Chapter 06. 반복

목차

01

반복 구조

01. 반복 구조

[일상 생활 속 반복]

- 알람 소리에 잠에서 깨어 학교나 직장에 가기
- 자전거를 타거나 운동을 할 때 동작 반복하기

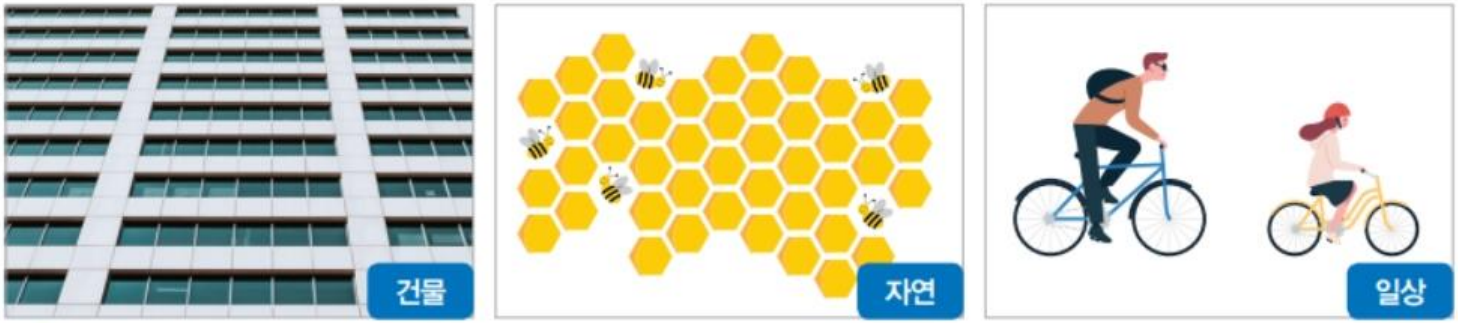


그림 6-1 일상에서의 패턴

01. 반복 구조

I. 일상에서 볼 수 있는 반복

실습 6-1 반복적인 규칙 찾기

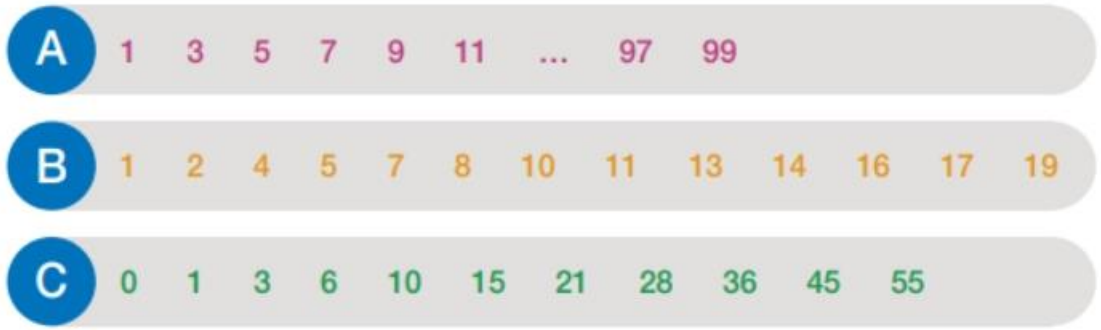


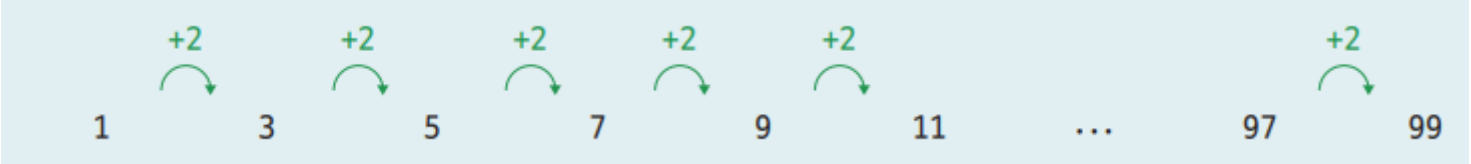
그림 6-2 숫자 속에서의 패턴

01. 반복 구조

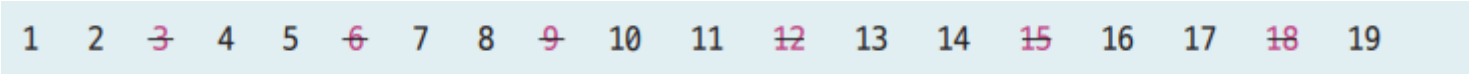
I. 일상에서 볼 수 있는 반복

실습 6-1 반복적인 규칙 찾기

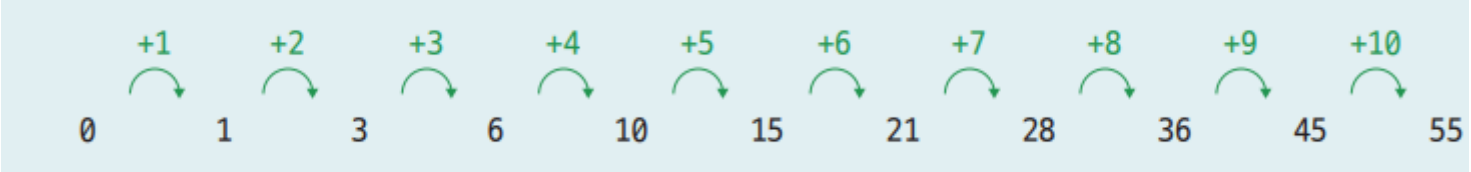
① A는 100보다 작은 자연수 중에서 홀수만 출력



② B는 20보다 작은 자연수 중에서 3의 배수를 빼고 출력



③ C는 0에서 10까지 정수를 차례대로 더한 값을 출력



01. 반복 구조

II. 반복 구조의 필요성

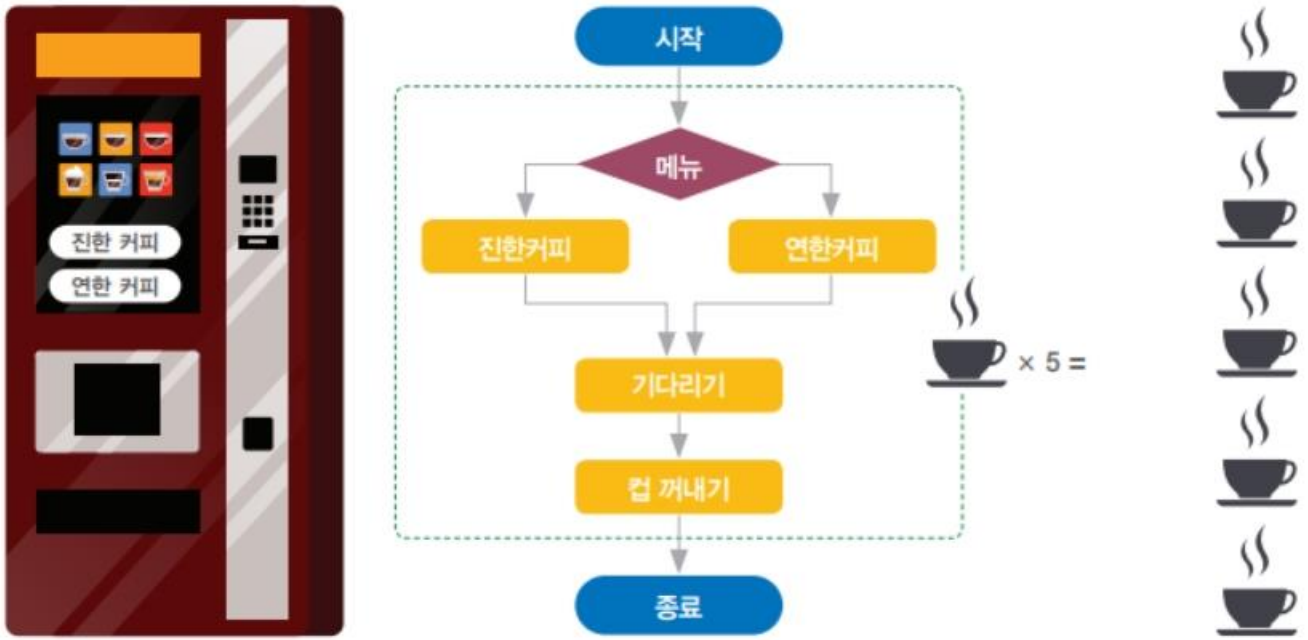


그림 6-3 커피머신의 반복 동작

01. 반복 구조

II. 반복 구조의 필요성

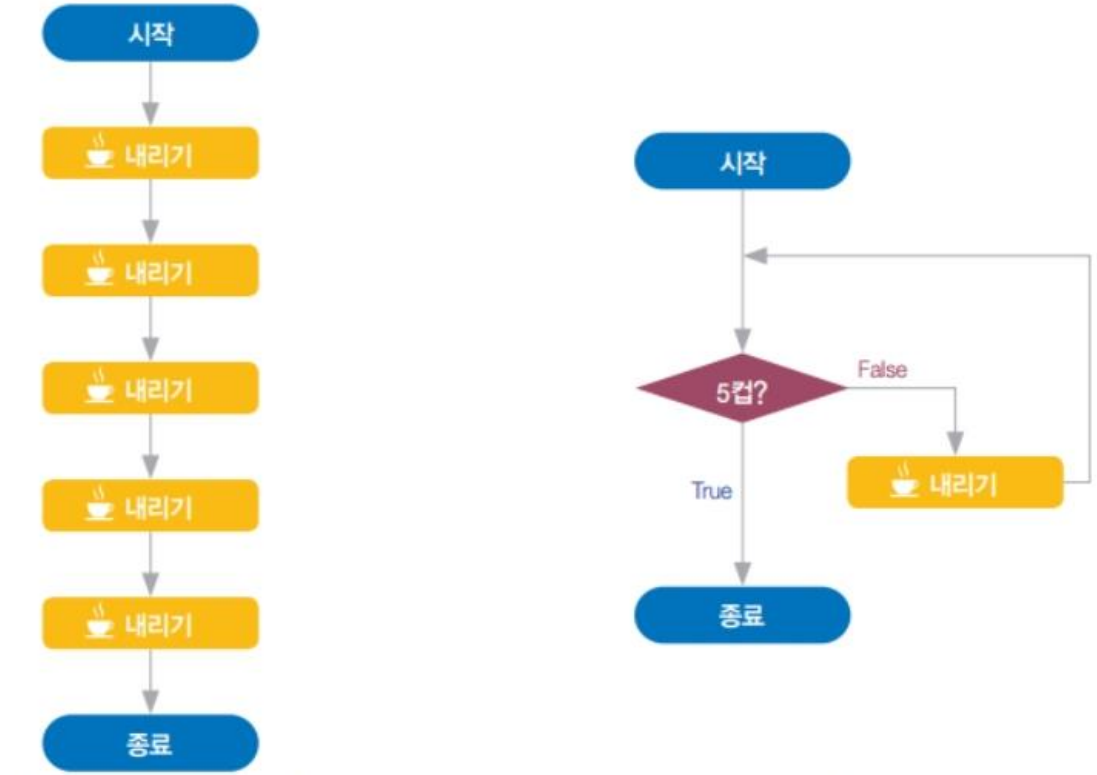


그림 6-4 순차 구조와 반복 구조 비교

- 반복 구조를 사용하면 정해진 횟수만큼 수행할 동작을 기술
- 반복 횟수가 더 많아지더라도 전체 코드가 길어지지 않음

02

반복문의 종류

02. 반복문의 종류

표 6-1 반복문의 종류

	조건 반복	횟수 반복	무한 반복
반복 기준	조건의 충족	반복 횟수	무한 실행
사용 예	지칠 때까지 반복하라	10번 반복하라	계속하라
반복문	while	for	while True

I. While 문

- 조건에 의해 반복을 제어하는 while 문은 조건식의 판단 결과에 따라 반복 여부를 선택

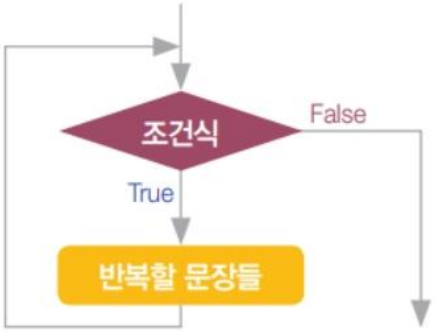


그림 6-6 while 문 구조

```
while 조건식 :
    반복할 문장들
```

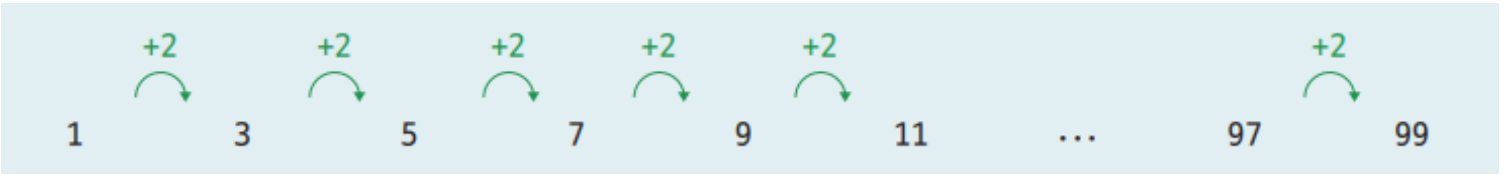
들여쓰기

02. 반복문의 종류

I. While 문

- 조건에 의해 반복을 제어하는 while 문은 조건식의 판단 결과에 따라 반복 여부를 선택

실습 6-2 while 문을 이용하여 규칙을 가진 숫자 출력하기 code06-02.py



①

```
01  x = 1                                  # 변수 x의 초기화
02
03  while x < 100 :                        # x의 값이 100보다 작으면 반복하고, 아니면 종료
04      print(x, end = " ")                # x의 값을 출력하고 줄 바꿈 대신 한 칸 띄움
05      x = x + 2                          # x를 2 증가시킴
```

②

```
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67
69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

02. 반복문의 종류

I. While 문

- 컴퓨터가 랜덤으로 정해 놓은 1부터 20 사이의 정수를 맞히는 게임
- 사용자가 입력한 수가 정해 놓은 값과 같으면 정답이라고 알려주고 프로그램을 종료
- 두 수가 다른 경우에는 숫자를 다시 입력받아 비교해서 더 큰지, 작은지를 화살표(↑, ↓)로 표시

그림 6-7 랜덤 게임 규칙과 실행 순서

실습 6-3 랜덤 숫자 맞히기 게임 만들기 code06-03.py

① random 모듈을 불러오고 random.randint()를 사용해 정수 값 하나를 추출하고 변수 answer에 저장, randint( )의 두 인수(1, 20)가 추출할 범위

```
01  import random                    # random 모듈 불러오기
02  answer = random.randint(1, 20)   # 1에서 20 사이의 랜덤 정수 저장
```

02. 반복문의 종류

I. While 문

실습 6-3 랜덤 숫자 맞히기 게임 만들기 code06-03.py

② 입력받은 값을 저장하기 위한 변수를 만들고, 정답이 나올 때까지 '입력 → 값의 비교 → 출력' 동작을 반복

```
03  number = 0                      # 입력하는 값을 저장할 변수
04
05  while number != answer:
06      number = int(input("숫자 입력(1~20) : "))
07
08      if number > answer:         # 입력한 값이 정답보다 큰 경우
09          print("↓")
10      elif number < answer:       # 입력한 값이 정답보다 작은 경우
11          print("↑")
12      else:
13          print("★정답★")
```

02. 반복문의 종류

I. While 문

실습 6-3 랜덤 숫자 맞히기 게임 만들기 code06-03.py

③ 작성한 프로그램을 파일로 저장하고 실행 결과를 확인

```
숫자 입력(1~20) : 10
↑
숫자 입력(1~20) : 13
↑
숫자 입력(1~20) : 16
*정답*
```

```
숫자 입력(1~20) : 6
↑
숫자 입력(1~20) : 13
↓
숫자 입력(1~20) : 11
*정답*
```

02. 반복문의 종류

II. 무한 반복과 반복의 제어

▪ 무한 반복문

- 반복할 문장들이 계속 실행
- 무한 반복을 끝내려면 키보드에서 Ctrl + C 를 눌러 프로그램을 강제로 종료

```
while True :
    반복할 문장들
```

조건식을 항상 참으로 기술하는 방법

실습 6-4 무한 반복하는 프로그램 만들기 code06-04.py

①

```
01  x = 0
02
03  while True :
04      x += 1                  # x의 값을 하나 증가
05      print(x, end = ' ')     # 한 줄로 나란히 출력
```

II. 무한 반복과 반복의 제어

▪ 무한 반복문

실습 6-4 무한 반복하는 프로그램 만들기 code06-04.py

② 프로그램을 실행하면 1부터 하나씩 증가한 값이 출력 / 종료를 위해 단축키 Ctrl + C 입력

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
68 69 70 71 72 73 74 75 76 77 78 79 80 81
Traceback (most recent call last):
    File "F:\code06-04.py", line 5, in <module>
        print(x, end = ' ')          # 한 줄로 나란히 출력
KeyboardInterrupt
```

02. 반복문의 종류

II. 무한 반복과 반복의 제어

▪ 반복의 제어

- 반복문 안에서 break와 continue를 사용하여 반복문을 끝내거나 반복할 문장을 건너뛰고 조건식 검사 위치로 이동하도록 프로그램 흐름을 제어
- break와 continue는 일반적으로 if 문과 함께 사용하여, 특정 조건이 만족할 때 반복문이 종료되거나 실행 순서의 변경 발생

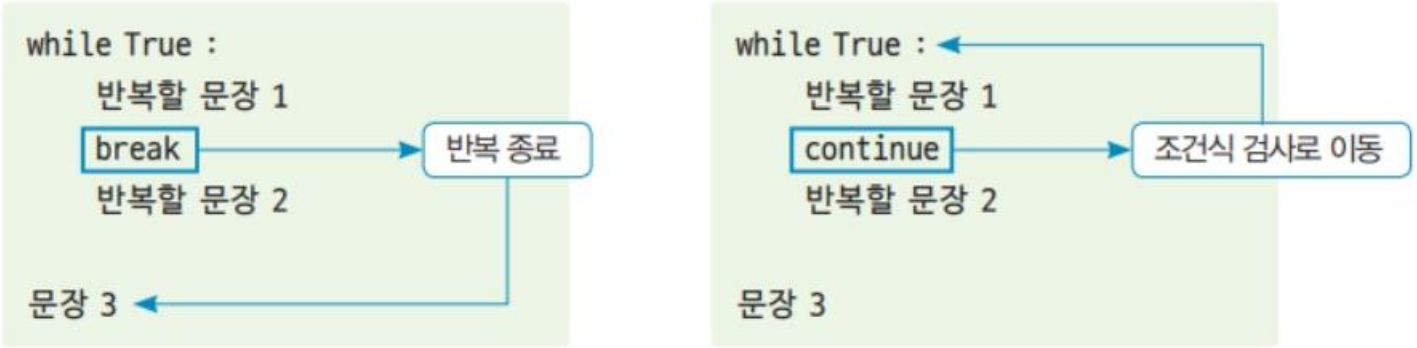


그림 6-8 break와 continue를 사용한 반복문의 제어 흐름

02. 반복문의 종류

II. 무한 반복과 반복의 제어

▪ 반복의 제어

실습 6-5 break와 continue를 사용하여 규칙을 가진 숫자 출력하기 code06-05.py

- 20보다 작은 자연수 중에서 3의 배수만 제외하고 출력
- 반복문의 흐름을 제어하는 break와 continue를 사용하여 출력 패턴에 알맞은 프로그램을 작성

1 2 ~~3~~ 4 5 ~~6~~ 7 8 ~~9~~ 10 11 ~~12~~ 13 14 ~~15~~ 16 17 ~~18~~ 19

```
01 x = 0
02
03 while True:
04     x = x + 1
05     if x >= 20:
06         break
07     if x % 3 == 0:
08         continue
09     print(x, end = ' ')
```

```
1 2 4 5 7 8 10 11 13 14 16 17 19
```

02. 반복문의 종류

II. 무한 반복과 반복의 제어

▪ 반복의 제어

실습 6-6 영화관 무인단말기로 메뉴 주문 후 종료하기 code06-06.py

- 메뉴를 선택하고 수량을 입력하면 계산한 금액 합계를 출력 하는 프로그램
- 주문을 끝내려면 메뉴를 '0'으로 입력
- 메뉴를 잘못 선택하면 프로그램을 종료하지 말고 다시 입력

①

```
01  menu = 0              # 선택 메뉴
02  number = 0            # 주문 수량
03  price = 0             # 상품 단가
04  total = 0             # 금액 합계
05
06  print("*" * 30)       # 메뉴 출력
07  print("[1]팝콘 [2]나쵸 [3]핫도그 [4]음료")
08  print("주문을 끝내려면 [0]을 입력하세요.")
09  print("-" * 30)
10
```

02. 반복문의 종류

II. 무한 반복과 반복의 제어

▪ 반복의 제어

실습 6-6 영화관 무인단말기로 메뉴 주문 후 종료하기 code06-06.py

①

```
11  while True :
12      menu = int(input("선택 메뉴 : "))
13      if menu == 0 :
14          break
15      if menu < 1 or menu > 4 :
16          print("메뉴 선택 오류...다시 선택하세요.\n")
17          continue
18
19      if menu == 1 : price = 5000          # 조건식과 명령문을 한 줄에 적기
20      elif menu == 2 : price = 4000
21      elif menu == 3 : price = 3500
22      else : price = 2000
23
24      number = int(input("주문 수량 : "))
25      total = total + (number * price)
26      print()          # 다른 메뉴를 선택하기 전에 빈 줄 추가
```

II. 무한 반복과 반복의 제어

▪ 반복의 제어

실습 6-6 영화관 무인단말기로 메뉴 주문 후 종료하기 code06-06.py

①

```
27
28  print("-" * 30)
29  print("금액 합계는", total, "원입니다.")
```

②

```
******************************
[1]팝콘 [2]나쵸 [3]핫도그 [4]음료
주문을 끝내려면 [0]을 입력하세요.
------------------------------
선택 메뉴 : 1
개수 입력 : 1

선택 메뉴 : 4
개수 입력 : 2

선택 메뉴 : 0
------------------------------
금액 합계는 9000 원입니다.
```

```
******************************
[1]팝콘 [2]나쵸 [3]핫도그 [4]음료
주문을 끝내려면 [0]을 입력하세요.
------------------------------
선택 메뉴 : 5
메뉴 선택 오류...다시 선택하세요.

선택 메뉴 : 3
개수 입력 : 1

선택 메뉴 : 0
------------------------------
금액 합계는 3500 원입니다.
```

02. 반복문의 종류

III. for 문

- for 문은 특정 횟수만큼 반복하기 위한 문장으로, range( ) 함수와 함께 많이 사용
- range( ) 함수의 인수로 사용하는 정수가 반복 횟수가 되고, 반복할 문장들을 들여쓰기로 입력

```
for 변수명 in range(반복 횟수) :
    반복할 문장들
```

(들여쓰기)

- range( ) 함수는 정해진 구간의 정수들(수열)을 생성하는 함수
- range(5)인 경우 0~4까지 5개의 값을 만들 수 있고, 반복이 실행될 때마다 값을 하나씩 반환

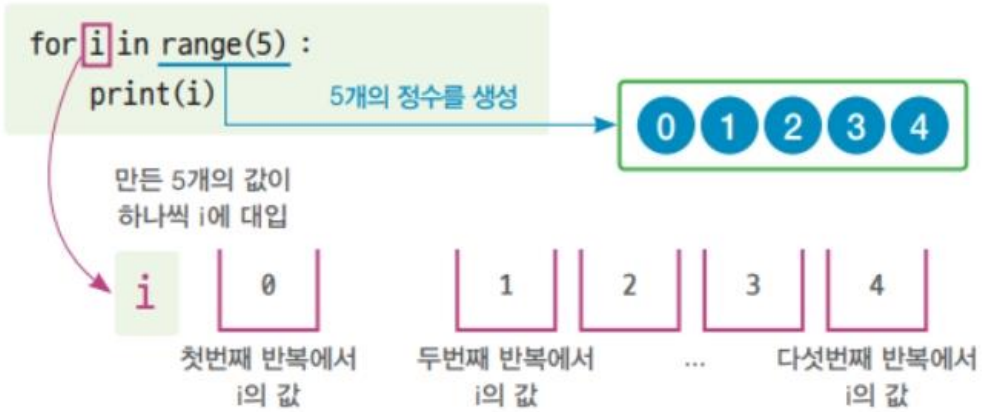


그림 6-10 for 반복문에서 range() 함수의 동작

02. 반복문의 종류

III. for 문

여기서 잠깐 **range() 함수**

- range() 함수는 다음 그림과 같이 3개의 인수를 가지고, start와 step은 생략할 수 있음
- 인수의 값에 따라 생성 되는 정수의 범위와 개수가 달라지며, start부터 시작해서 step 간격으로 증감하는 수열을 만들다가 stop이 되면 중지
- start를 생략하면 기본값은 0이고, step을 생략하는 경우 기본값은 1

```
range([start,] stop [,step])
```

02. 반복문의 종류

III. for 문

여기서 잠깐 **range() 함수**

- stop 값만 있는 경우 : range(5)는 start와 step이 모두 생략된 것으로, 0부터 4까지 하나씩 증가하는 정수들을 생성

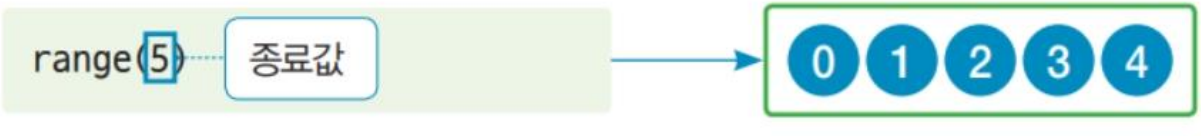


그림 6-11 stop 값만 있는 경우

- start, stop 값만 있는 경우 : range(1, 5)처럼 2개의 인수만 있는 경우, start는 1, stop은 5

그림 6-12 start, stop 값만 있는 경우

02. 반복문의 종류

III. for 문

여기서 잠깐 **range() 함수**

- start, stop, step 값이 모두 있는 경우 : range(1, 10, 2)는 1부터 9까지 2개씩 증가하는 정수

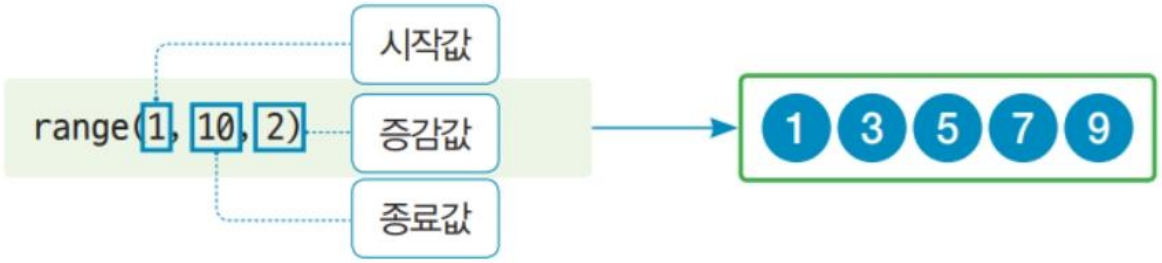


그림 6-13 start, step, stop 값이 있는 경우

- 감소하는 수열을 만드는 경우 : 9부터 시작해서 2씩 감소하는 수열을 3까지 만들 수 있음

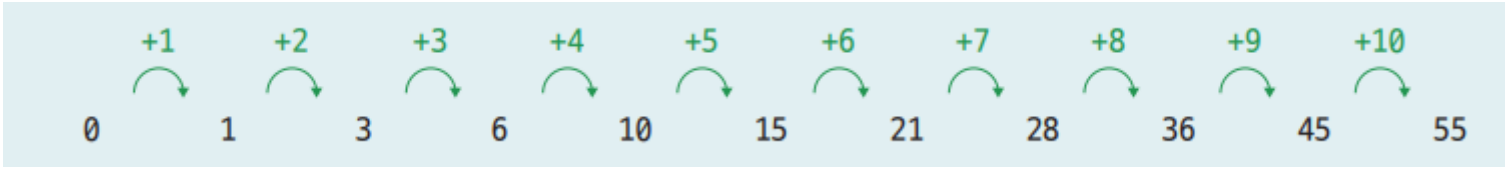
그림 6-14 감소하는 수열의 경우

02. 반복문의 종류

III. for 문

실습 6-7 for 문과 range() 함수로 규칙을 가진 숫자 출력하기 code06-07.py

- 0부터 10까지 정수를 차례대로 더한 값을 출력하는 프로그램을 for 문을 사용



①

```
01  s = 0                          # 합계를 저장하기 위한 변수
02
03  for x in range(0, 11):         # 0부터 10까지 반복
04      s += x                     # s = s + x와 동일, s가 x만큼 증가
05      print(s, end = " ")
```

②

```
0 1 3 6 10 15 21 28 36 45 55
```

✓ **TIP** 횟수 반복에는 for 문의 문장 구조가 더 간단하다는 장점이 있다. 반면 while 문은 조건 반복이나 무한 반복에 더 편리하게 사용할 수 있다.

02. 반복문의 종류

III. for 문

여기서 잠깐 **while 문과 for 문 비교**

- 두 반복문이 동일한 결과를 실 행하지만, 횟수 반복에는 for 문의 문장 구조가 더 간단하다는 장점이 있음
- 반면 while 문은 조건 반복이나 무한 반복에 더 편리하게 사용

```
01  x = 0
02  s = 0
03
04  while x <= 10 :
05        s = s + x
06        print(s, end = " ")
07        x = x + 1
```

02. 반복문의 종류

III. for 문

실습 6-8 구구단 프로그램 만들기 code06-08.py

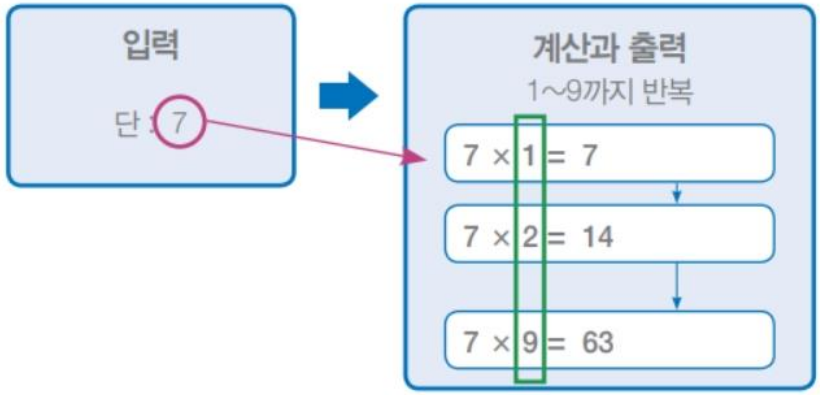


그림 6-15 구구단 출력하기

①

```
01  dan = int(input("몇 단을 출력할까요? : "))
02
03  for i in range(1, 10):
04      print("%d x %d = %2d" %(dan, i, dan * i))
```

②

```
몇 단을 출력할까요? : 7
7 x 1 =  7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
```

03

반복문의 활용

03. 반복문의 활용

I. 터틀 그래픽스 모듈

- 파이썬을 설치할 때 기본으로 제공되는 터틀 그래픽스라는 모듈을 이용하면 캔버스에 원하는 모양을 그리는 그래픽 프로그램을 간단하게 작성 가능

II. 터틀의 기본 사용법

- 좌표를 이용한 위치 이동뿐 아니라 픽셀pixel 단위로 거리를 지정하여 거북이를 움직일 수도 있음

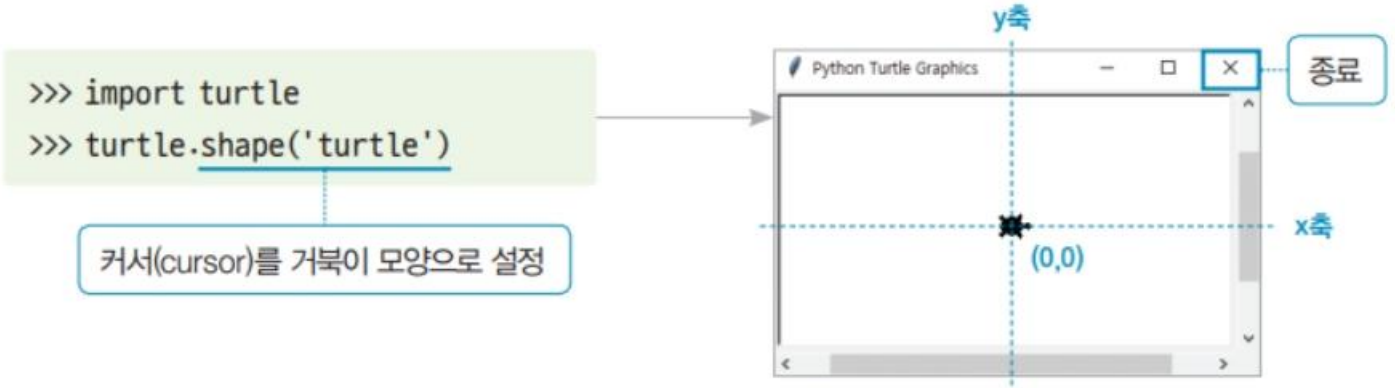


그림 6-17 거북이의 위치

03. 반복문의 활용

II. 터틀의 기본 사용법

실습 6-9 터틀을 이동시키면서 방향 바꾸기 code06-09.py

①

```
>>> import turtle
>>> turtle.shape('turtle')
>>> turtle.forward(100)        # 100픽셀 전진
>>> turtle.left(90)            # 왼쪽으로 90도 회전
>>> turtle.forward(100)
>>> turtle.right(90)           # 오른쪽으로 90도 회전
>>> turtle.forward(100)
```

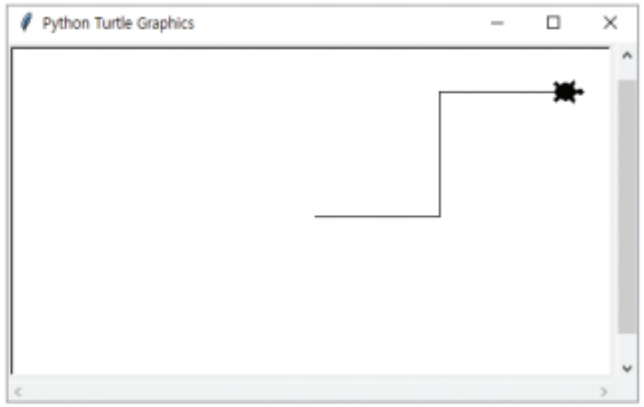


그림 6-18 거북이의 이동

②

```
>>> turtle.goto(0, 0)
```

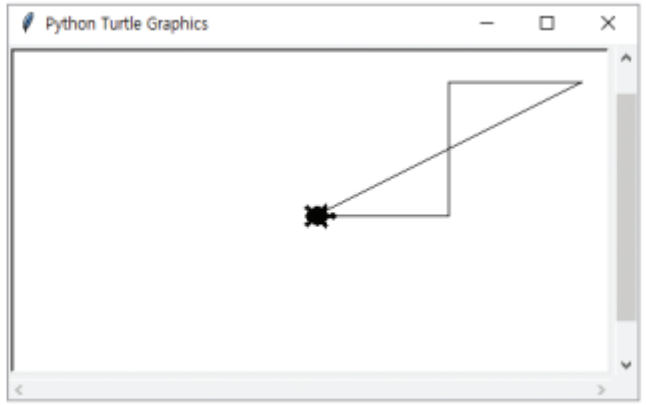


그림 6-19 거북이의 원위치

II. 터틀의 기본 사용법

실습 6-9 터틀을 이동시키면서 방향 바꾸기 code06-09.py

③

```
>>> turtle.penup()        # pu()와 동일
>>> turtle.fd(-100)       # back(100)과 동일
```

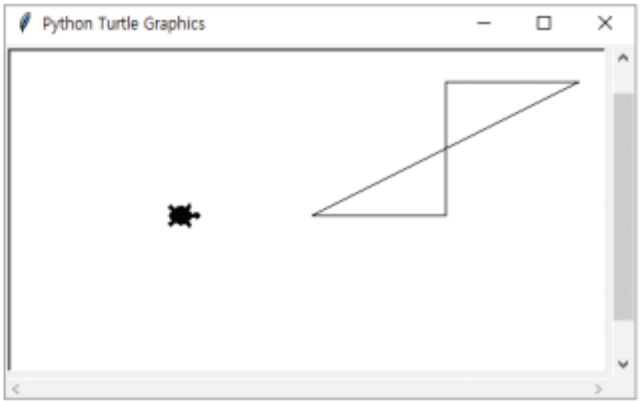


그림 6-20 거북이의 이동

④

```
>>> turtle.pendown()      # pd()와 동일
>>> turtle.fd(50)
```

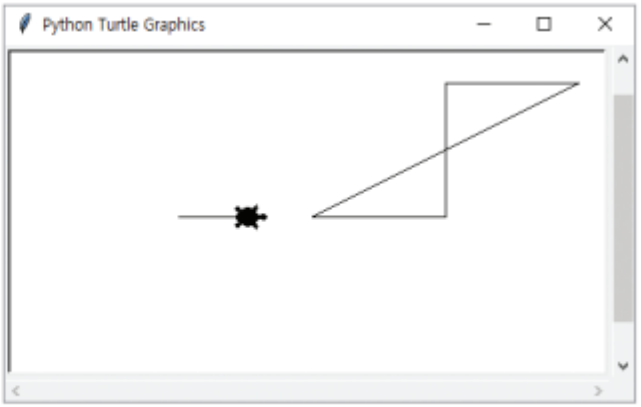


그림 6-21 다시 이동하며 그리기

II. 터틀의 기본 사용법

실습 6-10 터틀을 이용하여 정사각형 그리기 code06-10.py

- 터틀을 사용하여 한 변의 길이가 100 픽셀인 정사각형 그리기

① 터틀의 이동과 방향 전환 메소드를 이용하여 사각형을 그리는 코드를 작성해서 실행

```
01  from turtle import *
02
03  setup(300, 200)          # 캔버스 크기 설정하기(width, height)
04  fd(100)
05  lt(90)
06  fd(100)
07  lt(90)
08  fd(100)
09  lt(90)
10  fd(100)
```

II. 터틀의 기본 사용법

실습 6-10 터틀을 이용하여 정사각형 그리기 code06-10.py

- 터틀을 사용하여 한 변의 길이가 100 픽셀인 정사각형 그리기

② 만든 프로그램을 반복문으로 수정

```
01  from turtle import *
02
03  setup(300, 200)
04  for _ in range(4) :        # 변수가 필요 없을 때 밑줄 기호(_)를 사용
05      fd(100)
06      lt(90)
```

✓ **TIP** for 문에서는 range() 함수가 만드는 숫자들을 반환받는 변수를 사용

하지만, 반복문 안에서 변수를 사용할 필요가 없을 때 변수 자리에 밑줄 기호(underscore, '_')로 대치

III. 반복 구조의 다양한 그리기

실습 6-11 사각형 미로 만들기 code06-11.py

① 이동 거리를 10부터 시작해서 마지막은 100이 되도록 하고, 반복할 때 마다 5만큼 증가하는 for 문을 생성

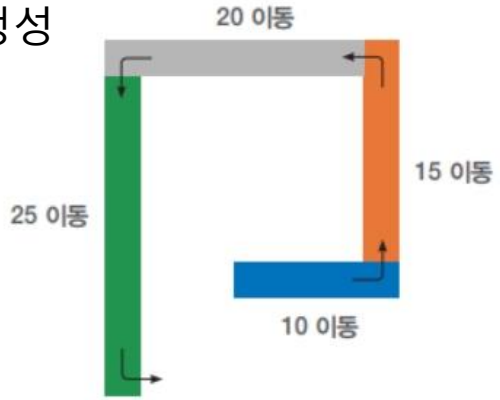


그림 6-24 반복되는 이동과 회전 동작

```
for x in range(10, 100, 5) :
    fd(x)          # x가 이동 거리
    lt(90)
```

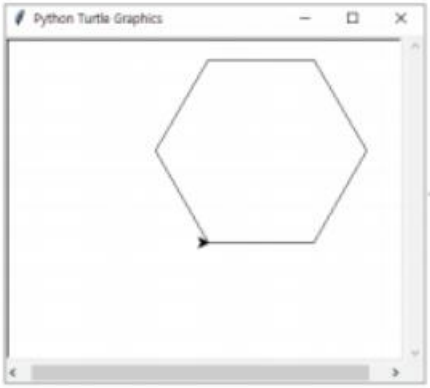
②

```
01 from turtle import *
02
03 setup(300, 200)
04 for x in range(10, 100, 5) :
05     fd(x)
06     lt(90)
```

III. 반복 구조의 다양한 그리기

실습 6-12 벌집 모양 그리기 code06-12.py

① 거북이를 이동시켜 한 변을 그리고 왼쪽으로 외각의 크기 (60)만큼 회전하는 동작을 6회 반복하면 처음 위치로 돌아옴



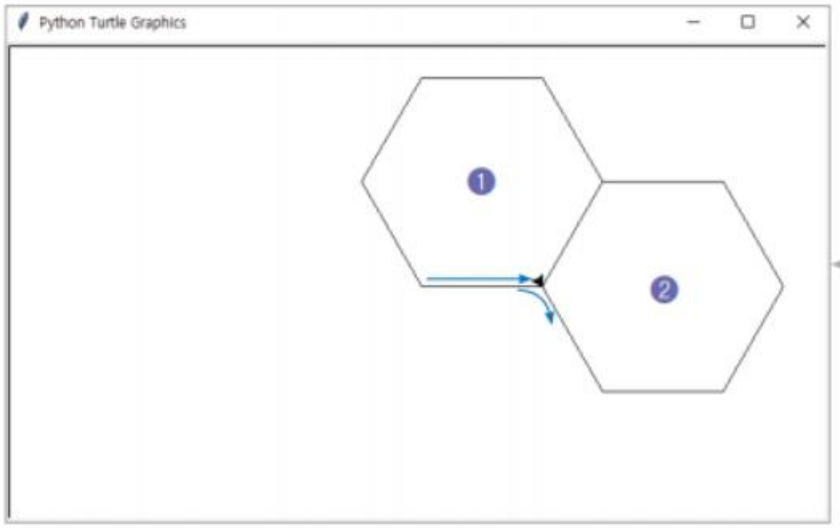

```
for _ in range(6):
    fd(100)
    lt(60)
```

그림 6-26 정육각형 1개 그리기

III. 반복 구조의 다양한 그리기

실습 6-12 벌집 모양 그리기 code06-12.py

② 시계 방향으로 다음 육각형을 그리기 위해, 거북이를 앞으로 이동하고 오른쪽으로 회전한 후 단계 ❶의 동작을 반복




```
for _ in range(6):    # ❶
    fd(100)
    lt(60)
fd(100); rt(60)
for _ in range(6):    # ❷
    fd(100)
    lt(60)
```

그림 6-27 정육각형 2개 그리기

III. 반복 구조의 다양한 그리기

실습 6-12 벌집 모양 그리기 code06-12.py

③ 기존 코드를 중첩 반복 구조로 다시 작성

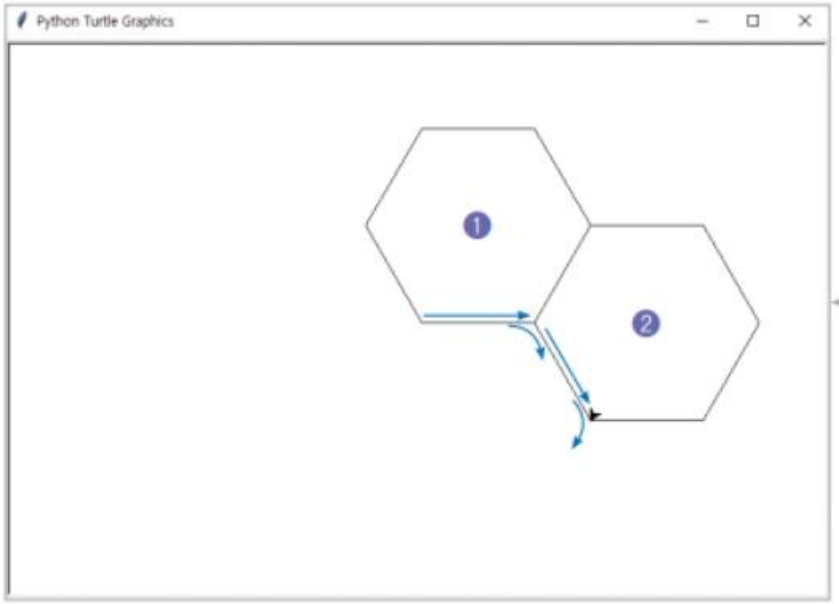


그림 6-28 정육각형 2개 그리기 반복

```
for _ in range(2):    # ❶, ❷
    for _ in range(6):
        fd(100)
        lt(60)
    fd(100); rt(60)
```

III. 반복 구조의 다양한 그리기

실습 6-12 벌집 모양 그리기 code06-12.py

④ 6개의 정육각형을 그리기 위한 전체 코드를 작성해서 실행

```
01  from turtle import *
02
03  setup(800, 800)
04
05  for _ in range(6):          # 6개의 정육각형을 위한 반복
06      for _ in range(6):      # 하나의 모양을 이루는 6개의 변을 위한 반복
07          fd(100)
08          lt(60)
09      fd(100); rt(60)         # 다음 모양을 그리기 위한 이동과 회전
```

들여쓰기 1회

III. 반복 구조의 다양한 그리기

실습 6-13 다양한 정다각형 그리기 code06-13.py

① 무한 반복문을 사용해 입력과 출력이 계속 실행되 도록 하고, 종료 조건은 다각형 모양을 0으로 입력하는 경우

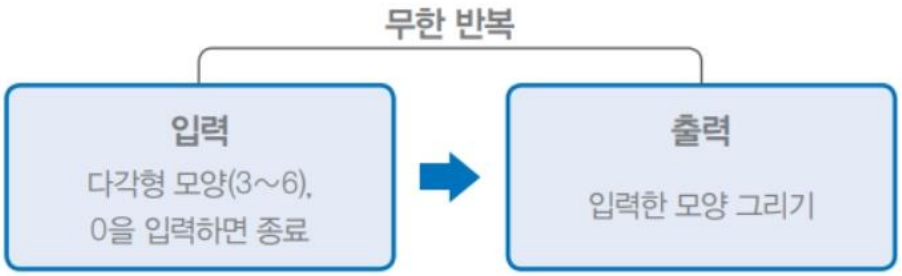


그림 6-30 입력과 출력 사이의 무한 반복

III. 반복 구조의 다양한 그리기

실습 6-13 다양한 정다각형 그리기 code06-13.py

② 코드를 입력하고 저장한 후 실행

```
01  from turtle import *
02
03  setup(500, 500)
04  pensize(10)                     # 펜 두께 설정
05  pencolor('red')                 # 펜 색상 설정
06
07  while True :                    # 무한 반복
08      shape = int(input("다각형 모양(3~6, 종료는 0) : "))
09      if shape == 0 : break       # 종료 조건
10      if shape < 3 or shape > 6 :   # 입력 오류
11          print("모양을 다시 입력하세요.")
12          continue
13
```

III. 반복 구조의 다양한 그리기

실습 6-13 다양한 정다각형 그리기 code06-13.py

② 코드를 입력하고 저장한 후 실행

```
14      clear()                     # 이전 그림 지우기
15      for _ in range(shape) :
16          fd(100)
17          lt(360 / shape)         # 회전 각도(=외각의 크기)는 360/꼭지점의 수
18
19  print("그리기를 종료합니다.")
20  bye()                           # 터틀 그래픽스 종료
```

III. 반복 구조의 다양한 그리기

실습 6-14 스크린 세이버 효과 만들기 code06-14.py

① 캔버스의 크기를 1200×800이라고 한다면 우측 상단의 좌표는 (600, 400)이 되고, 좌측 하단의 좌표는 (-600, -400)

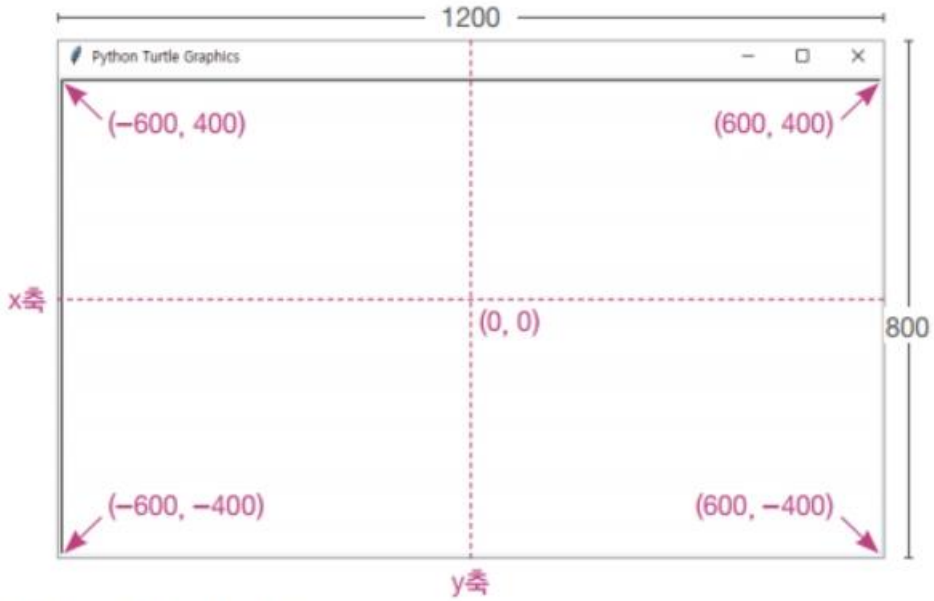


그림 6-32 캔버스의 좌표

III. 반복 구조의 다양한 그리기

실습 6-14 스크린 세이버 효과 만들기 code06-14.py

② 원의 좌표 뿐 아니라 원의 크기(반지름)도 랜덤하게 설정

```
from random import *
radius = randint(80, 130)          # 원의 반지름을 80에서 130으로 랜덤 선택
```

③

```
01  from turtle import *
02  from random import *
03
04  x = 0
05  y = 0
06  radius = 0
07
08  setup(1200, 800)
09  bgcolor("black")               # 창 색상 설정
10  speed(10)                      # 그리기 속도 설정
11  pensize(10)
12  color("white", "darkgray")     # 펜 색상과 채우기 색상 설정
13
```

03. 반복문의 활용

III. 반복 구조의 다양한 그리기

실습 6-14 스크린 세이버 효과 만들기 code06-14.py

③

```
14  for _ in range(30) :              # 30개의 원 그리기
15      x = randint(-500, 500)
16      y = randint(-400, 300)
17      radius = randint(80, 130)
18      penup()
19      goto(x, y)
20      pendown()
21      begin_fill()                  # 채우기 색상으로 원 채우기
22      circle(radius)
23      end_fill()
```

Thank You !

한빛아카데미 Hanbit Academy, Inc.