CHAPTER 3.
프로그램의 흐름 제어하기

section50. 그래픽 – turtle 모듈 사용하기

step 1.
알고가기

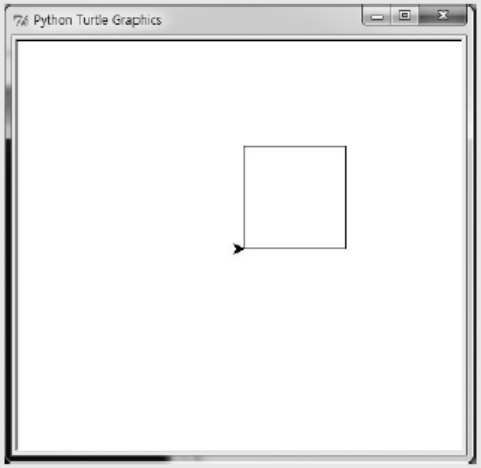
- 핵심내용
  - turtle 모듈은 그래픽을 그릴 수 있도록 파이썬이 제공하는 내장모듈이다.

- 코딩하기 전에
  - 잠시 쉬어가는 의미에서 파이썬이 제공하는 그래픽 모듈 거북이(turtle)를 한 번 사용해본다.
  - 거북이 모듈은 간단한 그림판 같은 기능을 제공하며 사용법도 쉽다.

section50. 그래픽 - turtle 모듈 사용하기

step 2.
코딩

```
1:  # section_050.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:
7:  t.forward(100)
8:  t.left(90)
9:  t.forward(100)
10: t.left(90)
11: t.forward(100)
12: t.left(90)
13: t.forward(100)
14: t.left(90)
```



section50. 그래픽 – turtle 모듈 사용하기

step 3.
코드분석

- turtle 모듈
  - 파이썬의 내장 모듈로 별도의 준비 없이 바로 import해서 사용할 수 있다.
- Pen() 메서드
  - 캔버스 화면을 만들어주는 기능을 가졌다.
  - 실행 후 캔버스 화면에 대한 사용권을 반환한다.
- forward() 메서드
  - 괄호 안의 숫자(픽셀단위)만큼 화살표 방향으로 전진하도록 한다.
- left() 메서드
  - 왼쪽(시계 반대방향)으로 회전시키는 기능이다.

section50. 그래픽 – turtle 모듈 사용하기

step 4.

3줄 요약

• 3줄 요약 •

- turtle 모듈은 간단한 그래픽을 그리는데 도움이 되는 코드들의 모음이다.
- Pen() 메서드는 캔버스를 생성함과 동시에 캔버스에 대한 사용권을 생성한다.
- 화살표가 이동하는 단위는 픽셀이고, 픽셀은 화면을 구성하는 단위이다.

section51. 조건문 – if문

step 1.
알고가기

- 핵심내용
  - if문 블록은 조건식의 판단 결과가 참(True)일 때만 실행한다.

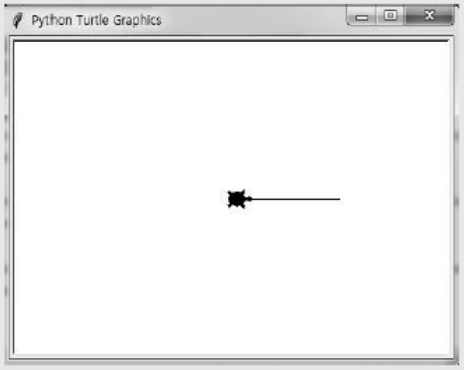
- 코딩하기 전에
  - if문을 **조건문**이라고 하는데 조건식의 결과가 참인지 거짓인지에 따라 실행할 명령을 따로 정해줄 수 있다.
  - **조건식**에는 언제나 참 또는 거짓으로 결정되는 식만 넣을 수 있다.
  - **블록**이란 프로그래밍 명령문들을 모아둔 것을 말한다. 파이썬에서는 블록을 구분할 때 들여쓰기를 이용한다.
  - **들여쓰기**를 하는 방법은 스페이스 키를 이용해 공백을 4칸 넣거나 탭(tab) 키를 한 번 사용한다.

section51. 조건문 – if문

step 2.
코딩

```
1:  # section_051.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  rabbit = 50
9:  carrot = 0
10:
11: if rabbit :
12:     t.forward(100)
13: if carrot :
14:     t.forward(100)
15: t.backward(100)
```

if문 끝에 콜론(:)은 꼭 써줘야 해.



section51. 조건문 – if문

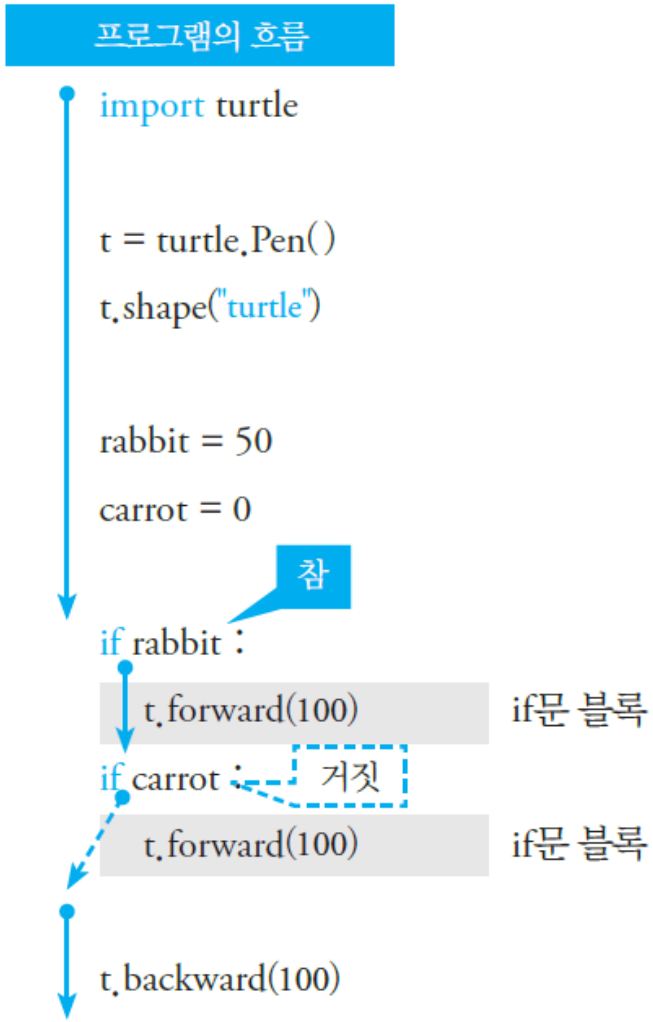
step 3.
코드분석

- if문
  - ‘만약’ 조건식이 참 (True) 이면 if문 블록의 명령들을 실행하고, 거짓(False)이면 아무것도 실행하지 않고 if문을 벗어난다.
- if rabbit:
  - 만약 rabbit이 참이라면 if문 블록을 실행한다. 그렇지 않으면(거짓이라면) if문 블록을 실행하지 않는다.
  - rabbit은 숫자 50이 저장되어 있는 정수형 변수이다.
  - 파이썬에서 숫자 0은 거짓이고 나머지 모든 숫자는 참이다.

section51. 조건문 – if문

step 3.
코드분석

- 파이썬 인터프리터는 코드의 1번 줄부터 차례대로 실행한다.
- if문을 만나면 조건식의 참/거짓을 판단한다.
- 참이면 if문 블록을 실행하고,
- 거짓이면 if문 블록을 실행하지 않는다.



section51. 조건문 – if문

step 4.

3줄 요약

• 3줄 요약 •

- 조건문은 if를 이용하고, 조건식의 참/거짓을 판단하는 것이 가장 중요하다.
- 각 자료형 별로 거짓인 경우를 알아두자.
- 블록은 반드시 들여쓰기를 해야 한다.

section52. 조건문 – if와 비교 연산자

step 1.
알고가기

- 핵심내용
  - 비교연산은 두 값을 비교하여 참 또는 거짓을 알려준다.

- 코딩하기 전에
  - if문에서 가장 결정적인 부분은 조건식이다.
  - 조건식에서 많이 사용되는 연산자에 비교 연산자가 있다.

기호	의미
x 〉 y	x값이 y값보다 크면 True, 그 외에는 False
x 〈 y	x값이 y값보다 작으면 True, 그 외에는 False
x 〉= y	x값이 y값보다 크거나 같으면 True, 그 외에는 False
x 〈= y	x값이 y값보다 작거나 같으면 True, 그 외에는 False
x == y	x와 y의 값이 같으면 True, 그 외에는 False
x != y	x와 y의 값이 다르면 True, 그 외에는 False

section52. 조건문 – if와 비교 연산자

step 2.

코딩

```
1:  # section_052.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  side1 = 100
9:  side2 = 50
10: if side1 > 0 :
11:     t.forward(side1)
12:
13: if side1 >= side2 :
14:     t.up()
15:     t.forward(side2)
16:     t.down()
17:     t.forward(side1)
18:
19: if 60 < side2 <= side1 :
20:     t.reset()
```

section52. 조건문 – if와 비교 연산자

step 3.
코드분석

- if side1 > 0:
  - if문의 실행과정은 다음과 같다.
  1. if문의 조건식(side1 > 0)을 평가한다.
  2. side1의 값이 100이므로 조건식은 100>0과 같다.
  3. 100>0은 참(True)이므로 결과적으로 조건식은 참이 된다.
  4. if문 블록의 코드(t.forward(side1))를 실행한다.
- if 60 < side2 <= side1:
  - 60 < 50 <= 100에서 60<50은 거짓이다.
  - 따라서 이 조건식은 거짓(False)이 된다.
  - if문 블록은 실행되지 않는다.

section52. 조건문 – if와 비교 연산자

step 4.

3줄 요약

• 3줄 요약 •

- 비교 연산은 두 값을 비교하기 위한 연산이다.
- 비교 연산의 결과는 반드시 참 또는 거짓이다.
- 비교 연산자의 우선 순위는 산술 연산자보다 낮다.

section53. 조건문 – if와 논리 연산자

step 1.
알고가기

- 핵심내용
  - 논리 연산자는 참과 거짓을 위한 연산자이다.

- 코딩하기 전에
  - 조건식에 많이 사용되는 또 다른 연산자가 논리 연산자이다.
  - 논리 연산자는 3 종류가 있으며, 피연산자는 논리값(True/False)이다.
  - 표에서 피연산자 x, y는 모두 참/거짓으로 결정될 수 있는 것이어야 한다.

기호	의미
x and y	x와 y가 모두 True이면 True, 그 외에는 False
x or y	x와 y 중 하나라도 True이면 True, 그 외에는 False
not x	x가 True이면 False, x가 False이면 True, 즉 x의 결과를 반대로 바꿔준다.

section53. 조건문 – if와 논리 연산자

step 2.
코딩

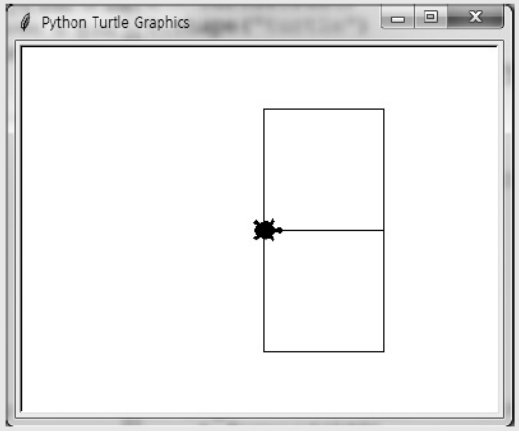
```
1:  # section_053.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  legs = input('거북이의 다리 개수는?')
9:  legs = int(legs)
10: skin = input('거북이의 색은?')
11:
12: if legs == 4 and skin == 'green' :
13:     t.forward(100)
14:     t.left(90)
15:     t.forward(100)
16:     t.left(90)
17:     t.forward(100)
18:     t.left(90)
19:     t.forward(100)
20:     t.left(90)
```

```
21: if legs == 4 or skin == 'black' :
22:     t.forward(100)
23:     t.right(90)
24:     t.forward(100)
25:     t.right(90)
26:     t.forward(100)
27:     t.right(90)
28:     t.forward(100)
29:     t.right(90)
```

section53. 조건문 – if와 논리 연산자

step 2.
코딩

```
거북이의 다리 개수는? 4
거북이의 색은? green
```



section53. 조건문 – if와 논리 연산자

step 3.
코드분석

- legs ==4 and skin == 'green'
  - 조건식에 세 개의 연산자(==, and, ===)가 있다.
  - 우선순위 원칙에 따라 비교 연산자를 먼저 실행한다.
  - 따라서 legs == 4와 skin == 'green'을 먼저 실행한다.
  - legs==4는 참(True), skin='green'도 참이다.
  - x and y 형태에서 참 and 참이므로, 이 논리 연산의 결과도 참이다.
- 논리 연산자의 우선순위
  - not > and > or 순으로 순위가 높다.
  - 논리 연산자는 비교 연산자보다 우선 순위가 낮다.

section53. 조건문 – if와 논리 연산자

step 4.

3줄 요약

• 3줄 요약 •

- 논리 연산은 두 논리값(참/거짓)을 위한 연산이다.
- 논리 연산의 결과는 반드시 참 또는 거짓이다.
- 논리 연산자의 우선 순위는 비교 연산자보다 낮다.

section54. 조건문 – in과 not in 연산자

step 1.
알고가기

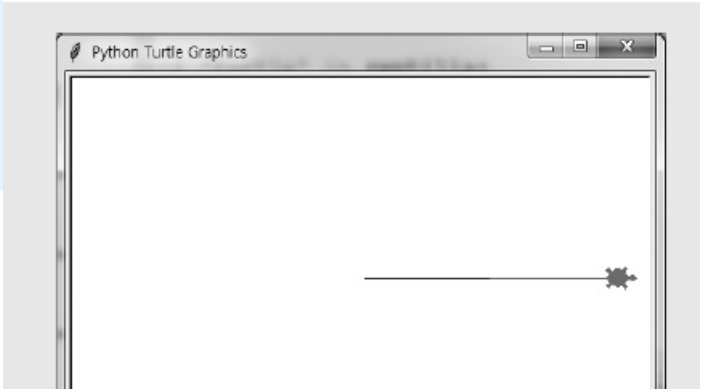
- 핵심내용
  - in은 존재하는지, not in은 존재하지 않는지를 확인해주는 연산자이다.

- 코딩하기 전에
  - in과 not in 연산자도 연산의 결과가 참/거짓으로 나오기 때문에 조건식에서 종종 사용된다.
  - in은 값이 객체 안에 존재하면 참, 그렇지 않으면 거짓을 반환한다.
  - not in은 값이 객체 안에 존재하지 않으면 참, 그렇지 않으면 거짓을 반환한다.

section54. 조건문 – in과 not in 연산자

step 2.
코딩

```
1:  # section_054.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  reptilia = ['turtle', 'crocodile', 'iguana', 'chameleon']
9:  amphibian = ['frog', 'toad', 'axolotl']
10:
11: if 'turtle' in reptilia:
12:     t.color('gray')
13:     t.forward(100)
14:
15: if 'turtle' not in amphibian:
16:     t.color('light gray')
17:     t.forward(100)
```



section54. 조건문 – in과 not in 연산자

step 3.
코드분석

- 'turtle' in reptilia
  - 'turtle' 문자열이 reptilia 리스트에 존재하는가를 확인한다.
  - 리스트 reptilia 안에 'turtle'이 존재하므로 참(True)를 반환한다.
  - 즉, 조건식이 참이 되어 if문 블록이 실행된다.
- 'turtle' not in amphibian
  - 'turtle' 문자열이 amphibian 리스트에 존재하지 않는지를 확인한다.
  - 리스트 amphibian 안에 'turtle'이 존재하지 않으므로 참(True)를 반환한다.
  - 즉, 조건식이 참이 되어 if문 블록이 실행된다.

section54. 조건문 – in과 not in 연산자

step 4.

3줄 요약

• 3줄 요약 •

- x in s는 s 안에 x가 존재하는가를 묻고, 있으면 True, 없으면 False를 알려주는 연산자이다.
- x not in s는 s 안에 x가 없는지를 묻고, 없으면 True, 있으면 False를 알려주는 연산자이다.
- 두 연산의 결과는 모두 True 또는 False로 반환된다.

section55. 조건문 – if와 else문

step 1.
알고가기

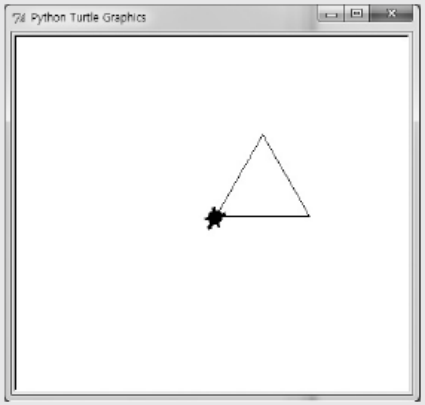
- 핵심내용
  - else 블록은 조건식이 거짓일 때 실행되는 코드이다.

- 코딩하기 전에
  - if문은 조건식이 참일 때 실행할 코드만 넣을 수 있다.
  - else문은 조건식이 거짓일 때 실행할 코드를 넣을 수 있는 문이다.
  - 따라서 조건식의 참인지 거짓인지에 따라 if문 블록과 else문 블록 중 단 한 블록만 실행된다.

section55. 조건문 – if와 else문

step 2.
코딩

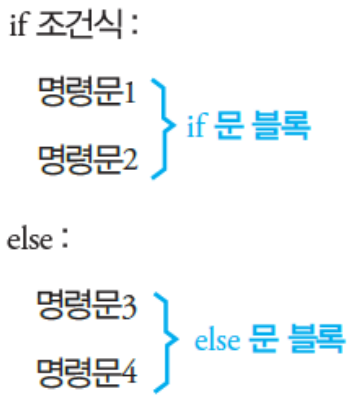
```
1:  # section_055.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  draw = True
9:  if draw :
10:     t.forward(100)
11:
12: if not draw :
13:     t.forward(-100)
14:
15: moyang = 3
16: if moyang == 4 :
17:     t.left(90)
18:     t.forward(100)
19:     t.left(90)
20:     t.forward(100)
21:     t.left(90)
22:     t.forward(100)
23: else :
24:     t.left(120)
25:     t.forward(100)
26:     t.left(120)
27:     t.forward(100)
```



section55. 조건문 – if와 else문

step 3.
코드분석

- if – else문



  - 조건식을 평가하여 참(True)이면 if문 블록을 실행한다.
  - 조건식을 평가하여 거짓(False)이면 else문 블록을 실행한다.
- if moyang == 4 :
  - moyang ==4 는 거짓이므로 else문 블록만 실행한다.

section55. 조건문 – if와 else문

step 4.

3줄 요약

• 3줄 요약 •

- if문의 조건식이 거짓일 때도 실행할 명령어를 지정하기 위해 else를 사용한다.
- else 블록은 조건식이 거짓일 때 실행할 코드들을 입력한다.
- if 블록은 참일 때만, else 블록은 거짓일 때만 실행되므로 if~else를 사용하면 조건식이 참이든 거짓이든 상관없이 실행되는 코드가 존재한다.

section56. 조건문 – if와 elif문

step 1.
알고가기

- 핵심내용
  - if문에 조건식을 2개 이상 사용하고 싶다면 elif 문을 사용한다.

- 코딩하기 전에
  - if문은 조건이 참일 때 실행할 코드만 지정할 수 있었고, if-else문은 조건이 참일 때와 거짓일 때 각각 실행할 수 있는 코드를 지정할 수 있다.
  - **elif**문은 조건이 두 개 이상일 때 사용하는 코드이다.

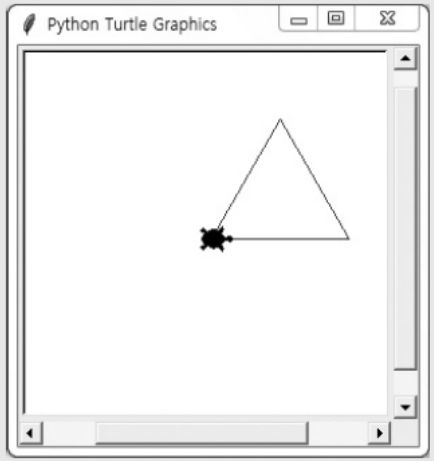
section56. 조건문 – if와 elif문

step 2.
코딩

```
1:  # section_056.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  moyang = int(input('색을 선택하세요(1:파랑,
            2:빨강, 3:노랑): '))
9:
10: if moyang == 1 :
11:     t.pencolor("blue")
12: elif moyang == 2 :
13:     t.pencolor("red")
14: elif moyang == 3 :
15:     t.pencolor("yellow")
16: else :
17:     t.pencolor("black")
18:
19: t.forward(100)
20: t.left(120)
21: t.forward(100)
22: t.left(120)
23: t.forward(100)
24: t.left(120)
```

input( ) 함수로 받은 값은 문자열형이야.

색을 선택하세요(1:파랑, 2:빨강, 3:초록): 4



section56. 조건문 – if와 elif문

step 3.

코드분석

```
if moyang == 1 :
    t.pencolor("blue")
elif moyang == 2 :
    t.pencolor("red")
elif moyang == 3 :
    t.pencolor("yellow")
else :
    t.pencolor("black")
```

- 변수 moyang이 가진 값에 따라 실행 결과가 달라진다.
  - if 조건식을 판단하여 참이면 blue가 출력
  - (만약 거짓이면) 첫 번째 elif 조건식을 판단하여 참이면 red 출력
  - (만약 첫 번째 elif 조건식이 거짓이면) 두 번째 elif 조건식을 판단하여 참이면 yellow 출력
  - 이것도 거짓이면 else 블록이 실행되어 black 출력

section56. 조건문 – if와 elif문

step 4.

3줄 요약

• 3줄 요약 •

- elif문을 이용하면 여러 개의 조건식을 추가할 수 있다.
- if 블록, elif 블록, else 블록 중 단 하나의 블록만 실행된다.
- else 블록에는 모든 조건식이 거짓일 때 실행할 코드를 삽입한다.

section57. 반복문 – while문

step 1.
알고가기

- 핵심내용
  - while문의 조건식이 참인 동안 while 블록이 반복 실행된다.

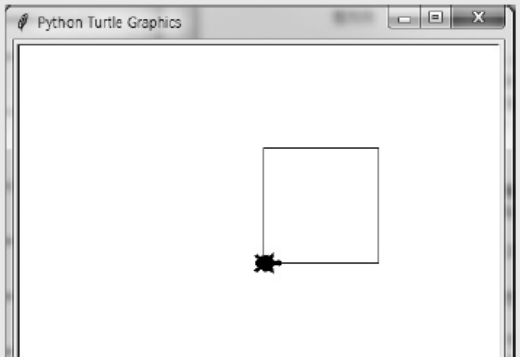
- 코딩하기 전에
  - 중복되는 코드는 코드 분석을 어렵게 하고 비효율적이다.
  - 어떤 코드를 반복적으로 실행하고 싶을 때 반복문을 사용하는데, 대표적인 반복문 중 하나가 while문이다.

section57. 반복문 – while문

step 2.
코딩

```
1:  # section_057.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  count = 0
9:
10: while count < 4 :
11:     t.forward(100)
12:     t.left(90)
13:
14:     count = count + 1
15:
```

while문 끝에 반드시 콜론(:)을 넣어야 해.



section57. 반복문 – while문

step 3.
코드분석

```
while count < 4 :
    t.forward(100)
    t.left(90)

    count = count + 1
```

while문 실행 과정

1. while문의 조건식을 평가한다.
2. 조건식이 참이면 while문 블록을 실행한다.
3. 조건식이 거짓이면 while문 블록을 실행하지 않는다.

- count의 값이 0이므로 count<4는 참이다.
- 조건식이 참이므로 while문 블록을 실행한다.
- count = count + 1에서 변수 count의 값이 1이 된다.
- count의 값이 1이므로 count<4는 참이다.
- 조건식이 참이므로 while문 블록을 실행한다.
- count = count + 1에서 변수 count의 값이 2가 된다.
- 이 과정을 조건식(count<4)이 거짓이 될때까지 반복한다.

section57. 반복문 - while문

step 4.

3줄 요약

• 3줄 요약 •

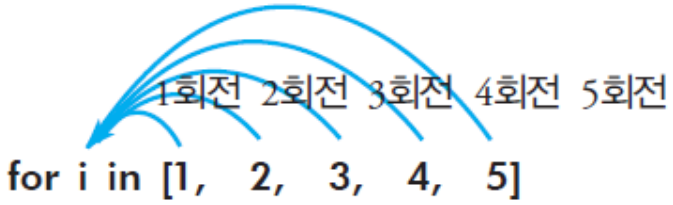
- while문은 특정 코드를 반복 실행하고 싶을 때 사용한다.
- while문 조건식이 참일 때만 while 블록이 실행된다.
- 조건식을 조절하여 while 블록 실행 횟수를 조절할 수 있다.

section58. 반복문 – for문

step 1.
알고가기

- 핵심내용
  - for문은 시퀀스형 객체의 항목 수만큼 반복한다.

- 코딩하기 전에
  - for문은 while문과 같이 블록을 반복하기 위한 명령어



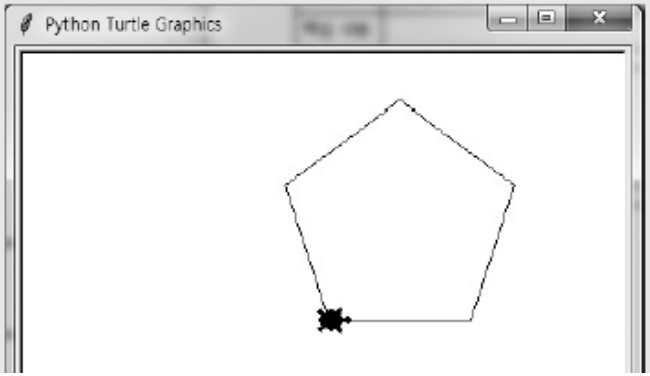
  - 시퀀스형 객체와 함께 사용되며 시퀀스형 객체의 항목 수 만큼 반복시킬 수 있다.

section58. 반복문 – for문

step 2.
코딩

```
1:  # section_058.py
2:
3:  import turtle
4:
5:  five = turtle.Pen()
6:  five.shape("turtle")
7:
8:  for i in [1, 2, 3, 4, 5]:
9:      five.forward(100)
10:     five.left(72)
11:
```

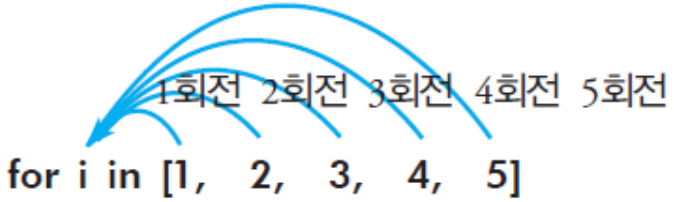
for문 끝에도 반드시 콜론(:)을 넣어야 해.



section58. 반복문 – for문

step 3.
코드분석

```
for i in [1, 2, 3, 4, 5]:
    star.forward(100)
    star.left(72)
```



- 1회전: 리스트의 첫 번째 항목인 1이 변수 i 에 저장 되면서 for문 블록을 실행한다.
- 2회전: 리스트의 첫 번째 항목인 2가 변수 i 에 저장 되면서 for문 블록을 실행한다.
- 3회전: 리스트의 첫 번째 항목인 3이 변수 i 에 저장 되면서 for문 블록을 실행한다.
- 4회전: 리스트의 첫 번째 항목인 4가 변수 i 에 저장 되면서 for문 블록을 실행한다.
- 5회전: 리스트의 첫 번째 항목인 5가 변수 i 에 저장 되면서 for문 블록을 실행한다.

section58. 반복문 - for문

step 4.

3줄 요약

• 3줄 요약 •

- for문은 시퀀스형 객체의 항목 수만큼 반복시킬 수 있다.
- 매 회전마다 for문 변수는 시퀀스형 객체의 항목을 순서대로 저장한다.
- for문은 실행 횟수가 정해져있을 때, while문은 정해져 있지 않을 때 주로 사용한다.

section59. 반복문 – for문과 range() 함수

step 1.
알고가기

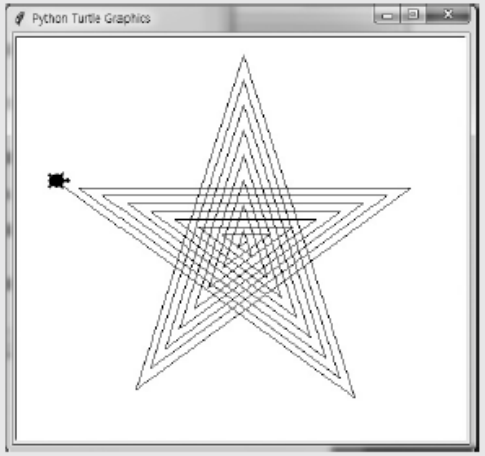
- 핵심내용
  - range()함수를 for문에서 이용하면 아주 많은 반복을 쉽게 구현할 수 있다.

- 코딩하기 전에
  - for문은 시퀀스형 객체랑 어울려 사용된다.
  - 문제는 1천번, 1만번 반복하기 위해 리스트나 튜플을 작성하는 것은 무리가 있다.
  - range() 함수는 이런 문제를 해결해주는 유용한 내장함수이다.
  - range(1000)은 0, 1, 2, ... 999를 만들어서 for 문 변수에 전달하는 식이다.
  - 때문에 함수 하나로 반복횟수를 설정할 수 있다.

section59. 반복문 – for문과 range() 함수

step 2.
코딩

```
1:  # section_059.py
2:
3:  import turtle
4:
5:  star = turtle.Pen()
6:  star.shape("turtle")
7:
8:  for k in range(40) :
9:      star.forward(k * 10)
10:     star.right(144)
11:
```



section59. 반복문 – for문과 range() 함수

step 3.
코드분석

- range(시작값, 끝값, 증가값)
  - 시작값과 증가값은 생략할 수 있다.
  - 끝값은 포함하지 않는다.
  - 예) range(0, 5)는 0, 1, 2, 3, 4 를 사용할 수 있다.
  - 예) range(0, 5)는 range(5)와 같다.
- for k in range(40):
  - for문을 40회 반복하라는 명령이다.
  - 이때 k값은 1회전 0에서 시작해서 40회전 39까지 증가할 것이다.
- list(range(10000))
  - 항목이 많은 리스트나 튜플을 만들 때 range() 함수를 사용하면 편리하다.

section59. 반복문 – for문과 range() 함수

step 4.

3줄 요약

• 3줄 요약 •

- range( ) 함수는 range 객체를 만들어준다.
- range(시작값, 종료값, 증가값)에서 종료값은 필수로 입력해야 한다.
- range( )로 만든 range 객체를 리스트나 튜플 등으로 변환할 수 있다.

section60. 반복문 – 중첩 for문

step 1.
알고가기

- 핵심내용
  - 중첩for문이란 for문 블록에 또 다른 for문이 들어있는 것을 말한다.

- 코딩하기 전에
  - for문 블록 안에서는 변수를 만들 수도 있고 if 문을 넣을 수도 있고 또 다른 반복문을 넣을 수도 있다.
  - 특히 반복문 안에 반복문을 넣으면 다채로운 알고리즘을 만들어낼 수가 있다.
  - 이렇게 for문 안에 for문이 들어있는 모양을 중첩 for문이라고 한다.

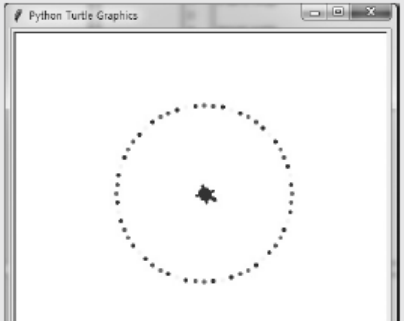
section60. 반복문 – 중첩 for문

step 2.
코딩

```
1:  # section_060.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:  t.up()
8:
9:  colors = ['red', 'green', 'blue', 'yellow', 'purple']
10: for c in colors :
11:     t.color(c)
12:     for i in range(12) :
13:         t.forward(100)
14:         t.dot()
15:         t.backward(100)
16:         t.right(30)
17:     t.right(6)
18:
```

for❷ 블록

for❶ 블록



section60. 반복문 – 중첩 for문

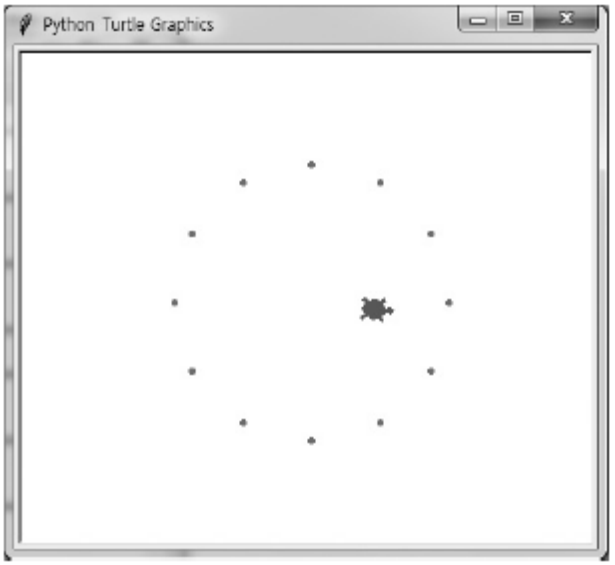
step 3.
코드분석

- 중첩 for문의 형태

```
for문 :
    명령어1
    명령어2
    for문 :
        명령어3
        명령어4
        명령어5
    명령어6
```

for②문 블록 : 명령어3, 명령어4, 명령어5

for①문 블록 : 명령어1, 명령어2, for문 :, 명령어3, 명령어4, 명령어5, 명령어6



- '명령어2'를 실행하고 '명령어6'을 실행하기 전에 for②문 블록이 실행완료 된다.
- for②문이 한 번 실행되고 나면 오른쪽 그림과 같은 결과가 나온다.
- for②문 블록의 총 회전수는 (for①문 회전수 * for②문 회전수)이다.

section60. 반복문 – 중첩 for문

step 4.

3줄 요약

• 3줄 요약 •

- 중첩 for문은 for문 안에 또 다른 for문을 사용하는 것을 말한다.
- for문 블록에는 변수 할당이나 if문, for문, while문 등 다양한 코드들이 삽입될 수 있다.
- for①문 블록이 한 번 실행될 때마다 for②문 블록이 모두 실행 완료되어야 for①문이 다음 회전으로 넘어갈 수 있다.

section61. 반복문 중지 – break

step 1.
알고가기

- 핵심내용
  - break는 반복문을 중지하기 위한 명령어이다.

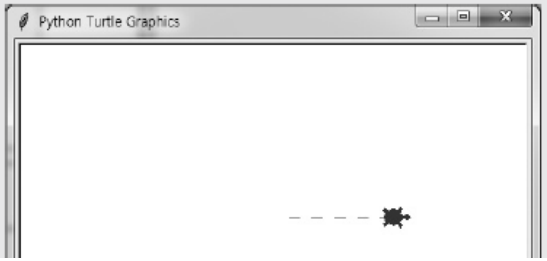
- 코딩하기 전에
  - while문이나 for문 등의 반복문은 기본적으로 내부 블록의 모든 코드를 조건이 맞는 동안 계속 반복한다.
  - 이러한 흐름을 바꿔주는 명령 중 하나가 break이다.
  - break : 반복문 안에서 break를 만나면 반복을 중지하고 무조건 반복문을 탈출한다.

section61. 반복문 중지 – break

step 2.
코딩

```
1:  # section_061.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  for item in range(30):
9:      if item%2:
10:         t.pendown()
11:     else:
12:         t.penup()
13:
14:     if item == 10:
15:         break
16:
17:     t.forward(10)
```

break가 실행되면 반복문을 중지하게 돼.



section61. 반복문 중지 – break

step 3.
코드분석

- break
  - break는 반복문을 중지하고 빠져나가는 기능이다.
  - break문을 만나면 반복문을 즉시 빠져나가기 때문에 보통 if문 안에서 특정 조건을 만족할 때만 실행되도록 작성한다.
  - 중첩 반복문에서 break를 사용하면 가장 가까운 반복문만 탈출한다.

```
if item == 10:
    break
```

- if문 조건식(item==10)이 참일 경우만 break가 실행되며, 그 외에는 break가 실행되지 않는다.

section61. 반복문 중지 – break

step 4.

3줄 요약

• 3줄 요약 •

- break는 반복문을 탈출하는 역할을 한다.
- break를 만나면 반복문을 즉시 탈출하기 때문에 보통 if문 등을 이용해 특정 조건 하에서만 실행되도록 한다.
- 중첩 반복문에서 사용하면 가장 가까운 반복문을 탈출한다.

section62. 반복문 – continue

step 1.
알고가기

- 핵심내용
  - continue는 프로그램의 흐름을 반복문의 시작점으로 즉시 이동시킨다.

- 코딩하기 전에
  - break문과 함께 반복문의 흐름을 바꿔주는 명령어가 continue문이다.
  - continue : 반복문 안에서 continue를 만나면 블록의 나머지 부분을 실행하지 않고, 무조건 반복문의 시작점으로 이동한다.
  - break문과 마찬가지로 continue문 아무데나 쓰면 안 되고, if문과 같이 특정 조건을 만족시킬 때만 사용하도록 코딩한다.

section62. 반복문 - continue

step 2.
코딩

```
1:  # section_062.py
2:
3:  import turtle
4:
5:  t = turtle.Pen()
6:  t.shape("turtle")
7:
8:  for item in range(30):
9:      if item%2:
10:         t.penup()
11:         t.forward(10)
12:         t.pendown()
13:         continue
14:
15:     t.forward(10)
```

8번째 줄이 반복문의 시작점이야.

section62. 반복문 – continue

step 3.
코드분석

- continue
  - break가 실행되면 반복문을 즉시 종료하는데 반해, continu가 실행되면 반복문의 시작점으로 흐름을 이동시킨다는 점에서 차이가 있다.

```
for item in range(30):
    if item%2:
        t.penup()
        t.forward(10)
        t.pendown()
        continue
    t.forward(10)
```

  - for문은 총 30회전 할 수 있다.
  - if 조건식을 만족할 때(즉, item이 홀수일때) 그림이 그려지고 continue가 실행된다.
  - continue가 실행되면 프로그램의 흐름은 for문으로 이동하게 된다.

section62. 반복문 - continue

step 4.

3줄 요약

• 3줄 요약 •

- continue는 프로그램의 흐름을 반복문의 시작점으로 즉시 이동시킨다.
- continue는 보통 if문 등을 이용해 특정 조건 하에서만 실행되도록 한다.
- 중첩 반복문에서 사용하면 가장 가까운 반복문의 시작점으로 이동한다.