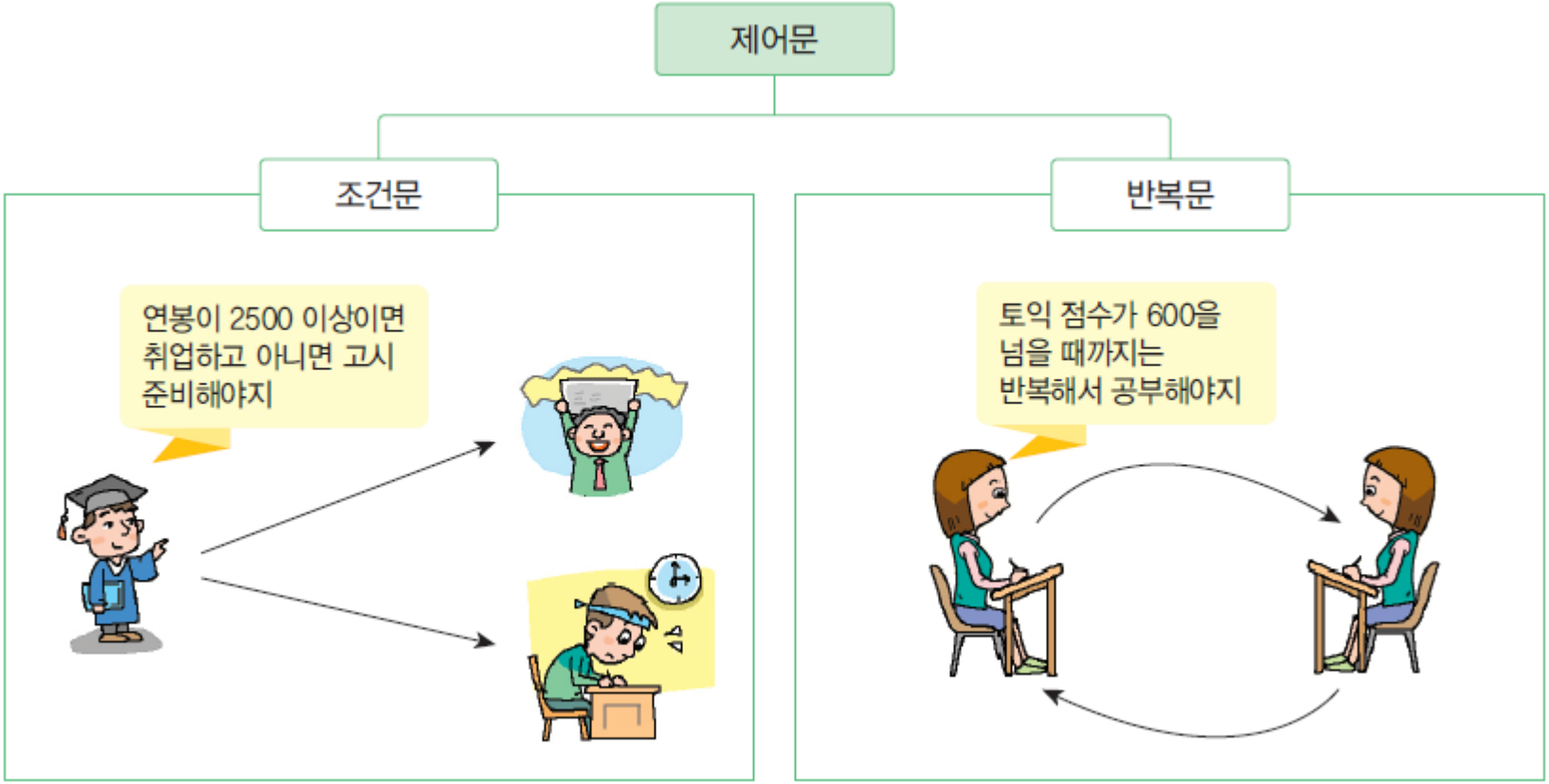
5장 반복문

제어문

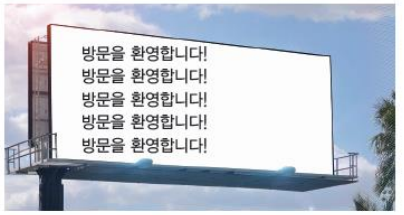

제어문
조건문
반복문
연봉이 2500 이상이면 취업하고 아니면 고시 준비해야지
토익 점수가 600을 넘을 때까지는 반복해서 공부해야지

왜 반복이 중요한가. p145

- 우리들의 생활에서는 반복적인 작업들이 필요하다. 반복적이고 단순한 작업은 컴퓨터를 이용하여 자동화하면 된다.
- 반복(iteration)은 같은 처리 과정을 되풀이 하는 것.

- 반복의 예 : 화면에 회사에 중요한 손님이 오셔서 대형 전광판에 "방문을 환영합니다!"를 5번 출력해야 한다고 가정하자.

```
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
```




```
for i in range(5):                                  # 아직 이해하지 않아도 된다!!
        print("방문을 환영합니다!")
```

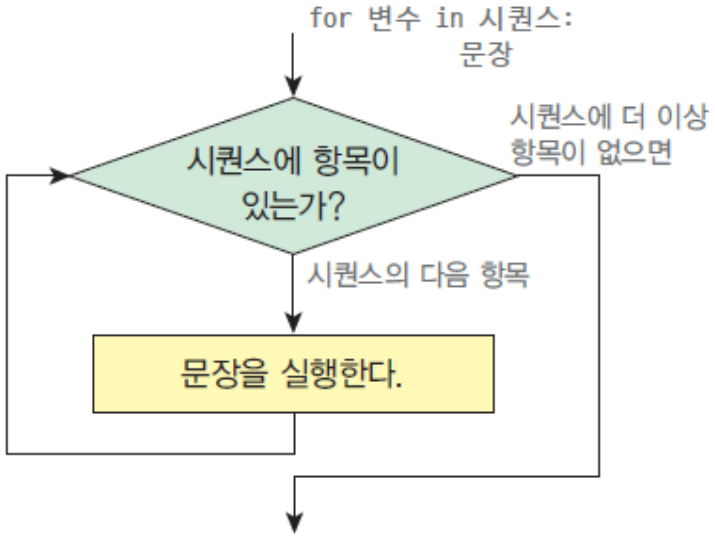
반복의 종류. p146

- 횟수 반복(for 문): 정해진 횟수만큼 반복한다.
- 조건 반복(while 문): 특정한 조건이 성립되는 동안 반복한다.

횟수 반복. p146

- 반복의 횟수를 미리 아는 경우에 사용한다.

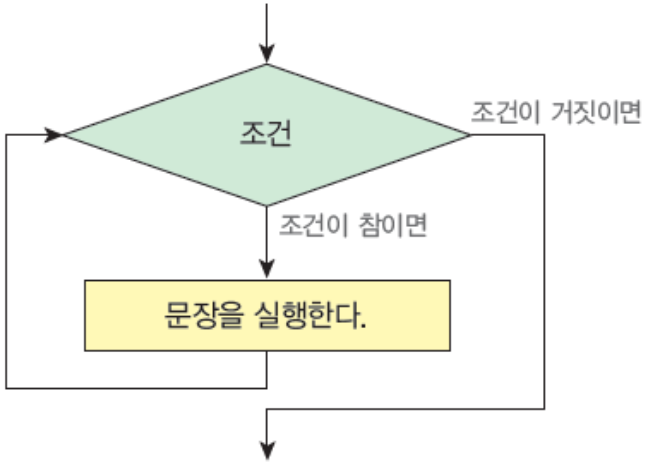
횟수 반복



조건 반복. p147

- 특정한 조건이 만족되는 동안 계속 반복한다.

조건 반복



리스트란. p147

- 리스트(list)
  - 여러 개의 자료들을 모아서 하나의 묶음으로 저장하는 데이터 구조
  - 반복 구조에서는 필수적으로 필요


```
slist = [ "영어", "수학", "사회", "과학“]
```

```
list1 = [1, 2, 3, 4, 5]                    # 숫자도 저장할 수 있다
```

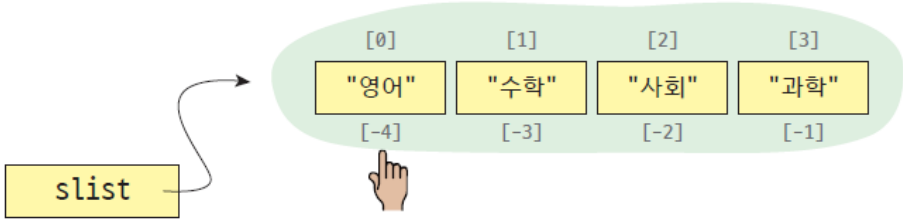
```
list = []                                  # 공백 리스트를 생성한 후에 값을 추가할 수도 있다.
list.append(1)
list.append(2)
list.append(6)
list.append(3)

print(list)                                # 리스트를 출력한다.
```

```
[1, 2, 6, 3]
```

리스트의 요소에 접근하기. p148

- 인덱스(index) : 리스트 안에 저장된 항목들에 0부터 시작하는 순차적인 번호

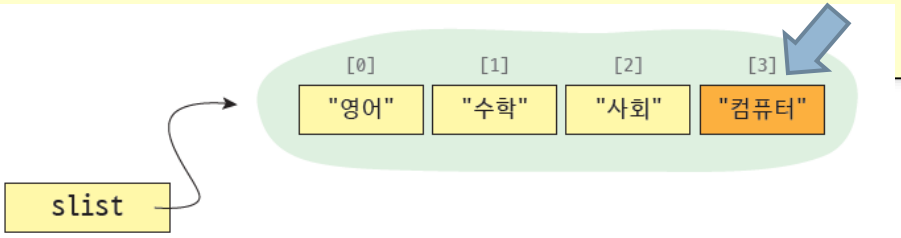
```
>>> slist = [ "영어", "수학", "사회", "과학" ]
>>> slist[0]
영어                                    # 리스트의 첫 번째 항목을 출력한다.
```



- 음의 인덱스는 리스트의 끝에서 부터 -1, -2, -3으로 매겨진다.

```
slist = [ "영어", "수학", "사회", "과학" ]
slist[-1] = "컴퓨터"
print(slist)
```

```
['영어', '수학', '사회', '컴퓨터']
```



중간점검. p149

1. [1, 2, 3, 4, 5]를 저장하는 리스트 myList를 생성해보자.
2. myList의 첫 번째 항목을 0으로 변경해보자.
3. myList의 마지막 항목을 9로 변경해보자.

횟수 제어 반복. p150

Syntax: for 문

형식
```
for 변수 in 리스트 :
    문장1
    문장2
```

예
```
for i in [ 1, 2, 3, 4, 5 ] :
    print("방문을 환영합니다.")
```

- 리스트의 값들이 하나씩 변수 i에 할당되면서 반복된다.
- 콜론(:)은 복합문을 의미한다.
- 반복되는 문장은 동일하게 들여쓰기가 되어야 한다.
- 반복되는 문장들

```
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
```

range() 함수. p152

- 리스트를 이용하는 앞의 방식은 반복 횟수가 크면 불가능한 방법
- range() 함수로 반복 횟수를 전달하면 range() 함수가 자동으로 순차적인 정수들을 생성해준다.

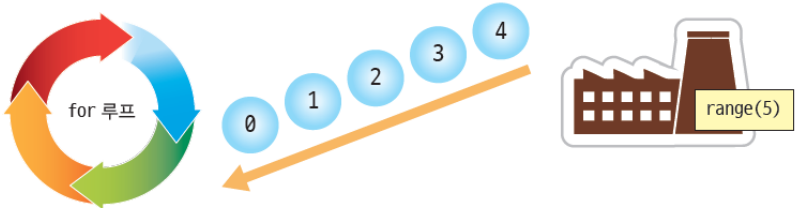
Syntax: range() 함수를 사용하는 for 문

형식
```
for 변수 in range(종료값) :
    문장1
    문장2
```

0, 1, 2, 3, 4의 값들이 생성되어서 변수 i에 할당된다.

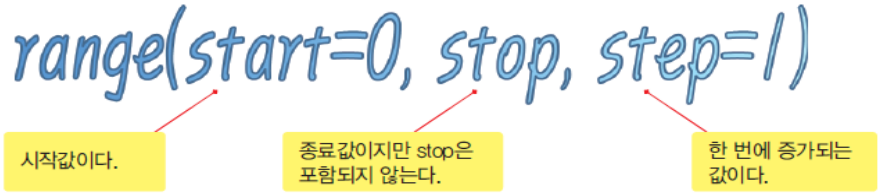
예
```
for i in range(5) :          # 0, 1, 2, 3, 4가 생성된다.
    print("방문을 환영합니다.")
```

반복되는 문장으로 들여쓰기 하여야 한다.



range() 함수. p153

- 일반적 형식




```
for i in range(1, 6, 1):              # 간격 지정
        print(i, end=" ")
```

```
1 2 3 4 5
```

```
for i in range(10, 0, -1):            # 역순
        print(i, end=" ")
```

```
10 9 8 7 6 5 4 3 2 1
```

range() 함수. p153

- 1부터 n까지의 합 계산 프로그램 (p.158 while과 비교)

```
sum = 0
n = 10
for i in range(1, n+1) :
        sum = sum + i
print("합=", sum)
```

```
합= 55
```

- 구구단 출력 프로그램

```
for i in range(1, 6) :
        print("9 *", i, "=", 9*i)
```

```
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
```

중간점검. p154

1. 다음 코드의 출력을 쓰시오.

```
for i in range(1, 5, 1) :
    print(2*i)
```

2. 다음 코드의 출력을 쓰시오.

```
for i in range(10, 0, -2) :
    print("Student" + str(i))
```

Lab 팩토리얼 계산하기. p155

for문을 이용하여서 팩토리얼을 계산해보자.

n! = 1 x 2 x 3 x ….. x (n-1) x n

```
정수를 입력하시오: 10
10!은 3628800이다.
```

```
n = int(input("정수를 입력하시오: "))
fact = 1
for i in range(1, n+1):
        fact = fact * i
print(n, "!은", fact, "이다.")
```

	i의 값	반복여부	fact의 값
1번째 반복	1	반복	1*1
2번째 반복	2	반복	1*1*2
3번째 반복	3	반복	1*1*2*3
4번째 반복	4	반복	1*1*2*3*4
5번째 반복	5	반복	1*1*2*3*4*5

도전문제. p155

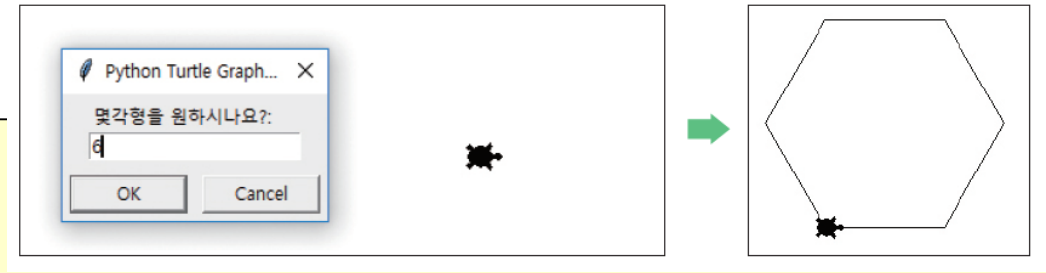
팩토리얼을 거꾸로 계산해보자.

$$n! = n \times (n-1) \times \cdots\cdots \times 2 \times 1$$

위의 프로그램을 어떻게 수정하여야 하는가?

Lab: n-각형 그리기. p156

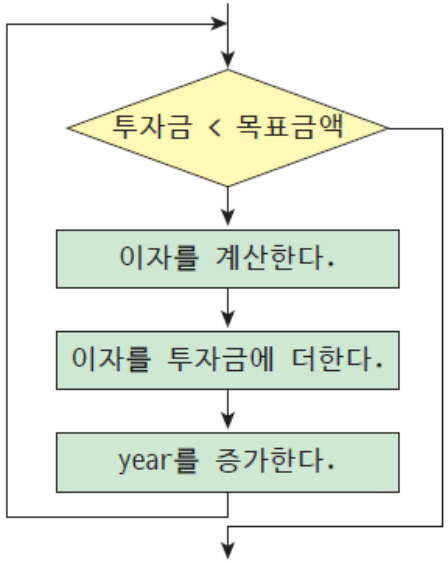
- 터틀 그래픽에서도 반복을 사용할 수 있다. 사용자로부터 정수 n을 받아서 n-각형을 그리는 프로그램을 작성해보자.




```
import turtle
t = turtle.Turtle()
t.shape("turtle")

s = turtle.textinput("", "몇각형을 원하시나요?:")
n = int(s)

for i in range(n):
        t.forward(100)
        t.left(360/n)

turtle.done()
```

조건 제어 반복. p157

- 어떤 조건이 만족 되는 동안 반복한다.
- (예) 1000만원의 투자금을 가진 사람이 있다고 하자. 투자금을 주식에 투자하여서 1년에 7%씩 버는 것을 목표로 하고 있다. 이 사람이 원금을 2배로 만들려면 기간이 얼마나 걸릴까?



조건 제어 반복. p157

Syntax: while 문

형식

```
while 조건식 :
        문장1
        문장2
```

참이나 거짓으로 계산되는 조건식,
관계 연산자 == != < > >= <=을 사용한다.

예

```
while money < TARGET :
        money = money + money * rate
        year = year + 1
```

콜론(:)은 복합문을 의미한다.

조건이 참이면 반복되는 문장들

반복되는 문장은 동일하게
들여쓰기가 되어야 한다.

Example: 투자금을 2배로 만드는 기간 프로그램. p158

```
TARGET = 2000                                   # 목표 금액
money = 1000                                    # 초기 자금
year = 0                                # 연도
rate = 0.07                             # 이자율

# 현재 금액이 목표 금액보다 작으면 반복한다.
while money < TARGET :
        money = money + money * rate
        year = year + 1

print(year, "년")
```

```
11 년
```

Example: 1과 10까지의 합 계산하기. p158

```
# 제어 변수를 선언한다.
i = 1
sum = 0

# i 값이 10보다 작으면 반복
while i <= 10 :
        sum = sum + i
        i = i + 1

print("합계는", sum)
```

```
합계는 55
```

p153. for와 비교

else가 있는 while 루프. p159

```
i = 0

while i < 3:
    print("루프 안쪽")
    i = i + 1
else:
    print("else 부분")
```

```
루프 안쪽
루프 안쪽
루프 안쪽
else 부분
```

Lab 사용자가 입력하는 숫자의 합 계산하기. p160

- 사용자가 입력한 숫자들을 더하는 프로그램을 작성해보자. 사용자가 yes라고 답한 동안에만 숫자를 입력받는다.

```
숫자를 입력하시오:  10
계속?(yes/no): yes
숫자를 입력하시오:  20
계속?(yes/no): no
합계는 :  30
```

Lab 숫자 맞추기 게임. p161

- 프로그램이 가지고 있는 정수를 사용자가 알아맞히는 게임이다. 사용자가 답을 제시하면 프로그램은 자신이 저장한 정수와 비교하여 제시된 정수가 더 높은지 낮은지 만을 알려준다.
- 정수의 범위를 1부터 100까지로 한정하면 최대 7번이면 누구나 맞힐 수 있다. 정수의 범위를 1부터 1,000,000까지 확대하더라도 최대 20번이면 맞출 수 있다. 이진탐색의 원리 때문이다. 정렬되어 있는 숫자 중에서 중간값과 한 번씩 비교할 때 마다 탐색의 범위는 ½로 줄어든다.

```
1부터 100 사이의 숫자를 맞추시오
숫자를 입력하시오: 50
너무 낮음!
숫자를 입력하시오: 75
너무 낮음!
숫자를 입력하시오: 89
축하합니다. 시도횟수= 3
```

Lab 산수 문제 생성 프로그램. p163

- 초등학생들을 위하여 산수 문제를 발생시키는 프로그램을 작성해보자. 한 번이라도 틀리면 반복을 중단한다.

```
25 + 78 = 103
잘했어요!!

3 + 32 = 37
틀려요. 하지만 다음번에는 잘할 수 있죠?
```

```
import random

flag = True
while flag:
    x = random.randint(1, 100)
    y = random.randint(1, 100)
    answer = int(input(f"{x} + {y} = "))
    if answer == x + y:
        print("잘했어요!!")
    else:
        print("틀려요. 하지만 다음번에는 잘할 수 있죠?")
        flag = False
```

Lab 로그인 프로그램. p164

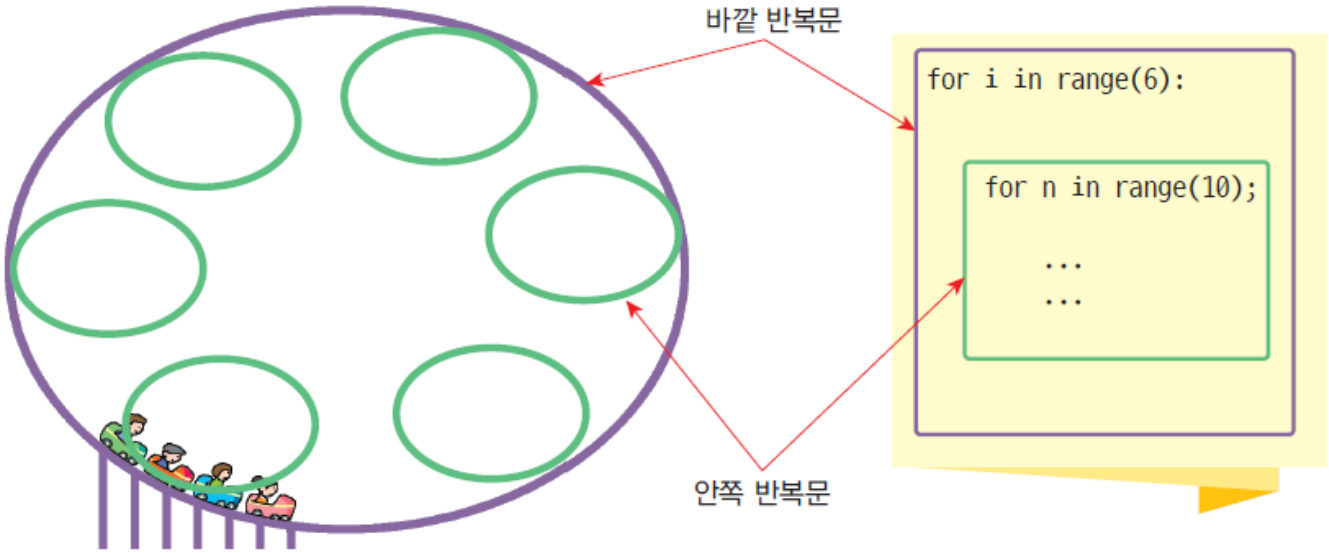
```
암호를 입력하시오: 12345678
암호를 입력하시오: password
암호를 입력하시오: pythonisfun
로그인 성공
```

```
password = ""
while password != "pythonisfun":
        password = input("암호를 입력하시오: ")
print("로그인 성공")
```

(직접 해보자) 루프를 5번 반복한 후 '로그인을 차단합니다' 메시지를 출력한 후 종료하도록 수정하자.

중첩 반복문. p165

- 반복 루프 안에 다시 반복 루프가 있을 수 있다.



Example. p166

- 사각형 패턴 출력하기

```
**********
**********
**********
**********
**********
```

```
for y in range(5) :
        for x in range(10) :
                print("*", end="" )
        print("")
```

- 삼각형 패턴 출력하기

```
*
**
***
****
*****
```

```
for y in range(1, 6) :
        for x in range(y) :
                print("*", end="" )
        print("")
```

Lab 주사위 합이 6이 되는 경우. p167

- 주사위 2개를 던졌을 때, 합이 6이 되는 경우를 전부 출력하여 보자.

```
첫 번째 주사위=1 두 번째 주사위=5
첫 번째 주사위=2 두 번째 주사위=4
첫 번째 주사위=3 두 번째 주사위=3
첫 번째 주사위=4 두 번째 주사위=2
첫 번째 주사위=5 두 번째 주사위=1
```

```
##
#          이 프로그램은 주사위 2개의 합이 6인 경우를 전부 출력한다.
#
for a in range(1, 7) :
    for b in range(1, 7) :
        if a + b == 6 :
            print(f"첫 번째 주사위={a} 두 번째 주사위={b}" )
```

Lab 모든 조합 출력하기. p168

- 다음과 같은 음료와 케이크가 저장된 두개의 리스트를 가지고 있다. 디저트 전문점에서 손님이 주문 가능한 모든 조합을 만들어 보자.

```
coffee = [ "Americano" , "Latte" , "Mocha" ]
cake = [ "CheeseCake" , "StrawberryCake", "ChocolateCake" ]
```

```
Americano CheeseCake
Americano StrawberryCake
Americano ChocolateCake
Latte CheeseCake
Latte StrawberryCake
Latte ChocolateCake
Mocha CheeseCake
Mocha StrawberryCake
Mocha ChocolateCake
```

```
coffee = [ "Americano" , "Latte" , "Mocha" ]
cake = [ "CheeseCake" , "StrawberryCake", "ChocolateCake" ]

for co in coffee:
    for ca in cake:
        print(co + " " + ca)
```

무한 루프와 break, continue. p169

- 무한 루프(infinite loop)
  - 프로그램이 무한히 반복하는 일
  - 무한 반복이 발생하면 프로그램은 빠져 나올 수 없기 때문에 문제가 된다
  - 신호등 제어 프로그램은 무한 반복을 사용하여야 함

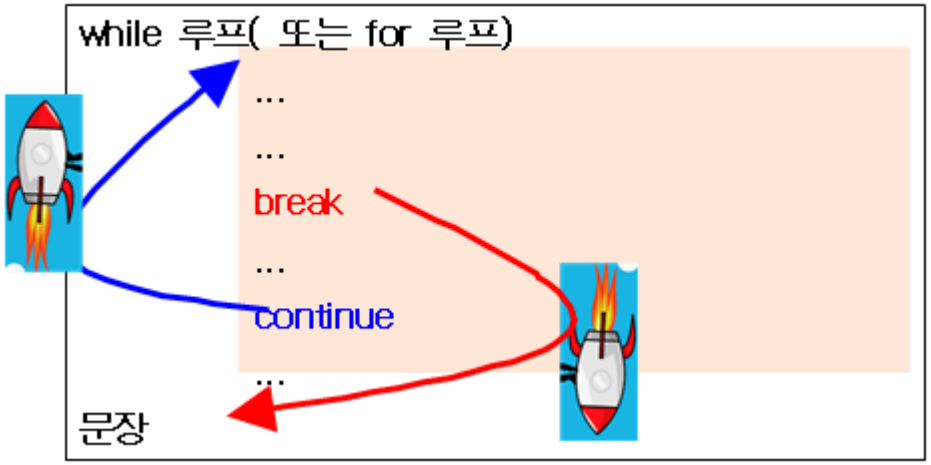
Syntax: 무한루프 문

형식

```
while True :
    if 조건 :
        break          # 반복을 중단한다.
    if 조건 :
        continue       # 다음 반복을 시작한다.
```

무한 루프와 break, continue. p169

- break : while 루프나 for 루프를 강제로 빠져 나올 때 사용
- continue : 강제적으로 반복 루프의 처음으로 돌아가게 한다.



Example 신호등의 색상이 녹색(green)이 될 때까지 대기하는 프로그램. p169

```
신호등 색상을 입력하시오 red
신호등 색상을 입력하시오 yellow
신호등 색상을 입력하시오 green
전진!!
```

```
while True:
        light = input('신호등 색상을 입력하시오 ')
        if light == 'green':
                break
print( '전진!!')
```

Example 1부터 10에서 3의 배수만 빼고 출력하기. p170

```
1 2 4 5 7 8 10
```

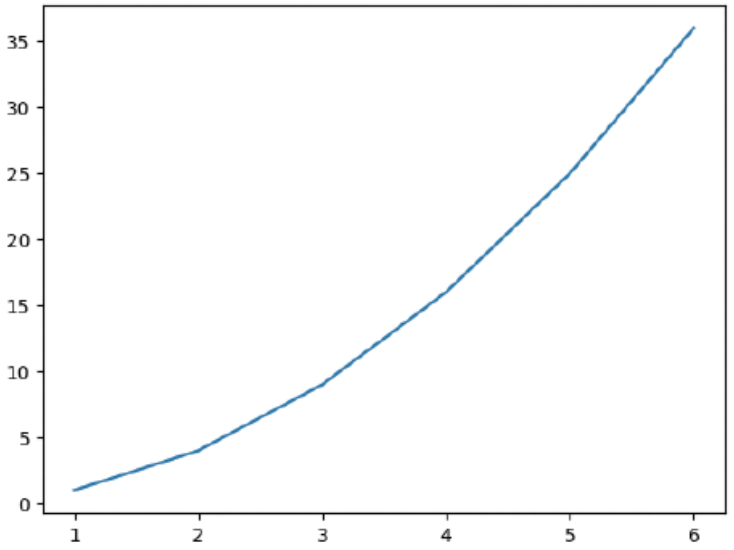
```
for i in range(1, 11):
        if i%3 == 0 :                   # 3의 배수이면
            continue # 다음 반복을 시작한다.
        print(i, end=" ")               # 줄바꿈을 하지 않고 스페이스만 출력한다.
```

Lab 파이 계산하기. p171

Lab 그래프 그리기. p173

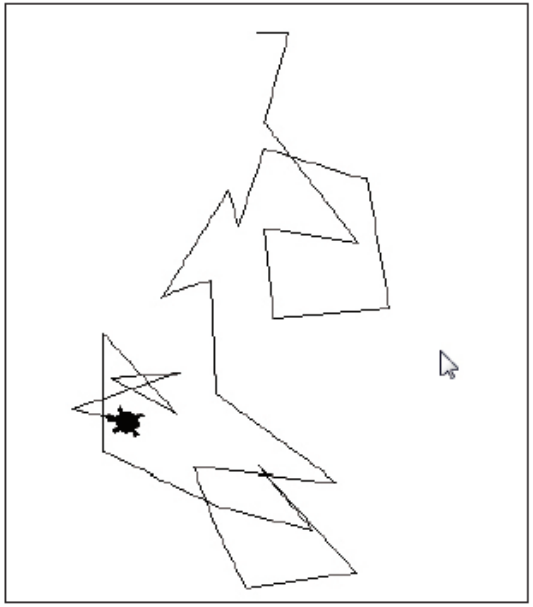
- 리스트에 저장된 데이터를 이용하여서 간단한 그래프를 그려보자. 그래프를 그리는 라이브러리는 Matplotlib이다.

```
import matplotlib.pyplot as plt

X = [1, 2, 3, 4, 5, 6]
Y = [1, 4, 9, 16, 25, 36]

plt.plot(X, Y)
plt.show()
```



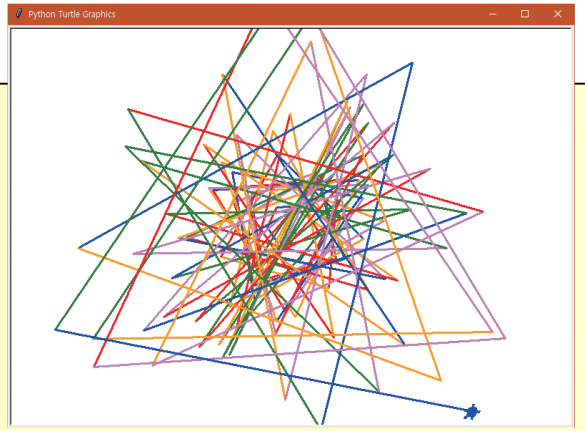
Lab 거북이 랜덤하게 움직이게 하자. p174

- 윈도우 상에서 거북이가 술에 취한 것처럼 랜덤하게 움직이게 하여 보자.



Lab 스파이럴 그리기. p176

- 색상을 변경하면서 거북이를 랜덤한 방향으로 전진시키고 회전시켜 보자.




```
import turtle
import random

t = turtle.Turtle()
t.shape('turtle')
t.width(3)
t.speed(7)
colors = ['orange', 'red', 'violet', 'green', 'blue']

for i in range(100):
    t.color(random.choice(colors))
    t.right(20 + i)
    t.forward(1 + (i * 6))
    t.right(30 + i)

turtle.done()
```

```
for i in range(100):
    t.color(random.choice(colors))
    t.forward(2+i/4)    # 다른 모습의 스파이럴
    t.left(30 – i/12)
```

Mini Project 사용자의 숫자 맞추기. p177

- **(직접 해보자)** 우리는 앞에서 컴퓨터가 숨기고 있는 숫자를 사람이 맞추는 프로그램을 살펴보았다. 이번에는 반대로 하여 보자. 사용자가 숨기고 있는 숫자를 컴퓨터가 알아맞히는 프로그램을 작성해보자. 게임 도중에 사용자는 숫자를 변경하면 안 된다.

```
컴퓨터가 당신이 생각하는 숫자를 알아맞히는 게임입니다.
하나의 숫자를 생각하세요.
컴퓨터가 제시한 숫자보다 정답이 높으면 high, 낮으면 low라고 하세요.
컴퓨터가 숫자를 맞추면 yes라고 하세요.

숫자가 50 인가요? low
숫자가 25 인가요? high
숫자가 37 인가요? high
숫자가 42 인가요? yes
```

연습문제. p179

Programming. p182