까지 기다림
- 메인 프로그램
  실제로 타자 게임을 처리하는 부분. 사용자에게 문제를 보여 주고 타자 입력을 받고 처리하는 과정을 다섯 번 반복(오타가 나면 다섯 번 이상 반복해야 할 수 있으므로 for가 아닌 while 명령을 사용).
- 결과를 계산해서 보여 주기
  사용자가 타자 문제를 모두 입력하는 데 걸린 시간을 소수점 둘째자리까지 계산해서 출력.

04

4. 프로젝트 구조

» 타자 게임 만들기

»
```
import random
import time

# 단어 리스트 : 여기에 단어를 추가하면 문제에 나옴
w = ["cat", "dog", "fox", "monkey", "mouse", "panda", "frog",
"snake", "wolf"]
n = 1                                   # 문제 번호
print("[타자 게임] 준비되면 엔터!")
input()                                 # 사용자가 엔터를 누를 때까지 기다림
start = time.time()                     # 시작 시간을 기록
```

04

4. 프로젝트 구조

```
q = random.choice(w)                # 단어 리스트에서 아무것이나 하나 뽑음
while n <= 5:                       # 문제를 다섯 번 반복
    print("*문제", n)
    print(q)                        # 문제를 보여줌
    x = input()                     # 사용자 입력을 받음
    if q == x:                      # 문제와 입력이 같을 때(올바로 입력했을 때)
        print("통과!")              # "통과!"라고 출력
        n = n + 1                   # 문제 번호를 1 증가
        q = random.choice(w)        # 새 문제를 뽑음
    else:
        print("오타! 다시 도전!")

end = time.time()                   # 끝난 시간을 기록
et = end - start                    # 실제로 걸린 시간을 계산
et = format(et, ".2f")              # 보기 좋게 소수점 둘째 자리까지만 표기
print("타자 시간 :", et, "초")
```

04

4. 프로젝트 구조

» 실행결과

[타자 게임] 준비되면 엔터!

*문제 1
snake
Snake
통과!
*문제 2
Frog
frog
통과!
*문제 3
frog
Frog
통과!
*문제 4
fox
fox
통과!
*문제 5
cat
cat
통과!
타자 시간 : 12.97 초
>>>

Unit 4. 거북이 대포 게임 만들기

01 거북이 대포 게임이란?

02 좌표

03 각도

04 글자 쓰기

05 프로젝트 구조

01

슬라이드1. 거북이 대포 게임이란?

- '곡사포' 게임의 한 종류.
  키보드 방향키인 ↑ , ↓ 로 발사 각도를 조절하고 SpaceBar 로 대포를 발사하면 화살촉 모양의 대포가 하늘로 날아감. 날아가던 대포가 땅에 닿을 때 초록색 목표 지점을 맞히면 'Good!'이라는 메시지를 보여 주고, 빗나가면 'Bad!'라는 메시지를 보여줌. 단, 이 게임은 발사 각도는 조절할 수 있지만 발사하는 힘은 조절할 수 없음

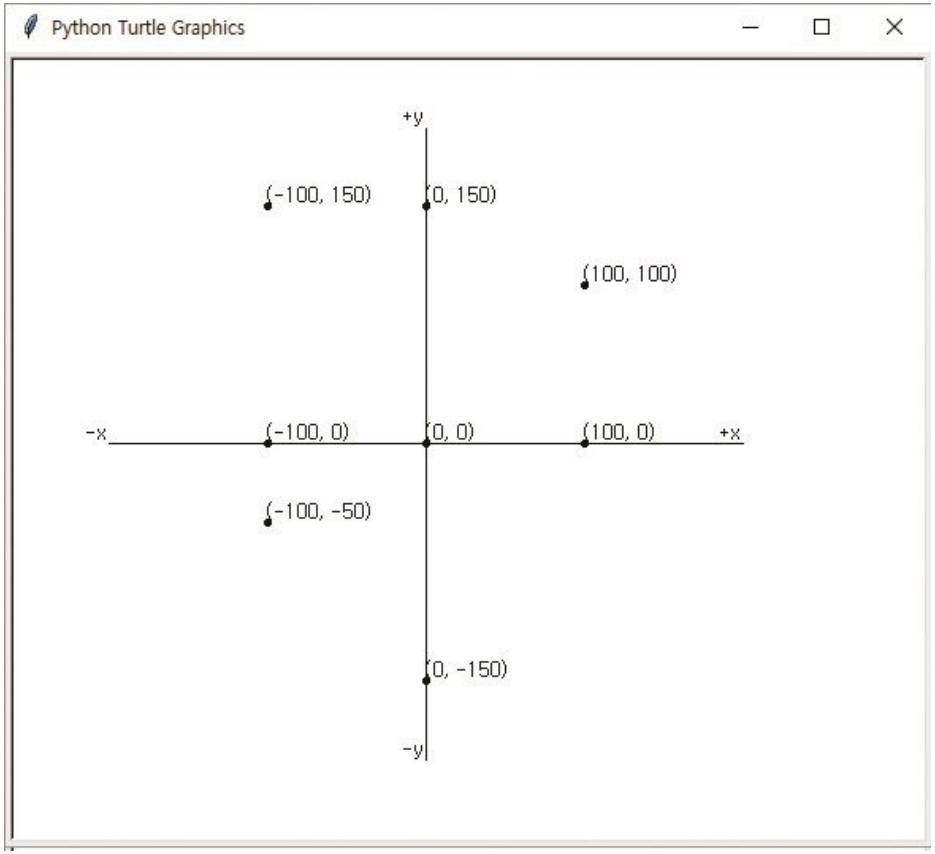
02

슬라이드 2. 좌표

» 거북이 그래픽 창의 좌푯값을 알면 t.goto(x, y) 명령으로 '한 번에' 거북이를 원하는 곳으로 이동시킬 수 있음. 이때 거북이 꼬리의 펜을 내리면(down) 이동한 경로를 따라 선이 그려지고, 펜을 올리면(up) 선이 그려지지 않고 거북이만 이동함.

거북이 그래픽 창에서 한 점의 좌표는 숫자 두 개로 표현할 수 있음. 왼쪽과 오른쪽 같은 화면의 수평 방향을 x 값으로 표시하고, 위와 아래 같은 수직 방향을 y 값으로 표시함.

```
>>> import turtle as t
>>> t.pos()
(0.00, 0.00)
>>> t.goto(100, 50)
>>> t.pos()
(100.00, 50.00)
>>>
```

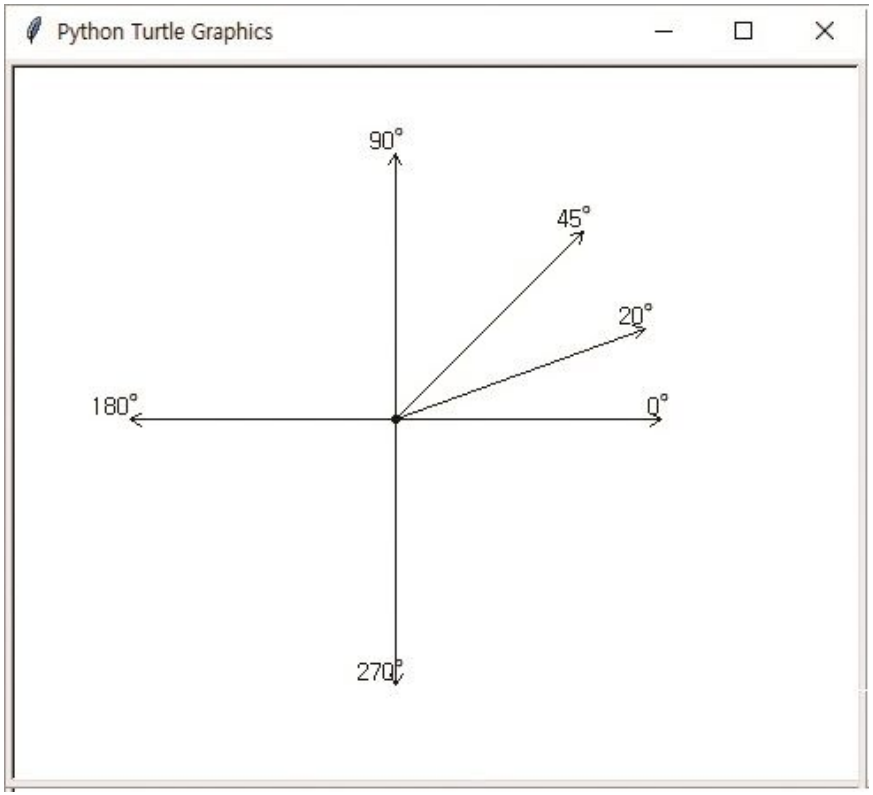


03

슬라이드 3. 각도

» left와 right 함수(명령)는 거북이를 현재 방향에서 원하는 각도만큼 각각 왼쪽과 오른쪽으로 회전시킴. 거북이 대포 프로젝트에서 거북이의 각도를 조절하려면 left와 right 함수뿐만 아니라 heading과 setheading 함수도 이용함.

heading()은 현재 거북이가 바라보는 각도를 구하고, setheading(ang)은 거북이가 특정 각도를 바라보도록 회전시킴.



04

슬라이드 4. 글자 쓰기

» t.write 함수는 현재 거북이가 있는 곳에 문자열(문장)을 쓰는 기능

```
t.write("문자열", False, "center", ("", 15))
```

거북이는 위치를 옮기지 않고(False), 현재 위치에서 문장을 가운데 정렬("center")하여 출력하겠다는 뜻.
마지막으로 전달된 인자 ("", 15)는 글자크기를 15로 출력하라는 의미

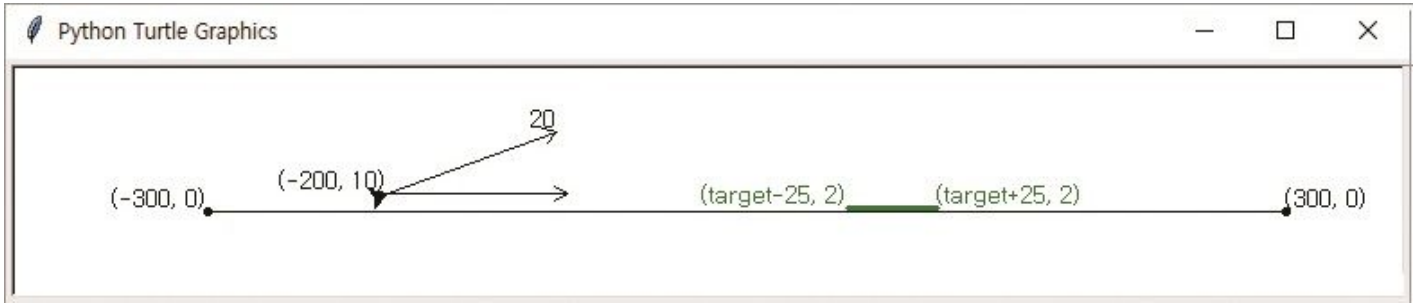
05

5. 프로젝트 구조

- turn_up/turn_down 함수
  사용자가 ↑ 와 ↓ 를 누르면 대포 각도를 조절

- fire 함수
  사용자가 SpaceBar 를 누르면 작동하는 함수. 거북이 대포를 발사하고 대포가 땅에 닿으면 목표 지점을 맞혔는지 확인하여 문자를 출력. 거북이 대포 게임에서 핵심 역할을 하는 함수임.

- 게임 준비 및 실행 부분
  땅, 목표 지점, 대포 위치를 지정하고 화면에 그림. onkeypress 함수를 사용하여 사용자가 키보드를 누르면 각 키에 맞게 turn_up, turn_down, fire 함수가 실행되도록 예약함



05

슬라이드 5. 프로젝트 구조

» 거북이 대포 게임 만들기

```
import turtle as t
import random

def turn_up():                  # ↑ 를 눌렀을 때 호출되는 함수
t.left(2)                       # 거북이를 왼쪽으로 2도 돌림

def turn_down():                # ↓ 를 눌렀을 때 호출되는 함수
t.right(2)                      # 거북이를 오른쪽으로 2도 돌림

def fire():                     # SpaceBar 를 누르면 거북이 대포를 발사함
    ang = t.heading()           # 현재 거북이가 바라보는 각도를 기억함
    while t.ycor() > 0:         # 거북이가 땅 위에 있는 동안 반복함
        t.forward(15)           # 15만큼 앞으로 이동함
        t.right(5)              # 오른쪽으로 5도 회전함
```

5. 프로젝트 구조

while 반복문을 빠져나오면 거북이가 땅에 닿은 상태임

```
d = t.distance(target, 0)                    # 거북이와 목표 지점과의 거리를 구함
t.sety(random.randint(10, 100))              # 성공 또는 실패를 표시할 위치를 지정함
if d < 25:                                   # 거리 차이가 25보다 작으면 목표 지점에 명중한
                                               것으로 처리함
    t.color("blue")
    t.write("Good!", False, "center", ("", 15))
else:                                        # 그렇지 않으면 실패한 것으로 처리함
    t.color("red")
    t.write("Bad!", False, "center", ("", 15))
t.color("black")                             # 거북이 색을 검은색으로 되돌림
t.goto(-200, 10)                             # 거북이 위치를 처음 발사했던 곳으로 되돌림
t.setheading(ang)                            # 각도도 처음 기억해 둔 각도로 되돌림
```

05

5. 프로젝트 구조

```
# 주의 : 여기서부터는 들여쓰기를 하지 않아도 됨

# 땅을 그립니다.
t.goto(-300, 0)
t.down()
t.goto(300, 0)

# 목표 지점을 설정하고 그림
target = random.randint(50, 150)  # 목표 지점을 50~150 사이에 있는 임의의 수로 지정
t.pensize(3)
t.color("green")
t.up()
t.goto(target - 25, 2)
t.down()
t.goto(target + 25, 2)
```

05

5. 프로젝트 구조

```
# 거북이 색을 검은색으로 지정하고 처음 발사했던 곳으로 되돌림

t.color("black")
t.up()
t.goto(-200, 10)
t.setheading(20)


# 거북이가 동작하는 데 필요한 설정을 함.

t.onkeypress(turn_up, "Up")          # ↑ 를 누르면 turn_up 함수를 실행
t.onkeypress(turn_down, "Down")      # ↓ 를 누르면 turn_down 함수를 실행
t.onkeypress(fire, "space")          # SpaceBar 를 누르면 fire 함수를 실행
t.listen()                           # 거북이 그래픽 창이 키보드 입력을 받도록 함
```

05

5. 프로젝트 구조

» 실행결과

05

5. 프로젝트 구조

거북이 대포가 날아가는 모양

이 프로젝트의 대포는 엄밀하게 말하면 '곡사포'가 아닙니다. 실제로 대포를 발사하면 포물선과 비슷한 곡선을 그리면서 날아갑니다. 이 프로그램의 대포는 매번 15만큼 앞으로 이동하고 5°씩 오른쪽으로 움직이므로 포물선보다는 원에 가까운 형태입니다(정확히는 72각형이지만, 원에 가까워 보입니다). 포물선을 정확히 그리려면 수학 계산이 필요한데, 그러려면 프로그램이 너무 복잡해질 수 있기 때문에 단순하게 바꾼 것이라고 이해하면 됩니다.